technische universität
dortmund

Dissertation

Object-Level Fusion for Surround Environment Perception in Automated Driving Applications

Michael P. Aeberhard

submitted on tt.mm.jjjjin fulfillment of the requirements for the degree of doctor of engineering (Dr.-Ing.) to the Department of Electrical Engineering and Information Technology of the Technische Universität Dortmund.

Advisors:
Univ.-Prof. Dr.-Ing. Prof. h.c. Dr. h.c. Torsten Bertram
Prof. N.N.

Technische Universität Dortmund
Department of Electrical Engineering and Information Technology
Institute of Control Theory and Systems Engineering
http://www.rst.e-technik.tu-dortmund.de/

Acknowledgments

Abstract

Driver assistance systems have increasingly relied on more sensors for new functions. As advanced driver assistance system continue to improve towards automated driving, new methods are required for processing the data in an efficient and economical manner from the sensors for such complex systems. The detection of dynamic objects is one of the most important aspects required by advanced driver assistance systems and automated driving. In this thesis, an environment model approach for the detection of dynamics objects is presented in order to realize an effective method for sensor data fusion. A scalable high-level fusion architecture is developed for fusing object data from several sensors in a single system, where processing occurs in three levels: sensor, fusion and application. A complete and consistent object model which includes the object's dynamic state, existence probability and classification is defined as a sensor-independent and generic interface for sensor data fusion across all three processing levels. Novel algorithms are developed for object data association and fusion at the fusion-level of the architecture. An asynchronous sensor-to-global fusion strategy is applied in order to process sensor data immediately within the high-level fusion architecture, giving driver assistance systems the most up-to-date information about the vehicle's environment. Track-to-track fusion algorithms are uniquely applied for dynamic state fusion, where the information matrix fusion algorithm produces results comparable to a low-level central Kalman filter approach. The existence probability of an object is fused using a novel approach based on the Dempster-Shafer evidence theory, where the individual sensor's existence estimation performance is considered during the fusion process. A similar novel approach with the Dempster-Shafer evidence theory is also applied to the fusion of an object's classification. The developed high-level sensor data fusion architecture and its algorithms are evaluated using a prototype vehicle equipped with 12 sensors for surround environment perception. A thorough evaluation of the complete object model is performed on a closed test track using vehicles equipped with hardware for generating an accurate ground truth. Existence and classification performance is evaluated using labeled data sets from real traffic scenarios. The evaluation demonstrates the accuracy and effectiveness of the proposed sensor data fusion approach. The work presented in this thesis has additionally been extensively used as the dynamics object detection platform for automated driving applications on highways in real traffic.

Contents

Abbreviations

ABS Anti-Lock Brakes.

ACC Active Cruise Control.

ADAS Advanced Driver Assistance Systems.

ALA Active Lane Assist.

ANIS Average Normalized Innovation Squared.

AUC Area Under the Curve.

BASt *Bundesanstalt für Straßenwesen*.

BBA Basic Belief Assignment.

BSD Blind Spot Detection.

CAN Controller Area Network.

DARPA Defense Advanced Research Projects Agency.

DSC Dynamic Stability Control.

DST Dempster-Shafer Evidence Theory.

EBA Emergency Brake Assist.

ESP Electronic Stability Program.

FCW Forward Collision Warning.

FISST Finite Set Statistics.

GDA Gaussian Discriminant Analysis.

GPS Global Positioning System.

IMM Interacting Multiple Model.

IPDA Integrated Probabilistic Data Association.

JIPDA Joint Integrated Probabilistic Data Association.

JPDA Joint Probabilistic Data Association.

LDW Lane Departure Warning.

LKA Lane Keeping Assist.

MHT Multiple Hypothesis Tracking.

NEES Normalized Estimation Error Squared.

NHTSA National Highway Traffic Safety Administration.

NIS Normalized Innovation Squared.

PDA Probabilistic Data Association.

PHD Probability Hypothesis Density.

RMSE Root Mean Squared Error.

ROC Receiver Operating Characteristic.

SAE Society of Automotive Engineers.

SVM Support Vector Machines.

TJA Traffic Jam Assist.

V2V Vehicle-to-Vehicle Communication.

VRU Vulnerable Road User.

List of Symbols

General Notation

a	Scalar
$\mathbf{a}$	Vector
$\mathbf{A}$	Matrix
$\mathbf{A}'$	Transpose of matrix $\mathbf{A}$
$\mathbf{A}^{-1}$	Inverse of matrix $\mathbf{A}$
$\hat{(.)}$	Estimate of the true value of (.)
$\tilde{(.)}$	Error between the estimate $\hat{(.)}$ and the true value (.)
$\bar{(.)}$	Complement of (.)
$(.)(k)$	Value of (.) at the discrete time step k
$(.)_k$	Value of (.) at the discrete time step k
$(.)(k \mid k)$	Value of (.) at the discrete time step k conditioned on information from the current time step k
$(.)_{k\|k}$	Value of (.) at the discrete time step k conditioned on information from the current time step k
$(.)(k \mid k-i)$	Value of (.) at the discrete time step k conditioned on information from a previous discrete time step $k-i$
$(.)_{k\|k-i}$	Value of (.) at the discrete time step k conditioned on information from a previous discrete time step $k-i$
$(.)(k, k-i)$	Transition from $k-i$ to k with (.)
$(.)(t)$	Value of (.) at the continuous time t
$\{(.)\}(t)$	Set of (.) at the continuous time t
$\{(.)\}_a^b$	Complete set of (.) from time a up to time b
Bel((.))	Belief function
BetP((.))	Pignistic transformation of (.)
$\mathrm{E}[(.)]$	Expected value of (.)

$F_{(.)}^{-1}$	Inverse cumulative distribution function for the (.) distribution
$m((.))$	Dempster-Shafer evidence theory basic belief assignment
Pl((.))	Plausibility function

Latin Letters

$\mathcal{C}$	Set of classification hypothesis
$\mathcal{D}$	Generic representation for some data
$\mathcal{E}$	Environment model
$\mathcal{G}$	Generic representation for spatial-based, or grid-based, objects/obstacles
$\mathcal{I}$	Generic representation for information
$\mathcal{I}^*$	Generic representation for a-priori information
$\mathcal{L}$	Log odds ratio
$\mathcal{M}$	Generic representation for digital map information
$\mathcal{O}$	Object list
$\mathcal{R}$	Generic representation for road infrastructure information
$\mathcal{S}$	Generic representation for data perceived from a sensor
$\mathcal{T}$	Training set
$\mathcal{U}$	Generic representation for control information from the host vehicle platform
$\mathcal{X}$	Generic representation for host vehicle localization and pose
$\mathbf{c}$	Classification vector of an object
$\mathbf{d}$	Dimension vector of an object
$\mathbf{d}_{\sigma^2}$	Dimension uncertainty vector of an object
$\mathbf{f}$	Feature vector of an object
$\mathbf{m}$	1-dimensional grid map for dimension estimation
$\mathbf{p}$	Position vector in a Cartesian coordinate system
$\mathbf{u}$	Host system control vector
$\mathbf{w}$	Normal vector to a decision boundary
$\mathbf{x}$	State vector of an object
$\mathbf{x}_\mathrm{a}$	State vector subset of $\mathbf{x}$ of an object used for association
$\mathbf{y}$	Attribute vector for classification

$\mathbf{z}$	Measurement vector
$\mathbf{A}$	Object association matrix
$\mathbf{B}$	Control transformation matrix
$\mathbf{C}$	Association cost matrix in the auction algorithm
$\mathbf{F}$	State transition matrix
$\mathbf{H}$	State-space transformation matrix
$\mathbf{I}$	Identity matrix
$\mathbf{K}$	Kalman gain
$\mathbf{S}$	Innovation covariance matrix
$\mathbf{P}$	Covariance matrix of a state estimate $\hat{\mathbf{x}}$
$\mathbf{P}^{ab}$	Cross-covariance between the state estimates $\hat{\mathbf{x}}^a$ and $\hat{\mathbf{x}}^b$
$\mathbf{P}_{ab}$	Cross-covariance matrices with retrodicted states
$\mathbf{Q}$	Process noise covariance matrix
$\mathbf{R}$	Covariance matrix of a measurement $\mathbf{z}$
$\mathbf{W}$	Kalman gain for an out-of-sequence measurement
a	Acceleration of an object
$a_{i,j}$	The element of the association matrix $\mathbf{A}$ in the ith row and jth column
w	Width of an object
b	Bias parameter
d	Geometrical dimension of an object
d^2	Mahalanobis distance
g	Boolean result from geometrical association
l	Length of an object
m	Single cell of the 1-dimensional map $\mathbf{m}$
$n_{\mathbf{a}}$	Number of elements, or dimension, of vector $\mathbf{a}$
p	Bid price for assignment in the auction algorithm
r	Range in a polar coordinate system
v	Velocity of an object
x	Position of an object on the x-axis in a Cartesian coordinate system
y	Position of an object on the y-axis in a Cartesian coordinate system
C_i	Object class i, where i corresponds to the ith element of $\mathbf{c}$ or $\mathcal{C}$

D^2	Extended Mahalanobis distance
G	Gating threshold during object association
H	Object association hypothesis
O_i	The ith object in an object list $\mathcal{O}$
Z^i	List of measurements from sensor i

Greek Letters

γ	Dempster-Shafer evidence theory prediction weight
δ	Offset/translation of a sensor's placement on the vehicle
$\Delta(.)$	Difference of (.) between two values
ϵ	Normalized Estimation Error Squared
η	Normalization factor
θ	Rotation of a sensor's placement on the vehicle along the vertical axis
Θ	Dempster-Shafer evidence theory frame of discernment
λ	Rate parameter of a Poisson process
μ	Mean
ρ	Correlation weighting factor
σ	Standard deviation
ϕ	Angle in a 2-dimensional polar coordinate system
ψ	Orientation angle of an object
$\dot{\psi}$	Orientation velocity of an object
ω	Covariance intersection weighting factor

Subscripts and Superscripts

$(.)^{\mathrm{S}_i}$	Value of (.) originates from sensor S_i
$(.)^{\mathrm{G}}$	Value of (.) result from a global fusion algorithm
$(.)^{\mathrm{obj}}$	Value of (.) in the object coordinate system
$(.)^{\mathrm{sensor}}$	Value of (.) in the sensor coordinate system
$(.)^{\mathrm{veh}}$	Value of (.) in the host vehicle coordinate system
$(.)_{\mathrm{f}}$	Value of (.) for the feature f
$(.)_x$	Scalar corresponding to the x component of (.) in a Cartesian coordinate system

$(.)_y$	Scalar corresponding to the y component of $(.)$ in a Cartesian coordinate system
$(.)_{\mathrm{a}\to\mathrm{b}}$	Transformation from a to b

Probabilities

$p(a)$	Continuous probability density function of the random variable a
$p(a \mid b)$	Continuous conditional probability density of the random variable a conditioned on b
$p(\exists \mathbf{x})$	Existence probability of an object
$p(\nexists \mathbf{x})$	Non-existence probability of an object
p_{b}	Birth probability
p_{c}	Clutter probability
p_{d}	Detection probability
p_{p}	Persistence probability
p_{trust}	Trust probability
$P_{(.)}$	Scalar probability value

1 Introduction

Since the introduction of automobiles in the early 1900s, vehicle manufacturers have continuously worked to increase the safety of their vehicles. The first barrier crash tests were performed in the 1930s, the three-point lap and shoulder seat belt was invented in the late 1950s and become standard with new vehicles in the 1960s, padded instrument panels were widely adopted in the 1960s, the first Federal Motor Vehicle Safety Standards took effect in the United States in 1968 and, arguably one of the most important vehicle safety devices, the air bag, was introduced in the 1970s and became widely adopted in the 1980s and early 1990s. The aforementioned safety systems, such as air bags, seat belts or improved vehicle structure, are examples of *passive safety* systems, meaning that they protect the driver and the passengers during the event of a crash, reducing bodily injuries.

Beginning in the 1980s, electrical and computer systems in vehicles became more capable and new *active safety* systems were introduced. Active safety systems aim to alleviate, lessen the consequences or altogether prevent a collision before it even occurs. One of the first active safety systems was Anti-Lock Brakes (ABS), which prevents wheel lock-up during emergency braking, effectively increasing vehicle stability during braking maneuvers [228, 278]. Sensor technology, such as wheel speed sensors, combined with new computer systems which could actively and independently in real-time control the brakes, allowed such systems to be conceived. Electronic Stability Program (ESP), or Dynamic Stability Control (DSC), takes ABS technology a step further by helping prevent oversteer and understeer instabilities of the vehicle using intelligent and electronically controlled brake intervention [228, 278]. Today, ABS and DSC systems are standard equipment in most production vehicles.

Active safety systems such as ABS and DSC are all a form of electrical and computer controlled automation of the vehicle. With such systems activated, the driver's inputs, or lack thereof, may be overridden or intervened by an electrical system when an imminent danger is detected, such as wheel lock in ABS or oversteer in DSC. More recently, such electrical and computer controlled systems have been introduced as comfort applications, helping the driver not only in critical situations, but also during everyday driving scenarios. One of the first examples of an electrical comfort system is cruise control, which allows the vehicle to maintain a driver-defined speed by automatically controlling acceleration and braking. Comfort and active safety systems can all classified as *driver assistance systems*. The late 1990s saw the introduction of Active Cruise Control (ACC), where the typical cruise control system is supplemented with an environment-sensing sensor, such as a radar, in order to automatically adapt the vehicle's speed and keep a safe distance to the vehicle in front.

Automotive research in this area has recently focused on *Advanced Driver Assistance Systems (ADAS)*, which use more complex environment-sensing sensors [200] to help the driver in certain situations or even take over complete control of the vehicle, such as in highly automated driving systems. The ultimate ADAS is an autonomous driving system, where the vehicle can take the driver and its passengers to their destination completely

autonomously, with no driver intervention at all. A great overview of ADAS can be found in [228, 303].

This thesis focuses on advancing the methods of environment perception technology for ADAS, particularly those necessary for automated driving applications. This introductory chapter will focus on the motivation of such technology for improving driving safety and comfort and give a brief overview of the state of the art of such technology for varying degrees of vehicle automation. The last section of this chapter gives a description of the contribution of this thesis and an overview of the structure of the remaining chapters.

1.1 Motivation

Advances in passive safety systems, such as the air bag, have decreased the number of fatal injuries due to traffic accidents. However, despite these advances, traffic accidents are still very common. The challenge of the future will be to reduce the number of traffic accidents that occur, requiring a preventive strategy for accident avoidance. In order to avoid an accident in the first place, active safety and assistance systems must be developed, as opposed to the passive systems which simply reduce the severity of an accident after it has already occurred.

The key to accident prevention is to understand the underlying cause of most accidents. The basic fact of the matter is that today's vehicles are operated solely by people, and people tend to make mistakes. Accident statistics show that driver error is the main cause of traffic accidents today. One of the main reasons for driver error is driver distraction while driving. A study by the US Department of Transportation's National Highway Traffic Safety Administration (NHTSA) showed that in 2011 10% of all fatal crashes were due to driver distraction [204]. Further studies have shown that conversing with a passenger to be the most common type of distraction before an accident [203]. In a 2008 report to the US Congress, data from 2005 to 2007 showed that a large portion of investigated crashes could be attributed to driver error, where various reason were identified, such as recognition errors, decision errors or an internal non-driving activity [202]. Internationally, a report by the United Nations Economic Commission for Europe shows that between 1998 and 2008, the general trend of traffic accidents and fatalities has decreased from 4 million road accidents in 1998 to about 3.5 million road accidents in 2008 [275]. This trend would need to continue for the next 70 years in order to reach an accident-free society. Despite this trend in the reduction of accidents, another fact is that the number of vehicles and road users is continuously on the rise worldwide. In the United States, for example, the number of registered vehicles has increased by more than 50 million in the last 20 years [60]. It remains to be seen if the increase of the number of vehicles on the road will continue the decreasing trend of traffic accidents or if it may actually result in a reversal of the trend, if technology does not develop and adapt to the traffic of tomorrow.

The solution to an accident-free society, despite the increasing number of vehicle on the road, is to equip tomorrow's vehicles with new technologies aimed at preventing accidents. Only in the recent 10-15 years have driver assistance systems been on the market, mainly in premium vehicles. It is still too early to tell if these systems will have an impact on traffic safety, as the penetration rate of such systems is still far too low. The trend, however, is that assistance systems are becoming cheaper and more available in all vehicle segments. Initial studies have shown that systems such as ACC and Forward Collision

Warning (FCW) have a positive effect on traffic safety [32]. Rating agencies for vehicle safety, such as Euro NCAP, are already including active safety systems, such as FCW, in their safety assessments and ratings [94]. Due to these activities, it can be assumed that in the near future, some active safety systems will become standard equipment on most vehicles, just like the air bag and ABS.

The pinnacle of active safety is autonomous driving. Vehicles which are developed to function completely without the intervention of the driver have the potential of eliminating all driver-related accidents which are common today. In order to realize this goal, practically all automobile manufacturers are working on autonomous driving technology. In addition to reducing accidents, autonomous driving may also lead to significant financial savings for our society [243]. In the United States alone, a total of $1.3 trillion per year could be saved if autonomous vehicles reach full penetration, where the savings are calculated based on accident avoidance savings, fuel savings and productivity gains [243]. It is clear that autonomous driving could have many benefits to society. However, many technological problems and legislative issues still need to be solved in order to make autonomous driving a reality.

Apart from the gains in safety, the increased comfort gained from assistance systems is also major selling point for automobile manufacturers. Driver assistance systems, which were once only available on high-end sedans, such as park distance control, are now mainstream features which are well-received by customers. Rear-view camera systems and active cruise control are also example of assistance systems which increase comfort in different driving situations. This trend will most likely continue in the future, as customers expect technology to aid them in daily driving situations.

One of the key technological components of active safety and assistance systems is the sensors which perceive the environment. Common sensor technologies in production vehicles are radar [301], camera [254], lidar [114] and ultrasonic sensors [207], which are mainly used for detecting other vehicles and obstacles or lane markings and road boundaries. Today's driver assistance systems have fairly basic sensor processing architectures - a single sensor is used for one or more applications and another sensor for other applications. As driver assistance systems become more complex, edging towards autonomous driving technology, this simple sensor processing architecture will not suffice. It will be necessary to combine the data from all of the vehicle's sensors into a single description of the environment, called an *environment model*, and making this information available to all assistance and safety applications in the vehicle, thereby increasing efficiency and reducing the costs of the overall system [75, 137, 201]. The combination of data from many sensor sources is refereed to as *sensor data fusion* and is the main topic of this thesis. Figure 1.1 shows the difference between these two basic architectural concepts for the processing sensor data.

Before diving into the details of the proposed sensor data fusion concepts and algorithms in this thesis, an overview of driver assistance and safety systems with respect to their level of automation is given in the following section.

1.2 Automation in Driver Assistance and Safety Systems

In recent years, an increase in the level of automation in vehicle systems has emerged. Systems such as ABS have laid the foundation and showed the benefits of electrical automation in vehicles. With the increase of computational power and advances in sensor technology,

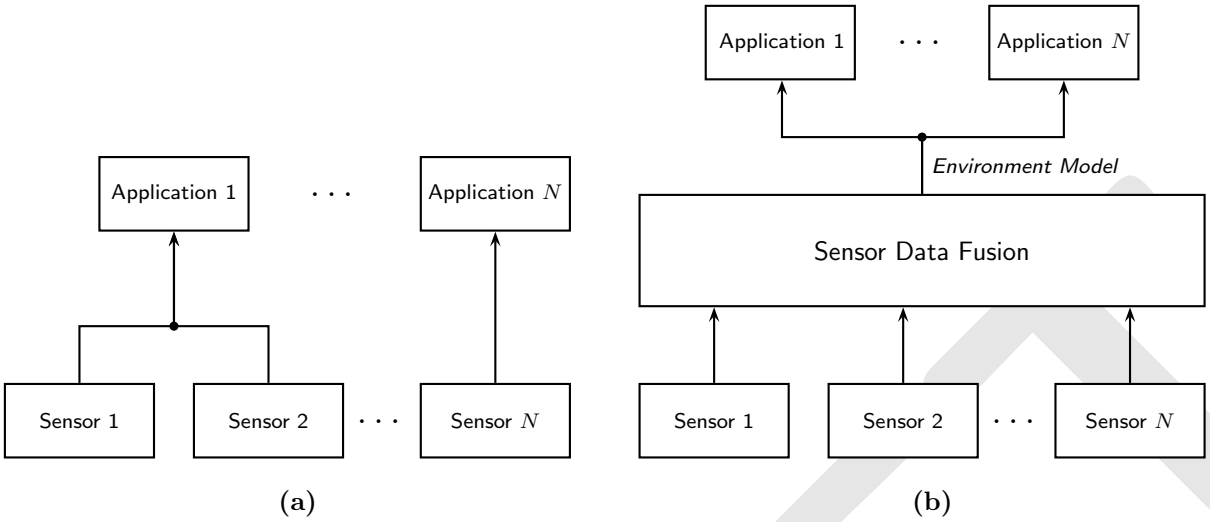


Figure 1.1: Basic architectures for sensor data processing: direct sensor data processing by an ADAS application (a) versus sensor data processing through a common sensor data fusion module (b).

higher levels of automation are being developed and will be possible in the future. However, legislation around the world currently requires that a driver be in control of a vehicle at all times [273, 274]. These current laws make it difficult to introduce vehicles with a higher degree of automation [248]. In the United States, several states, such as Nevada [205], California [61], Florida [97] and Michigan [179], have already passed legislation to pave the way for a safe introduction and testing procedures of autonomous vehicles. In Germany, the Federal Highway Research Institute (*Bundesanstalt für Straßenwesen* (BASt)) have organized a project group to investigate the legal consequences of increasing automation in vehicles [112]. As of 2010, this project group has defined a preliminary classification of different levels of automation in vehicles and driver assistance systems [111, 112]. In the United States, similar definitions of vehicle automation from NHTSA and their legal impact have also been investigated [304] and similar results compared to the BASt studies and definitions have been reached. In 2014, the Society of Automotive Engineers (SAE) have released a standard, SAE J 3016 [231], which also formally define the automation levels. The SAE standard differs slightly in the nomenclature with regards to the BASt definitions, but the implications of the automation levels are the same. Table 1.1 gives a brief overview of the SAE automation levels along with a description of an example system for each level. SAE J 3016 also provides a table describing the various responsibilities of the automated vehicle and the driving for each of the automation levels. In the recent years, the SAE automation level definitions have become internationally well accepted and will therefore be used in this section when describing driver assistance and automated driving systems.

In this section, current and past projects will be discussed with respect to the automation levels described in Table 1.1 in order to present an overview of the state of the art of driver assistance systems and automated driving. For each automation level and project, the technology used for environment perception will be mentioned, as this is the main focus of the thesis.

Table 1.1: Vehicle automation levels, as defined by SAE J 3016 [231].

Automation Level	Brief Description	Example System
0 - No Automation	The driver continuously carries out longitudinal (acceleration and braking) and lateral (steering) control of the vehicle.	No driver assistance system is active which intervenes in the longitudinal and lateral control. Safety systems, such as ABS and DSC, or warning systems may still be active.
1 - Driver Assistance	The driver continuously carries out *either* longitudinal *or* lateral control of the vehicle. The driver must continuously monitor the system and driving environment and at any time be prepared to take over full control of the vehicle.	Adaptive Cruise Control (ACC): The vehicle takes over longitudinal control and is able to automatically keep a safe distance to the vehicle in front.
2 - Partial Automation	The system takes over longitudinal *and* lateral control of the vehicle under certain conditions and situations and for a limited amount of time. The driver must still permanently monitor the system and driving environment and be prepared to take over control of the vehicle at any time.	Highway or Traffic Jam Assistant: Vehicle automatically carries out longitudinal and lateral control on the highway or in a traffic jam up to a certain speed. The driver permanently monitors the system and takes over full control when required to do so.
3 - Conditional Automation	The systems takes over longitudinal and lateral control of the vehicle under certain conditions and situations and for a limited amount of time. The driver *need not* permanently monitor the system nor the driving environment when it is activated, but must still serve as a fallback and take over control in case the automation fails or reaches a limit.	Highway Pilot: The vehicle takes over longitudinal and lateral control up to a certain speed. The driver need not permanently monitor the system, but must be able to take over control upon request within a certain time frame.
4 - High Automation	The systems takes over longitudinal and lateral control of the vehicle under certain conditions and situations and for a limited amount of time. In case the driver fails to take over during a system failure or limit, the vehicle will automatically initiate a maneuver to bring the vehicle to a minimum risk condition (this is the main difference to Level 3 conditional automation).	Remote Valet Parking: Performs all driving tasks at low speeds in a parking environment, autonomously driving the vehicle from a start position to an available parking spot. A driver is not physically necessary, as the vehicle will simply come to a stop in case of a system failure or limit.
5 - Full Automation	The system completely takes over longitudinal and lateral control of the vehicle during all conditions, all driving situations and at all times. In the event of a system failure or limit, the systems automatically brings the vehicle to a minimum risk condition.	Automated Taxi: The vehicles completely takes over longitudinal and lateral control of the vehicle during the complete journey, without any intervention or monitoring necessary by any of the passengers (a driver is most likely not present).

1.2.1 Level 0 - No Automation

When no driver assistance system is activated which actively takes longitudinal or lateral control of the vehicle, then the vehicle is said to be in *no automation* mode, where the driver must assume direct and complete control of the vehicle. However, active safety systems such as ABS and DSC may still be active and will still have the ability to take control during critical situations when the vehicle is in an unstable state.

Driver assistance systems which are active when the driver is in control usually inform or warn the driver visually or haptically of a potential hazard. Today's production vehicles include many such driver assistance systems, such as Lane Departure Warning (LDW), which alerts the driver when an unintentional lane departure is detected [286], or Blind Spot Detection (BSD), which informs the driver of the presence of other traffic in the blind spot during a lane change [27]. These and other warning systems all depend on environment perception sensors for detecting certain situations and then communicating potential hazards to the driver. Typical sensors for such systems are cameras, for detecting lanes or speed limit signs, radar, for detecting other traffic or ultrasonic sensors, for detecting obstacles near the vehicle, such as during parking. Each driver only assistant systems for the most part uses just one sensor, due to their simplicity and low requirement on reliability, as the driver is always fully in control. Table 1.2 gives an overview of such driver assistance systems which warn or inform the driver, along with the sensor technology each system is based on.

Table 1.2: Overview of current driver assistance systems which inform or warn the driver along with the sensor technology used.

System	Brief Description	Sensor
Lane Departure Warning	Warns the driver in the event that the vehicle is imminently about to unintentionally depart the lane.	Camera
Blind Spot Detection	Informs the driver, usually visually on the side view mirror, when other traffic is in the vehicle's blind spot, preventing a possible lane change.	Radar
Forward Collision Warning	Acoustically warns the driver of an imminent impact if immediate braking intervention is not initiated.	Radar
Park Distance Control	Acoustically informs the driver about obstacles around the vehicle during parking maneuvers.	Ultrasonic
Parking Surround and Rear View	Gives the driver a visual reference of the vehicle's surrounding during parking maneuvers.	Camera
Night Vision	Provides an enhanced visual view at night, sometimes with an integrated pedestrian warning system.	Thermal camera

Despite not making an active intervention in the longitudinal or lateral control of the vehicle, driver assistance systems which help the driver with warning signals have been proven to increase safety. The euroFOT project, the first large-scale field test evaluating driver assistance systems, collected data from over 1,000 vehicles from different manufacturers, where it was shown that a FCW system has a positive effect on traffic safety [32, 33]. A report from the Insurance Institute of Highway Safety, which studied the insurance claims

of vehicles equipped with and without driver assistance systems, however, found that some systems, such as LDW, did not have a noticeable impact on insurance claims. A lot of these driver only systems, such as park distance control [145], have become quite common in today's production vehicles and can be added as an option to vehicles from almost all segments.

1.2.2 Level 1 - Driver Assistance

The next level of automation defines systems that actively assist the driver, hence driver assistance, by making an intervention in the longitudinal *or* lateral control of the vehicle. Such systems have also already been available in production vehicle for many years. The most common is ACC, which is an adaptation of cruise control, where the vehicle automatically keeps a safe distance to the vehicle in front and adjusts its speed appropriately [228, 302]. This allows the driver to leave the task of braking and accelerating, especially on the highway, to the vehicle. Another such system is Emergency Brake Assist (EBA), where the vehicle will automatically brake in order to avoid or reduce the impact of a collision [300]. Other assisted systems can make a slight intervention in the vehicle's lateral control, such as an Active Lane Assist (ALA), where the vehicle will make a slight adjustment in steering if the vehicle is about to cross the lane markings. Assisted parking systems are also already available in production vehicles, where a vehicle will take over control of the steering wheel during a parallel parking maneuver [145]. As with the driver only systems, most of these systems usually rely on just one sensor, or in some cases a combination of two sensors. Table 1.3 gives several examples of driver assistance systems which actively assist the driver in longitudinal or lateral control, along with the sensors used. It is important to note that despite taking an active control of the vehicle, all of these systems are limited in their capabilities and heavily rely on the fact that a driver is always in control and ready to override the system if a situation occurs which the system cannot handle.

The euroFOT project also showed a positive feedback for assisted systems, specifically ACC, where drivers were surveyed about the usefulness of the systems. It has also been shown that vehicle equipped with systems such as EBA tend to have lower insurance claim rates [144]. Despite their relative simplicity, assisted systems are the first step towards automated driving and are a critical step in introducing drivers to such technology.

1.2.3 Level 2 - Partial Automation

The current generation of driver assistance systems tend more towards partial automation, meaning that an assistance system can make an active intervention in the longitudinal *and* lateral control of the vehicle. However, as with systems classified under *assisted*, the driver must continuously monitor the system and the vehicle's environment and be ready to take over control at any given moment when a system limit is reach. Therefore, it is essential that the driver remain active in observing the current driving situation in order to take over control when the situation becomes critical for the assistance system. Such systems can also be designated as driver-in-the-loop systems, where the driver remains a integral part of the driving task. An overview of some example systems which fall under the category of partial automation, such as a Lane Keeping Assist (LKA) [113], is given in Table 1.4.

Table 1.3: Overview of current driver assistance systems which actively assist the driver with vehicle control along with the sensor technology used.

System	Brief Description	Sensor
Active Cruise Control	A cruise control system which automatically keeps a safe distance to the vehicle in front.	Radar
Emergency Brake Assist	A braking system which imitates automatic braking in a critical, near-collision situation.	Radar (and camera)
Active Lane Assist	Keeps the vehicle from crossing the lane boundaries by actively applying a force in the opposite direction when the vehicle is about to inadvertently cross a lane boundary.	Camera
Lateral Collision Avoidance	Makes an active steering intervention during a lane change situation in order to avoid a collision when another vehicle is overseen during the lane change.	Radar
Narrow Passage Assistant	Assists the driving with light steering input in narrow passage situations, such as in construction sites.	Ultrasonic
Park Assistant	Takes over steering control during a parallel parking maneuver.	Ultrasonic

Several automobile manufacturers have been offering partial automation driver assistance systems. The Traffic Jam Assist (TJA) system in new premium class vehicles is capable of longitudinal and lateral control in traffic jam situations up to a certain speed, usually around 60 $^{\mathrm{km}}/_{\mathrm{h}}$. Due to the requirement that the driver must continuously monitor the system, the driver's hands must usually remain on the steering wheel when the system is activated and a hands-on request is initiated if the driver keeps his/her hands off the steering wheel for too long. New advanced park assist systems have also been introduced, where the vehicle is capable of a complete automated parallel parking maneuver, for example as in [2]. In order to keep the "driver in the loop" during the parking maneuver, the driver must continuously press and hold a button on the middle console for the system to function, where the system will automatically come to a halt if the driver releases the button. For higher speeds, LKA systems have also been recently introduced, which when combined with an ACC functionality, is able to keep the vehicle in the lane as well as control the distance to the vehicle in front, practically realizing an automated driving system in basic scenarios, such as highways. However, such systems are still classified as level 2 automation since there may still be limits in their capabilities, requiring the driver to immediately take over control if necessary.

Due to the complexity of level 2 partial automation systems, the sensor configurations are usually more complex. In contrast to assisted driving systems, where one sensor system or technology is directly integrated with the functionality, partial automation has introduced the concept of an environment model in the vehicle's architecture [117, 119, 209]. The environment model combines the data from many sensors and provides the ADAS with an abstracted and generic description of the vehicle's environment. Such an architecture will be the norm in future ADAS and automated driving systems.

Table 1.4: Overview of driver assistance systems classified as level 2 partial automation, where the system takes over longitudinal and lateral control, but the driver must continuously monitor the system.

System	Brief Description	Sensor
Traffic Jam Assistant	Hands-on steering wheel automated driving in traffic jams up to a certain speed limit.	Several
Advanced Park Assist	Automated steering, braking and gas control during a complete parallel parking maneuver.	Ultrasonic
Lane Keeping Assist	Keeps the vehicle actively in the center of the lane while keeping a safe distance to the vehicle in front.	Several

1.2.4 Levels 3, 4 and 5 - The Higher Automation Levels

Level 3 and above in the SAE definitions differ from the lower levels in that driver's continuous monitoring of the system and the vehicle environment is *not* necessary. These automation levels allow the driver to carry out a secondary task, such as reading E-Mails, making a phone call, watching a movie, etc. The main difference between these higher automation levels lies in either the fallback scenario in case the system fails or in the types of scenarios or driving speeds in which the system can be activated. For level 3, for example, it is still assumed that a driver is available to take over control, given an ample amount of time, in case of a system failure. For level 4, the system must be capable of automatically performing a minimum risk maneuver, transitioning the vehicle into a safe state in case of a system failure and the driver does not respond properly (or is not physically in the vehicle). An example of such a minimum risk maneuver can be performed is described in [315]. Level 5 automation is the highest, which represents a completely autonomous vehicle, carrying passengers from one destination to another, being able to cope with all situations along the way without any driver intervention.

Since today's legal framework [273, 274], as well as the technological and verification requirements, for such higher levels of automation are not yet mature, these levels of automation are currently in the research and prototype phase. Many vehicle manufacturers, research institutes, as well as new companies, are currently building prototype vehicles to prove the feasibility of level 3 and above vehicle automation. This section gives a brief overview of various projects which have or are currently researching level 3 and above automated driving systems. Due to the not so obvious differences between level 3, 4 and 5 automation levels, it can be difficult to categorize various research projects into a specific automation level. In order avoid an unfair categorization, the projects mentioned in this section will not be assigned an automation level, instead it is important to recognize that all of these projects have the common goal of removing the driver's responsibility from continuously monitoring the system and the vehicle's environment, even though most testing of such research vehicles still require a safety driver due to their prototype nature.

Automated Driving on Race Tracks

A fun application of automated driving is to drive in an automated fashion around a racing circuit. One of the first projects to tackle the problem of driving automated around a race

Figure 1.2: The BMW TrackTrainer: autonomous driving of the ideal line on race tracks (Images Source: [46]).

track was the BMW TrackTrainer, shown in Figure 1.2. The idea was to demonstrate the performance of vehicle control algorithms at the limits of a vehicle's dynamics and to develop a system to teach the ideal racing line of a track for the purpose of driver training [284]. Using a highly accurate differential Global Positioning System (GPS), the ideal racing line is recorded by a professional driver, which can then later be driven back by the vehicle autonomously. This method was successfully demonstrated on any many circuits around the world. In addition to autonomously driving the ideal racing line, the vehicle also provided a visual feedback of the deviation from the ideal line when driven manually, allowing a driver to assess his/her own performance on the circuit.

However, the system was limited to the availability of the differential GPS signal. This posed a challenge when the system was to be demonstrated on one of the most challenging racing circuits in the world: the Nürburgring Nordschleife in Germany. This circuit has several locations where GPS availability was poor or even non-existent. This problem was solved by combining the GPS positioning algorithms with a camera-based lane marking detection system [285]. The use of this type of sensor data fusion and environment perception made it possible to complete the first fully autonomous lap around the Nürburgring Nordschleife on October 21st, 2009.

Researchers at Stanford University in conjunction with Audi and Volkswagen Group Electronics Research Lab have been involved in a similar project. They have developed an autonomous Audi TTS, pictured in Figure 1.3, with the goal of driving the vehicle on the Pike's Peak International Hill Climb [109, 250]. The goal of this project was to develop control algorithms that are capable of driving a vehicle at the limits [150]. The system also uses a differential GPS system to determine the exact location of the vehicle, and then autonomously guides the vehicle at the limits along a predefined path. The vehicle achieved its goal of ascending the 14,110-foot, 12.42 mile course at the end of 2010, doing so in 27 minutes and reaching speeds of up to 72 km/h [10].

These types of projects had the main goal of further developing vehicle control algorithms. Environment perception was not a priority and was solely used as additional information for improving vehicle localization. Perception of the rest of the environment was neglected and therefore environment sensors and perception algorithms were non-existent.

Automated Driving in Emergency Situations

The emergency stop assistant, an example of automated driving in an emergency situation, was developed in conjunction with the project SmartSenior, funded by the Ger-

Figure 1.3: Stanford University's autonomous Pike's Peak Audi TTS (Image Source: [252]).

man Ministry for Education and Research (German: *Bundesministerium für Bildung und Forschung*) [47, 48, 138, 246]. In [320], the overall system is described in detail. In the case of a medical condition, such as a heart attack, the emergency stop assistant, shown in Figure 1.4, is able to take over full control of all driving functions. In the worst case scenario, driving at freeway speeds on the left lane, the vehicle automatically slows down to a safe speed and performs successive lane change maneuvers, taking into account all surrounding traffic, until the shoulder of the road is reached and a complete stop is safely possible. The system is activated only when necessary and remains activated only for the duration of reaching the shoulder of the road. Even though the system operates only in this critical situation, several new technologies are required in order to successfully perform the automated maneuvers, the most important of which is the sensing of the vehicle's environment. Surround environment perception using several sensors, such as laser scanners, radar and cameras, were used in order to detect the vehicle's environment and localize the vehicle on the freeway in order to bring it to a stop on the shoulder of the road. The early algorithms developed during this thesis were used in the emergency stop assistant project [320].

Figure 1.4: The emergency stop assistant from the BMW Group Research and Technology (Images Source: [48]).

In critical scenarios, such as an emergency collision avoidance maneuver, an automated vehicle must react correctly at the vehicle's limit in a short amount of time. Further development of vehicle control algorithms from projects such as the BMW TrackTrainer noted above has led to the development of new control algorithms which can safely control the vehicle in sudden critical situations, even if such a situation leads to an oversteer scenario, forcing the vehicle to recover from a drift-like situation [51], as shown in Figure 1.5. Despite the fact that the demonstration presented in [51] neglected environment perception

technology, the integration of such vehicle control algorithms and their ability to react to a perceived environment will be critical in bringing a reliable highly automated driving system to market.

Figure 1.5: Highly automated driving at the vehicle limit in emergency situations, such as oversteer (Image Source: [51]).

Automated Driving on Freeways

In daily driving situations, level 3 or above automated driving applications can help increase comfort and safety by taking over the task of driving during specific situations, allowing the driver to do something else during the trip. A typical situation where safety and comfort can be increased is freeway driving. During routine or longer trips, a driver tends to be less attentive and alert while driving, possibly also being distracted from the relatively monotonous task. In such situations, automated driving has a great benefit and therefore much research is being done in this area.

The goal of automated driving on freeways is not new. Several projects in the 1990s were able to successfully demonstrate the potential of such technology. During the European Prometheus Project, the Universität der Bundeswehr (German Military University), under the lead of Prof. Dickmanns, demonstrated automated driving on freeways using camera systems with their vehicles VaMoRs and VaMoRs-P [85]. Based on the developments from Prof. Dickmanns and in cooperation with Daimler, two further vehicles were developed, VaMP (see Figure 1.6a) and VITA II, which at the end of the project demonstrated automated driving on freeways between Munich, Germany and Odense, Denmark [175]. Carnegie Mellon University also pioneered automated driving in the 1990s with their Navlab 5 vehicle (see Figure 1.6c), equipped with a camera and GPS [131], where several thousand of automated driving miles were accomplished during a trip across America from Pittsburgh, PA to San Diego, CA. A team in Italy from the University of Parma also pioneered automated driving on freeways with their vehicle ARGO [36, 57], which also made extensive use of camera systems. The ARGO vehicle (see Figure 1.6b) ended up making a 2,000 km journey through Italy using the road following functionality [57]. In the late 1990s, similar automated driving tests on freeways were developed in the United State during Demo '97 in San Diego, CA, organized by the National Automated Highway System Consortium. Several vehicles were developed for automated driving on freeways, where different approaches were demonstrated, such as magnetic-rail following or lane-following with environment perception sensors [215, 265]. These early projects made extensive use of cameras and custom actuator systems, and even with the limited computing power available at the time, basic automated driving functionalities could be successfully demonstrated.

(a) VaMP (Image Source: [298])

(b) ARGO (Image Source: [57])

(c) Navlab 5 (Image Source: [131])

Figure 1.6: Early prototype vehicles for automated driving on freeways.

Figure 1.7: Volkswagen's Temporary Auto Pilot, developed during the project HAVEit (Images Source: [123]).

Today's sensors, actuators and computing technology has helped push automated driving technology on freeways forward. Some vehicle manufacturers, for example, have developed a temporary auto pilot (Figure 1.7) system during the HAVEit project [123]. The temporary auto pilot is able to take over the tasks of driving on a freeway by adapting its speed and keeping itself in the lane automatically. A lane change maneuver, however, is not possible and must be made by the driver. The system is able to detect frontal traffic and the lane using a combination of radar, camera, laser scanner and ultrasonic sensors [28]. The sensors used are integrated into the vehicle body and retain the production look and feel of the vehicle, as opposed to the more complex sensor configurations used in some fully automated driving projects, described in Section 1.2.5. Work in HAVEit has also been done to approach the problem of automated driving on freeways from a legal safety perspective, taking into account the necessary traffic rules [279]. This led to the development of an intelligent automated driving system capable of following the rules and making decisions in traffic with other human drivers.

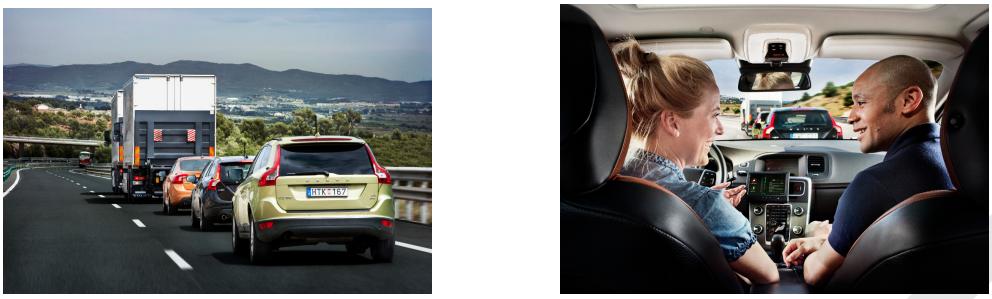


Figure 1.8: Project SARTRE: Highly automated driving on the freeway in a platoon (Images Source: [233]).

Similar to the temporary auto pilot, automated driving in platoons was demonstrated during the SARTE project [34, 229, 233]. In this application, a lead vehicle, usually a truck, is the head of a platoon. Other vehicles, in this case passenger cars, are then able to join the platoon and active an automated driving mode. The vehicle then maintains a close distance to the vehicle in front in the platoon and drives fully automatically. This allows the driver the freedom to do something else whilst in the platoon, such as conversing with a passenger (see Figure 1.8). A platoon can be composed of many vehicles and drivers can choose to enter and leave a platoon as they wish. Basic automotive sensors which are already found in production vehicles, such as ultrasonic sensors for parking assistance or radar sensors, are used to maintain the following distance in the platoon. Additional information is obtained by each vehicle in the platoon by communicating wirelessly with the lead vehicle using Vehicle-to-Vehicle Communication (V2V) technology, which distributes its driving path information. Platooning, as demonstrated in the SARTE project, uses a minimalistic approach to on-board environment perception sensors, but is a very restricted highly automated driving application, relying on the availability of lead platoon vehicles and V2V technology in order to activate the system.

Figure 1.9: Testing automated driving on freeways in real traffic from Continental AG and the Bosch Group (Images Source: [54, 69]).

Automated driving has also recently been tested on highways in real traffic by several vehicle manufacturers and automotive suppliers [9, 49, 54, 69, 82], examples of which are shown in Figure 1.9. These systems use near-production sensors, such as radar and stereo cameras, to detect other vehicles and obstacles, as well as the lane markings. The goal is to develop and test a system that is able to take over the task of driving during mundane situations, such as traffic jams or long stretches of highway. These systems, however, have their limitations and are not able to handle all situations, still requiring a take over control at times.

This thesis originates from such an automated driving on freeways project, described in [49], [312]. The goal of the project was to drive completely automated on freeways, with an artificial intelligence that is able to initiate and complete automatic lane change maneuvers in order to maintain a desired speed. The first fully automated drive between Munich and Ingolstadt in Germany was made on June 16th, 2011 with over 30 completely automated lane change maneuvers [7]. A combination of environment perception, localization and driving strategy technology was used to achieve this type of automated driving on freeways [319, 325]. The environment perception algorithms for object detection presented in this thesis were successfully implemented into the prototype vehicles for this project.

Figure 1.10: Automated driving on the freeway from the BMW Group Research and Technology (Images Source: [49, 50]).

Despite the success of the above-mentioned projects, many challenges still remain in such automated driving systems, the most important of which is how to deal with the driver. The main problem of automated driving is that the system must give the driver a certain amount of time to retake control of the vehicle if a system limit is reached. The amount of time required for a driver to retake control while out-of-the-loop is still a topic of research [73, 118]. This poses a new requirement on the vehicle sensing technology: self-diagnosis and evaluation is necessary in order to detect ahead of time if a system limit will be reached. Redundant systems with different sensing technologies will be required, where the software, hardware and electrical architecture must guarantee a fail-safe operation, at least for the time required for the driver to retake control of the vehicle or for the vehicle to perform the minimum risk maneuver. These problems still need to be solved before a true automated driving system on freeways can make it to a production vehicle.

1.2.5 Towards Fully Automated Driving

Fully automated driving is more commonly known as autonomous driving, self-driving cars or driverless cars, where the vehicle takes over complete control without any driver intervention throughout the trip or for a specific use-case, such as parking. Such systems are typically classified as level 4 or level 5 automation. Autonomous driving has been a dream shared by many for decades as the Utopian future of personal mobility. The first mention of an autonomous vehicles is from the Futurama exhibit at the 1939 World's Fair in New York by General Motors. The dream of such an envisioned automated highway system was for the first time experimentally realized in 1958 with the General Motors Firebird III concept [297]. The Firebird III used a pair of coils installed in the front of the vehicle which could detect an alternating current embedded in the road in order to guide the vehicle in its own lane. However, beyond this simple demonstration, the

(a) Image Source: [251]

(b) Image Source: [259]

Figure 1.11: Winners of the 2005 DARPA Grand Challenge, Stanley, from Stanford University (a) and the 2007 DARPA Urban Challenge, Boss, from Carnegie Mellon University(b).

dream of autonomous driving took a break for several decades. Recent advancements made in electronics and computer technology allow for more advanced sensor systems and computational power in much smaller form, making autonomous driving soon a possible reality.

Many projects have claimed to develop autonomous driving vehicles or driverless cars. However, only few projects have actually achieved truly autonomous driving, as the fail-safe requirements in all situations that the vehicle may encounter are still quite difficult to achieve with today's technology. In this section, projects which have tested fully automated driving are presented, as well as projects whose continued goal is to achieve high or full automation.

DARPA Grand and Urban Challenge

The Defense Advanced Research Projects Agency (DARPA) Grand and Urban Challenge mark the first real tests of truly driverless vehicles, where vehicles were tasked to complete a course without any persons on board. In 2002, DARPA announced the first Grand Challenge: a competition to design a driverless vehicle that could navigate a 240 km course in the Mojave Desert autonomously. The first DARPA Grand Challenge in 2004 consisted of 21 teams who qualified to participate, with teams coming mostly from universities [79]. The $1 million prize for the 2004 DARPA Grand Challenge remained unclaimed, as none of the teams came even close to finishing the course.

Shortly after the first competition, the DARPA Grand Challenge was announced with a $2 million prize for the team that can finish the 212 km course in the fastest time [80]. The proposed challenge turned the 2005 DARPA Grand Challenge into a large research effort from academia and industry to demonstrate and develop fully autonomous driving. This time, 5 teams managed to finish the course, with the team from Stanford University, in cooperation with Volkswagen of America, Inc. and Intel Research, taking first place with their vehicle, Stanley (see Figure 1.11a). The system and technology developed for Stanley is described in Stanford's DARPA Challenge technical report [267]. The major step forward is that a lot of the hardware, software and ideas used by Stanley are very similar to the type of technology relevant to current and future production vehicles and has set the foundation for continued robotic research with application to driver assistance systems and autonomous driving.

In 2007, the DARPA Urban Challenge took place [81]. This challenge required vehicles

to autonomously navigate various urban scenarios while obeying traffic laws and avoiding and passing other moving vehicles and stationary obstacles, still with no one on board the vehicle. The 96 km urban course would become one of the toughest challenges facing these types of autonomous vehicles. In the end, a total of six vehicles completed the course, with the team from Carnegie Mellon University finishing in first place with their vehicle, Boss (see Figure 1.11b).

Most of the vehicles in the DARPA Grand and Urban Challenge were highly-prototypical vehicles with many large and expensive sensors mounted on the roof or hanging off of the side of the vehicle in order to achieve the detailed perception required for autonomous driving. For dynamic object detection, the Stanford entry in the DARPA Urban Challenge relied primarily on a 360° multi-beam laser scanner mounted on the roof of the vehicle [184]. This single sensor provided a high-resolution 3D point cloud of the surround environment, on which object detection and tracking was accomplished. This sensor has become a popular sensor platform for autonomous driving projects. However, such a sensor is highly unrealistic in automotive applications at the consumer level, since the sensor is highly exposed and not integrated into the design of the vehicle. The winning entry from Carnegie Mellon University used a fusion approach to object detection and tracking, where raw measurements from sensors were pre-processed and then fused into a global tracking system to generate a single object list [276, 277]. Team LUX attempted the Urban Challenge using integrated laser scanner sensors, therefore allowing their vehicle to maintain its production design. However, the team failed to make it to the final competition [86]. Other teams used similar approaches, combining data from large laser scanner sensors and vision systems for perceiving the environment and localizing the vehicle very accurately [58, 143, 225].

The DARPA Grand and Urban Challenge competitions were a great step forward in demonstrating the possibility of autonomous driving in complex situations. However, the competitions showed that a lot more research needs to be done in order to make such technology a reality in production vehicles. Current sensor technology is too expensive and bulky and large amount of computing power is required to process all of the data and make the correct decisions while driving. But the DARPA Challenge competitions brought autonomous driving to the attention of the general public, and the universities and industry partners that participated in the competitions have continued their research since the challenges have taken place.

Recent Attempts at Autonomous Driving

The DARPA Grand and Urban Challenges proved to be a major milestone in autonomous driving research and technology. Since then, a lot of the teams that participated in the competitions are still doing research in the field of autonomous driving and have taken it upon themselves to achieve new goals.

After their participation in the DARPA Urban Challenge, the team from the Technische Universität Braunschweig decided to take autonomous driving technology into real urban traffic with its Stadtpilot project (Figure 1.12). The goal of this project is to drive autonomously on Braunschweig's inner ring road in real traffic, including the handling of intersections, merging into moving traffic and lane change maneuvers, all at speeds up to 60 km/h [260, 299]. The test vehicle "Leonie" uses a lot of the same technology from the DARPA Urban Challenge. However, due to the fact that the vehicle must operate in real traffic, the test vehicle has been extensively redesigned with a more optimized packaging

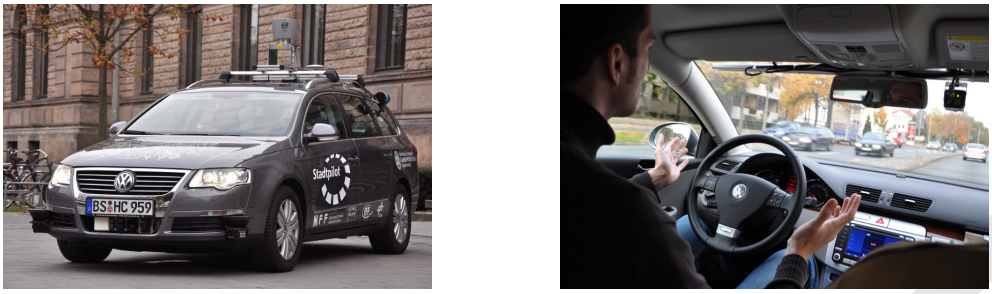


Figure 1.12: Driving autonomously on Braunschweig's inner ring road with the Stadtpilot project from the Technische Universität Braunschweig (Images Source: [260]).

(a) VIAC (Images Source: [282])

(b) BRAiVE (Image Source: [56])

Figure 1.13: Recent prototype vehicles for autonomous driving from VisLab at the University of Parma in Italy.

of sensor systems [299]. Additional safety precautions so that the driver can take over control of the vehicle at any time were also added. The first autonomous test drives on the 12 km inner ring road took place during the summer of 2010. However, in this initial phase, only a 2.5 km part of the complete course has been driven autonomously. Despite this, several hundred kilometers have already been successfully driven [234]. Even though the goal is to drive fully automated, driver intervention was at times still necessary due to uncontrolled situations or technical difficulties [234], proving that full automation is still quite a challenging task.

The VisLab, from the University of Parma in Italy, have recently undertaken a large expedition, attempting to drive autonomously from Italy to China in just over three months, during their Intercontinental Autonomous Challenge [35, 282]. The team drove autonomously using two vehicles equipped with various camera and laser scanner sensors (see Figure 1.13a). Their approach to autonomous driving was a "follow the leader" platooning-type method, where a lead vehicle is driven semi-autonomously and a following vehicle drives fully autonomously, using information from the lead vehicle. During the team's adventure, they covered a distance of over 15,000 km and collected over 20 terabytes of data, which will be used to improve state-of-the-art vision algorithms. However, their "follow the leader" approach and extensive use of external and roof-mounted sensor systems are far from a realistic implementation of fully automated driving technology for mass production vehicles. The team from the University of Parma also uses another prototype vehicle BRAiVE (see Figure 1.13b) for further researching autonomous driving technology [56]. As compared to other vehicles for fully automated driving, BRAiVE uses sensor

technologies, mainly cameras and laser scanners, which are integrated into the vehicle's design.

Figure 1.14: Freie Universität Berlin's AutoNOMOUS Labs autonomous driving car (Images Source: [13]).

The Freie Universität Berlin's AutoNOMOUS Labs has also continued to develop their autonomous driving technology beyond the DARPA Urban Challenge [13]. Their prototype vehicle, "MadeInGermany", pictured in Figure 1.14, has recently successfully driven autonomously on the roads of Berlin, including around Brandenburg Gate. In 2012, the vehicle even drove autonomously around Mexico City [105]. The goal of the project is to advance the technology for future driver assistance systems and autonomous driving. The prototype vehicle uses a combination of laser scanners to detect other traffic, including a large roof-mounted multi-layer laser scanner, shown in Figure 1.14, the same sensor used by many teams during the DARPA Urban Challenge.

Other university projects have also recently developed prototype vehicle platforms for autonomous driving. A consortium of industry and academic partners have developed a platform for autonomous vehicle testing in cities with close-to-market sensors using an electrical vehicle for the European Union Project V-Charge [110]. The project will focus on further developing all aspects of autonomous driving technology. Carnegie Mellon University, with their history and experience with autonomous driving, have also built new prototypes for autonomous driving [288], where the vehicle can already navigate intersections, work zones and deal with pedestrians and bicyclists.

Figure 1.15: Daimler "Bertha" Prototype Mercedes S-Class for autonomous driving (Image Source: [103]).

Vehicle manufacturers, such as Daimler, have also demonstrated prototype vehicles with the goal of fully autonomous driving in challenging environments. In 2013, Daimler presented a Mercedes S-Class capable of driving a 100 km route consisting of urban and country road environments, replicating the Bertha-Benz route from 1888 [103, 307]. The

vehicle makes strong use of Daimler's vision systems, which have been in development for urban environment for many years [102] and parts of which are available in the current production version of the S-Class.

The world's premier search engine company, Google, has also been researching autonomous driving technology since 2009 [43, 96, 122, 173]. Also using the popular 360° multi-beam laser scanner mounted on the roof the vehicle, Google has developed a fleet of cars that have driven several hundreds of thousands of miles in an autonomous driving mode. Main testing is on freeway roads, as described in [96], but more complex scenarios, including urban scenarios, have also been driven autonomously. The system relies heavily on the laser scanner sensor for object detection, mapping and localization, with some use of cameras for applications such as traffic light detection. Despite using a sensor technology which is far from being integrated in production vehicles, Google seems to have made a lot progress in the development of software and algorithms critical for autonomous driving. Since 2014, Google has further developed the system and have produced a fleet of small, low-speed pod-like vehicles (still using similar sensor technology), which are designed to not have a steering wheel and pedals, thereby focusing on a truly fully autonomous vehicle [43].

Figure 1.16: Google's self-driving car project (Image Source: [43, 122]).

A special situation where fully automated driving may be feasible much sooner is parking. Due to the generally low speeds and therefore lower risk, a fully automated vehicle in a valet-parking situation may be quite realistic in the near future. Such valet-parking functions usually require some form of map of the parking infrastructure in order to localize the vehicle within the parking infrastructure (usually without GPS due to the indoor nature of parking facilities), sensors to aid in localization and object detection and a motion planning algorithm for generating a drivable path into the parking space [130, 151, 161, 181]. Several university projects demonstrations have shown that such fully automated parking applications are technically possible [130, 151, 161, 181]. Automobile manufacturers have also demonstrated such parking applications. A recent demonstration of fully autonomous valet parking with an electric vehicle equipped with four laser scanner sensors [52] is shown in Figure 1.17.

Even with the advances made within the aforementioned projects, all prototype vehicles still heavily rely on the fact that a driver is necessary and ready to take over control at a moment's notice, just in case something does go wrong, demonstrating the fact that much research still needs to be done before true fully automated driving will be possible.

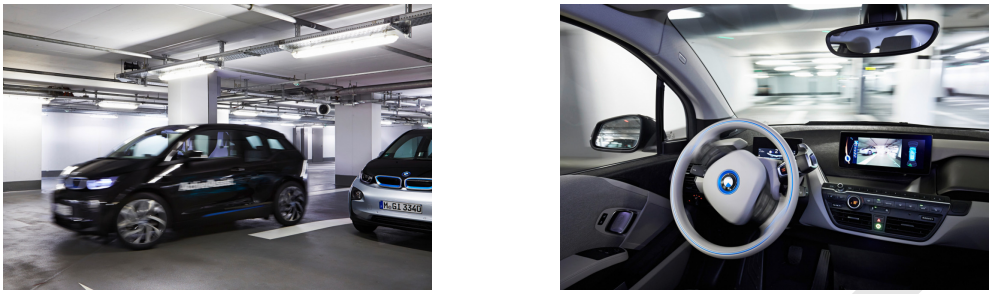


Figure 1.17: Valet parking project with a BMW i3 demonstrating fully autonomous driving in parking situations (Images Source: [52]).

1.3 Problem of Object Detection

As shown in Figure 1.1b, future driver assistance systems will require a unified framework and description of the environment for the processing of sensor data. This unified framework is commonly known as an *environment model*, denoted in this thesis as $\mathcal{E}$. Mathematically, the problem of estimating the environment model can be formulated as a conditional probability density function

$$p(\mathcal{E}(t) \mid \mathcal{I}) \tag{1.1}$$

where $\mathcal{E}(t)$ is the current estimation of the environment model conditioned on some information $\mathcal{I}$. Starting with this basic formulation, the problem of object detection can be derived.

The information $\mathcal{I}$ can generally be broken down into some sensed data $\mathcal{D}$ and some a-priori information $\mathcal{I}^*$ stored in memory, where the sensed data can further be broken down into data detected from sensors perceiving the environment, $\mathcal{S}$, and sensors detecting something about the host system, $\mathcal{U}$, in this case the vehicle with an ADAS or automated driving system:

$$p(\mathcal{E}(t) \mid \mathcal{D}, \mathcal{I}^*) = p(\mathcal{E}(t) \mid \mathcal{S}, \mathcal{U}, \mathcal{I}^*) \tag{1.2}$$

A vehicle can be equipped with any number of sensors perceiving the environment, N_S, where a set of measurements from the moment the system is turned on to the current time t, denoted as $\{Z^{1:N_S}\}_0^t$, contribute to the estimation of the environment model, leading to the formulation $\mathcal{S} = \{Z^{1:N_\mathrm{S}}\}_0^t$. Similarly, with the sensors detecting something about the host system, a set of control vectors is defined such that $\mathcal{U} = \{\mathbf{u}\}_0^t$. The a-prior information can be defined as a digital map at time t, $\mathcal{I}^* = \{\mathcal{M}\}_0^t$, where the map's data source comes from previously stored information, for example a navigation map on a hard disk drive or from a database. The estimation of the environment model can now be described as

$$p(\mathcal{E}(t) \mid \{Z^{1:N_\mathrm{S}}\}_0^t, \{\mathbf{u}\}_0^t, \{\mathcal{M}\}_0^t). \tag{1.3}$$

The problem can further be formulated as a recursive estimation problem, using the prior estimation of the environment model in the current estimation along with the Markov assumption. This leads to the problem of estimating the environment model conditioned on the previous estimation and current measurement data:

$$p(\mathcal{E}(t) \mid \mathcal{E}(t-1), \{Z^{1:N_\mathrm{S}}\}(t), \mathbf{u}(t), \mathcal{M}(t)) \tag{1.4}$$

where the prior estimation of the environment model $\mathcal{E}(t-1)$ is used to estimate the current environment model $p(\mathcal{E}(t)$. Map information is also only retrieved for the current time t, such that $\mathcal{M}(t)$ represents the subsection of the map currently relevant for the host vehicle. This formulation assumes that all of the data from the sensors N_S gets processed synchronously at time t. Another possibility is to process sensor data asynchronously, where the current environment model is updated with the data from just one sensor at a time:

$$p(\mathcal{E}(t) \mid \mathcal{E}(t-1), \{Z^i\}(t), \mathbf{u}(t), \mathcal{M}(t)) \tag{1.5}$$

The next important step is to think about the type of representations which are of interest in an environment model. In ADAS, typically three types of representations are important: host-vehicle localization and pose, dynamic and static objects/obstacles and the road infrastructure. The representations for objects and obstacles are usually categorized into model-based approaches and spatial-based approaches. In total, four environment representations are to be estimated in an environment model, replacing $\mathcal{E}$ with $\mathcal{X}$ (host-vehicle localization and pose), $\mathcal{O}$ (model-based object detection), $\mathcal{G}$ (spatial-based object detection, usually realized with a grid-based representation) and $\mathcal{R}$ (road infrastructure), leading to the detailed environment model estimation problem of

$$p(\mathcal{X}(t), \mathcal{O}(t), \mathcal{G}(t), \mathcal{R}(t) \mid \mathcal{X}(t-1), \mathcal{O}(t-1), \mathcal{G}(t-1), \mathcal{R}(t-1), \{Z^i\}(t), \mathbf{u}(t), \mathcal{M}(t)) \tag{1.6}$$

In this thesis, it is assumed that the various representations are independent of one another, allowing a separate and parallel estimation for each representation, where (1.6) can now be written as

$$\begin{aligned} &p(\mathcal{X}(t) \mid \mathcal{X}(t-1), \{Z^i\}(t), \mathbf{u}(t), \mathcal{M}(t)) \cdot p(\mathcal{O}(t) \mid \mathcal{O}(t-1), \{Z^i\}(t), \mathbf{u}(t), \mathcal{M}(t)) \cdot \\ &\quad \cdot p(\mathcal{G}(t) \mid \mathcal{G}(t-1), \{Z^i\}(t), \mathbf{u}(t), \mathcal{M}(t)) \cdot p(\mathcal{R}(t) \mid \mathcal{R}(t-1), \{Z^i\}(t), \mathbf{u}(t), \mathcal{M}(t)) \end{aligned} \tag{1.7}$$

However, for more complex environment models, some dependencies between the representations may exist and would need to be properly accounted for. In this thesis, however, such dependencies are ignored and will be left to future work.

The problem to be solved in this thesis is that of the model-based object detection, where it is additionally assumed that a-priori map information does not contribute to the estimation, leading to the problem to be solved in this thesis as follows

$$p(\mathcal{O}(t) \mid \mathcal{O}(t-1), \{Z^i\}(t), \mathbf{u}(t)). \tag{1.8}$$

The following chapters will go into much further detail on how to estimate a consistent representation of objects in the environment from several sensors sources and present a novel approach for doing so.

1.4 Contribution and Outline of the Thesis

The focus of this thesis will be the development of an architecture and concept, including all necessary algorithms, for combining the data from multiple vehicle-integrated environment perception sensors with the goal of achieving reliable object detection for the highly automated driving on freeways project by the BMW Group Research and Technology, as

described in Section 1.2.4. The project sets itself apart from most other projects by achieving a full vehicle surround perception without any large and expensive sensors mounted externally onto the vehicle, allowing the original body design of the vehicle to be maintained. However, the architecture should also meet the requirements for a general and modular framework for sensor data fusion for future driver assistance systems.

The problem with modern driver assistance systems in production vehicles is that sensor technologies are unique to a specific application, as was shown in Figure 1.1a. However, with new and more complex driver assistance systems on the horizon, such as highly automated or autonomous driving and even partially automated, more sensors will be required. A single, general framework for environment perception will be necessary in order to share sensor data between different applications. The output of this general framework is an environment model, where a part of this model is the description of detected objects. This will reduce complexity and and costs in integrating several driver assistance systems in the future.

This thesis focuses on the problem of sensor data fusion with respect to object detection for future advanced driver assistance systems and in particular highly automated driving on freeways. Previous approaches rely on unrealistic sensor configurations for production vehicles and lack the modularity required for cost-saving and multi-platform applications. These problems are addressed in this thesis with the proposition of a modular high-level sensor data fusion architecture for surround environment perception that conforms to the future requirements of the automotive industry. This novel sensor data fusion architecture is demonstrated for highly automated driving applications that are currently being developed at the BMW Group Research and Technology (Figure 1.18). Experience and results from this project can be found in [312].

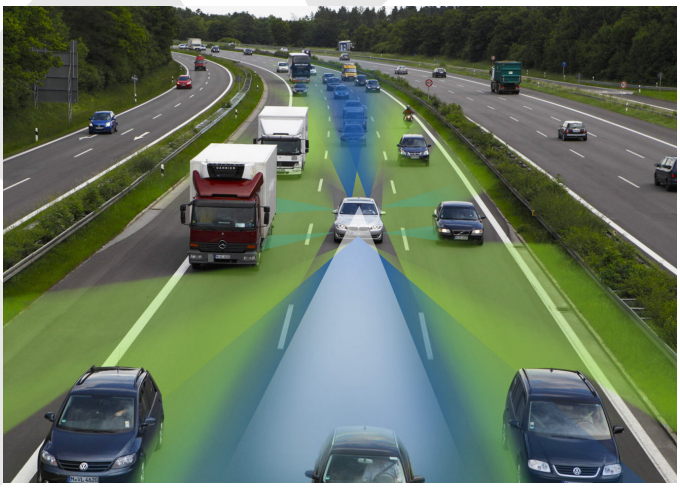


Figure 1.18: Surround environment perception for highly automated driving on freeways using laser scanner (green), radar (blue), camera (white) and ultrasonic (dark green) sensors.

In order to develop the best sensor data fusion architecture for surround environment perception, algorithms and architecture configurations which already exists in the literature should be considered and analyzed. The requirements to be met for a sensor data fusion architecture for future driver assistance systems will be considered based on the following criteria:

- The architecture must be modular and easily reconfigurable for various sensor config-

urations and applications, which will allow the system to be used for different vehicle platforms.

- Different sensor technologies must be supported in order to guarantee reliability and robustness through the fusion of different measurement principles.
- The design of the architecture must be held general, in order to support sensors from different suppliers and manufacturers.
- The algorithms for fusing sensor data must allow for a complete surround environment perception, including the reliable detection of large vehicles such as trucks and buses.
- The quality and uncertainty of the detected objects must be quantified in order to allow for a probabilistic treatment of sensor data for situation interpretation and application parametrization.

The developed architecture and its algorithms will be tested and evaluated in practice using a prototype vehicle for advanced driver assistance systems and highly automated driving. This vehicle is equipped in a manner that is realistic for realizing surround environment perception for production vehicles in the future. The practical implementation of the architecture must meet the following requirements:

- The sensors used for surround environment perception should be integrated into the vehicle's body in order to preserve the production design of the vehicle.
- Whenever possible, sensors already in production vehicles should be used so that modern sensor technology already found in vehicles can be taken advantage of, thereby making the sensor configuration as realistic as possible of future driver assistance systems.
- The sensor data fusion architecture must run in real-time.
- The architecture should be able to detect, with good reliability, all relevant traffic participants while reducing false detection for an application, such as highly automated driving on freeways.
- The detected objects from the environment must be accurately enough detected in order to make a reliable situation interpretation of the environment.

Given these architectural and practical requirements, this thesis makes the following novel contributions to the problem of surround environment object detection for ADAS applications:

- A modular high-level sensor data fusion architecture for object-level detection and fusion, where low-level or feature-level fusion may also contribute.
- A consistent object representation which contains not only the object's state, but also attributes such as its existence and classification.

- Object association techniques which consider complex object constellations during association, in particular for area around the vehicle where sensor field-of-views overlap, thereby enabling 360° object association.
- Integration of attribute information for existence and classification into popular data association techniques.
- Combination of typical state-space data association techniques with a novel geometrical association method which considers an object's dimensions during object association.
- A sensor-to-global fusion strategy such that new sensor data is fused asynchronously into a global estimate of detected objects.
- Object state fusion algorithms (track-to-track fusion) within the sensor-to-global fusion strategy concept.
- Demonstration that the information matrix fusion algorithm provides optimal results in terms of performance compared to a low-level fusion approach using a central Kalman filter.
- A novel method for the fusion of an object's dimensions.
- A simple Bayes formulation for object existence estimation at the sensor-level.
- Method for object existence fusion using the Dempster-Shafer evidence theory which considers sensor existence estimation performance.
- An example Gaussian approach to sensor-level object classification.
- Method for object classification fusion using the Dempster-Shafer evidence theory which considers sensor classification estimation performance.
- Method for parametrization of the Dempster-Shafer evidence theory based existence and classification fusion algorithms which take into consideration different sensors' performance in attribute estimation.
- Thorough evaluation methods and results for object detection algorithms for state, existence and classification for surround environment perception with a multitude sensors.

The thesis begins in Chapter 2 with an overview of different types of sensor data fusion architecture configurations that have been used in the literature for automotive applications. Each architecture will be considered for its viability for future driver assistance systems meeting the requirements from above. Based on this analysis, a sensor data fusion architecture for surround environment perception is then proposed.

The chapters that follow describe how to process and fuse sensor data in order to obtain a full description of a detected object in the vehicle's environment. Chapter 3 considers how to decide which detected objects from different sensors represent the same object in reality. This process is called *object association*. Additionally, Chapter 3 will also consider the different strategies for actually synchronizing and fusing the sensor data together in

order to form a single object. The problem of how to estimate and fuse the position, velocity, acceleration, length and width (the *state* of an object) and its uncertainties, or the *covariance*, is described in Chapter 4. In Chapter 5, the calculation of an object's *existence* is described, which is a measure of the detection quality of an object. The likelihood of what type of object is detected, such as a car, truck, motorcycle, etc. is called *classification* and is described in Chapter 6.

The complete sensor data fusion architecture is evaluated in Chapter 7 using a prototype vehicle designed for testing future driver assistance systems and highly automated driving. A summary of the results and experience in developing the sensor data fusion architecture is given in Chapter 8 with an outlook to future work that could improve surround environment perception for more applications. Each chapter is followed with a short discussion about topics covered in that chapter.

2 Sensor Data Fusion Architectures

Single sensor systems are restricted to the performance and field of view of that single sensor. Applications with a high degree of automation, such as highly automated driver assistance systems, require a high degree of robustness and reliability. Through the use of sensor data fusion, the data from multiple sensors can be used to increase the system's overall performance and guarantee a certain degree of robustness. This is achieved by using sensors which are based on different measuring principles, so that errors or lapses in performance of a certain sensor type is compensated by a complementary sensor with better performance. Sensor data fusion is also able to increase the field of view, where a single sensor is not able to fully observe the required area, as is the case in automotive surround environment perception with vehicle-integrated sensors. Sensor data fusion is a critical part in estimating the environment model, as described in Section 1.3.

Multi-sensor systems for detecting and tracking objects require an architecture for combining the data from multiple sensor sources, with sensors possibly having different properties and measurement principles. These types of architectures are called *sensor data fusion architectures*. Such systems were first required and designed for the aerospace industry, where multiple radar sensors and stations were used to track aircraft for defense purposes. With the advent of stealth aviation technology, the algorithms and methods of sensor data fusion have become all the more important for reliably detecting aircraft.

Many configurations of sensor data fusion architectures have been proposed and successfully used in the aerospace industry [16, 25, 40]. However, in aerospace applications, detected aircraft can be modeled as point targets since the distances between the radar stations and targets are relatively large. In automotive applications, the point target assumption is not valid due to the fact that a single vehicle can take up a significant portion of a sensor's field of view. With the advent of new Advanced Driver Assistance Systems (ADAS) and the need for more sensor technologies to support these systems, the sensor data fusion architectures and algorithms have to be redesigned and optimized for use in automotive applications.

This chapter aims to give a brief overview of the different types of sensor data fusion architectures that have been proposed and demonstrated for automotive applications. The different types of fusion architectures will then be more closely examined and compared to one another in order to get a better idea of each architecture's advantages and disadvantages. Finally, a novel sensor data fusion architecture for model-based object detection is proposed, designed to be modular and optimized for future ADAS, particularly automated driving applications.

2.1 Overview

There are many possible ways to organize and process the data coming from a multi-sensor system. These different types of sensor data fusion architectures each have their

advantages and disadvantages. In this section, some basic sensor data fusion architectures will be presented in order to gain an understanding of the different ways that a multi-sensor system can be configured. Fusion architectures can in theory be separated into two general categories: low-level, or centralized, and high-level, or decentralized, fusion architectures [16]. The following sections will describe and compare several types of centralized and decentralized fusion architecture that have been successfully used in ADAS.

2.1.1 Low-Level

A basic and intuitive sensor data fusion architecture is based on a low-level fusion method. In a low-level fusion architecture, the basic idea is that each sensor transmits raw sensor data to a central fusion module. At the fusion module, the raw sensor data is then processed using a single central tracking algorithm. Because of this central filtering algorithm, a low-level fusion architecture is also called a centralized fusion architecture. There are several variations of sensor data processing in a low-level fusion architectures.

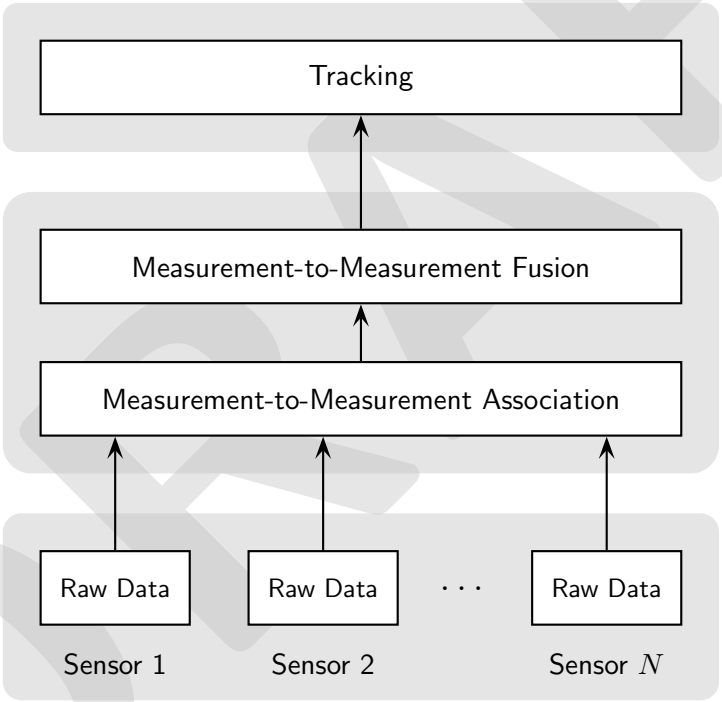


Figure 2.1: Low-level fusion architecture using measurement-to-measurement fusion.

Measurement-to-Measurement Fusion

One method of low-level sensor data fusion is to combine the raw sensor data using a measurement-to-measurement association and fusion algorithm and then feeding the fused measurements to a central tracking algorithm. The drawback is that sensors must be highly synchronized in order to achieve successful association results. The low-level measurement-to-measurement fusion architecture is illustrated in Figure 2.1. This type of low-level measurement-to-measurement fusion architecture has been successfully used for an automotive pre-crash application using forward-facing radar and laser scanner sensors [219].

Naab and Frey also describe this type of low-level measurement-to-measurement fusion and centralized tracking system in order to realize a modular architecture for driver assistance and active safety systems [201]. In [257], a low-level fusion method between a laser scanner and camera sensor is described, where the measurements of the two sensors are combined. A variation of this measurement-to-measurement low-level fusion approach is used in [106, 255], called scan data fusion, where the raw data measurements from several laser scanner sensors are first aligned to a common coordinate system and time before a central object recognition module performs the segmentation and tracking of objects. More recently, [14] describes a measurement-to-measurement fusion approach between laser scanner and stereo camera sensors to detect moving objects at intersections. As shown in the applications above, such a fusion method has proving to be quite effective in simple, synchronized two-sensor systems.

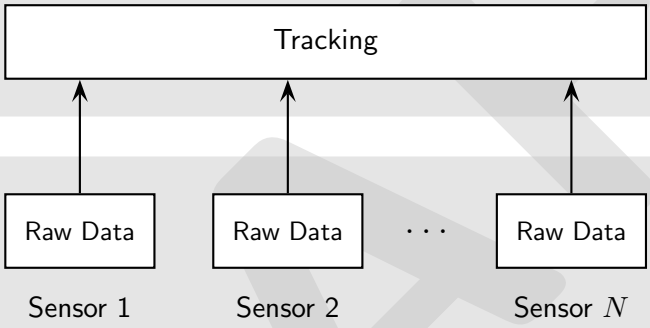


Figure 2.2: Low-level fusion architecture using direct measurement updates.

Centralized Filter with Direct Measurement Updates

Another typical low-level fusion method is to directly use the sensor measurements as the input measurements to a tracking algorithm, without prior fusion (Figure 2.2). The measurements from different sensors are chronologically integrated into the central tracking algorithm. For this type of low-level fusion, sensors have to have an accurate global time base so that the central tracking algorithm can properly be updated from one measurement to the next, especially if the sensors have different cycle times, making them asynchronous [127, 294]. If the sensors are asynchronous and have varying latencies, it is possible that the measurements arrive at the fusion module out-of-sequence. This problem has been solved in the literature so that out-of-sequence measurements can be directly integrated into a central tracking algorithm using out-of-sequence measurement algorithms [19, 22, 227]. This direct type of low-level fusion method has been used in automotive pre-crash and safety applications using asynchronous radar sensors [84] and out-of-sequence measurement algorithms [321–323],[189]. A unique low-level fusion approach based on a belief network is described in [253] using a radar and a camera sensor in order to detect the most plausible and correct position of vehicles.

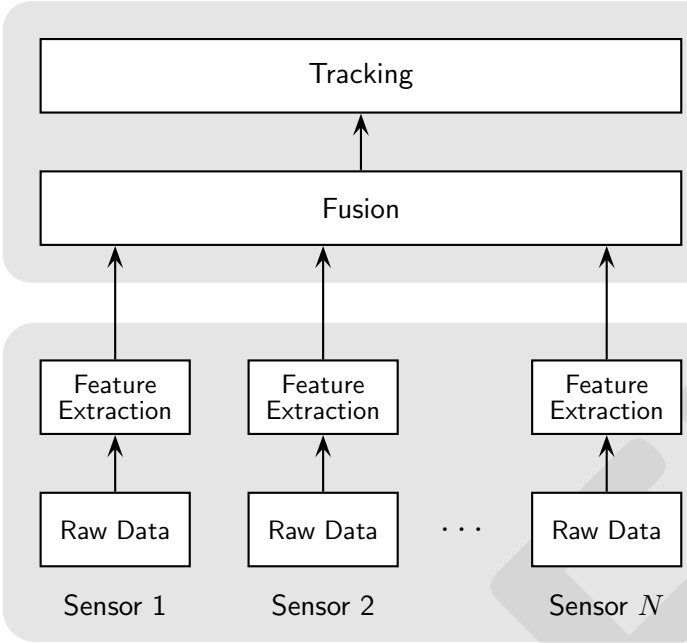


Figure 2.3: Feature-level fusion architecture.

Feature-Level Fusion

Kaempchen developed the idea of a feature-level fusion architecture, which is a derivative of a low-level fusion architecture [139, 140]. Figure 2.3 illustrates a feature-level fusion architecture. In a feature-level fusion architecture, a sensor does not transmit raw sensor data to the fusion module, but extracted features. This requires that the sensors operate using a global object model that has defined features which a sensor must be able to detect and transmit. In some configurations, the sensors also perform the data association step in the tracking process, requiring a feedback of the fusion module's already tracked objects. The fusion module is then responsible for the filtering, or tracking, of sensor feature data.

A similar architecture is described by Darms [75, 76], which also has some characteristics of a feature-level fusion architecture. In this architecture, the sensors may or may not do some pre-processing of raw sensor data to generate features. Data association can also occur at in the sensor instead of at the fusion module, which requires information feedback from the fusion module. Additionally, the described architecture allows for application-specific filtering properties and models of the sensor data. However, the interfaces between the sensors and the fusion modules seems to be quite loosely defined, making it difficult to realize a general architecture when more sensors are necessary. This feature-level fusion architecture was used in a modified form by Carnegie Mellon University, the winning team in the DARPA Urban Challenge [78].

The feature-level fusion approach was further developed by the University of Ulm in the works lead by Mählish [167, 168, 171]. It is argued that the feature-level fusion approach provides superior classification and detection performance. This feature-level fusion architecture was further extended using the Joint Integrated Probabilistic Data Association (JIPDA) algorithms, which combine state and existence information of detected objects [170, 172, 192, 194]. A classification approach combining JIPDA and the Dempster-Shafer theory was also developed for this feature-level fusion architecture [191–193].

Surround environment perception using contours as a feature was first developed for the Defense Advanced Research Projects Agency (DARPA) Urban Challenge by Team CarOLO from the Technical University of Braunschweig [225]. A central extended Kalman filter capable of tracking arbitrary contour shapes was used to track objects from laser scanner and radar sensors mounted in the front and in the rear of the vehicle [89, 225]. This low-level contour-based tracking and sensor data fusion architecture was extended using evidence theory for the Stadtpilot project in [211–213].

A feature-based object model is used in [70, 71] to track objects in a road environment. A three-dimensional feature space is defined for each object type, where sensor-specific features are defined inside this feature space. The idea is that each sensor has the ability to detect specific and different features of an object. The detected features are then transformed into the feature space and fused using a multiple model filtering algorithm. The features, however, are very sensor specific, making this approach difficult as the number and different types of sensors is increased for surround environment perception. In [30], features are extracted from radar, laser scanner and vision sensors and then fused using an approach based on an information filter.

Summary

A low-level fusion architecture has proven to be a successful means of sensor data processing in automotive applications. However, most of the applications described above, with a few exceptions, were designed for simple two-sensor perception systems mostly for Active Cruise Control (ACC) or collision avoidance applications. As more sensors are required, the complexity of a low-level fusion architecture can quickly increase and make a practical implementation quite difficult. Feature-level attempts to capitalize on some of the weakness of a traditional low-level fusion architecture by extracting features from the raw sensor data before performing association and centralized tracking of objects.

2.1.2 High-Level

A high-level fusion architecture can be considered the opposite of a low-level or feature-level architecture. Otherwise known as a decentralized or distributed fusion architecture, a high-level fusion architecture depends on the fact that each sensor has its own filtering and tracking algorithm and the fusion module simply combines the already-filtered results from multiple sensors. Therefore, high-level fusion consists of fusion at the object-level, or track-level, where the objects have already been tracked by each sensor independently. A basic high-level fusion architecture is shown in Figure 2.4.

For surround environment perception, Becker describes and compares several ways of fusing sensor data from independent sensor tracks for autonomous driving on test tracks [29]. In [201], the idea of object-level fusion describes a higher-level fusion module, which combines data from lower-level fusion modules or sensors directly, creating so-called enhanced object lists. However, the architecture is only loosely described and does not seem to depend on standard interfaces between modules, which could increase complexity and reduced the overall architecture's modularity.

High-level sensor data fusion architectures have also been used for simple two-sensor fusion systems. In [182], a high-level architecture is used between a radar and infrared sensor for improved vehicle detection. Labayrade describes a collision mitigation system

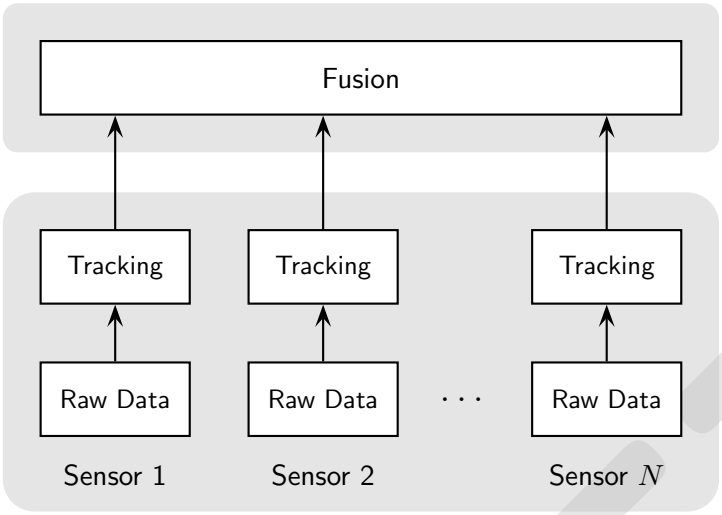


Figure 2.4: High-level fusion architecture with decentralized tracking algorithms.

using a laser scanner and stereo-vision sensors [153]. The fusion, however, merely consists of using the stereo camera to verify the relevance of the objects detected by the laser scanner.

In a high-level sensor data fusion architecture, track-to-track fusion algorithms [63, 268, 270, 271] are usually used to fuse objects from different sensors. Floudas describes a high-level sensor data fusion architecture between long range radar, short range radar and camera sensors using track-to-track fusion algorithms that take into account the correlation of object data between sensors [100]. A similar approach is used in [98, 99] for a fully-automatic braking application for trucks. Track-to-track fusion is also applied to two-sensor perception systems for infrared/radar and radar/lidar in [42]. A maximum likelihood approach for track-to-track fusion is used in [66] for lidar and radar fusion for frontal object detection. In order to quantify the performance of different fusion algorithms, several track-to-track fusion algorithms were compared in simulation for automotive applications in [174]. In [263], track-to-track fusion is applied to vehicular networks, where a vehicle equipped with a long range radar sensor fuses its data with communicated object data over vehicular networks in a high-level fusion architecture. Track-to-track fusion is also used in the proposed fusion architecture of [232], where covariance intersection is used as the main fusion method.

In summary, high-level sensor data fusion architectures have also been successfully used in automotive applications. However, unlike with the low-level or feature-level fusion architectures, not much work has been done for surround environment perception of sensor data at the object-level. With complex sensor configurations, the object association and fusion process can become more challenging in a high-level sensor data fusion architecture.

2.1.3 Hybrid

As the name suggests, a hybrid fusion architecture is a combination of different types of fusion architectures. In some cases, a specific type of fusion architecture may have advantages over another in specific sensor configurations. Therefore, in order to take advantage of this, hybrid fusion architectures have been developed.

Naab describes on overall sensor processing architecture for driver assistance and safety systems, which could consist of both low-level fusion modules and high-level, or object-level, fusion modules in one generic overall architecture [200]. This architecture is further developed in [201]. However, the interfaces between the modules and its detailed implementation are vaguely described. In [235], Scheunert describes a multi-level fusion architecture used in the project European PEeVENT. The architectures defines a perception memory object (PMO), which can be accessed at all levels of data processing and fusion with a PMO can occur at any level, depending on the type of object and model dealt with.

Hybrid fusion architectures seem to have the advantages of both low-level and high-level fusion architectures. However, depending on how the architecture is implemented in practice, a hybrid fusion architecture may end up being quite complex to realize if fusion must occur between many types of sensors distributed over different types of communication channels and hardware.

2.1.4 Comparison

The previous section presented several types of sensor data fusion architectures, all which have distinct advantages and disadvantages. Depending on the application and sensor configuration used, one type of architecture may be favorable over another. In this section, sensor data fusion architectures will be compared to one another in terms of their performance in an automotive surround environment perception application, which is necessary for ADAS and automated driving systems, examples of which were presented in Chapter 1.

Previous work has also compared sensor data fusion architectures for automotive applications. Becker described three approaches to the fusion of sensor data for autonomous vehicles in [29], two of which can be considered low-level fusion and the other high-level fusion. Simulation results concluded that a low-level architecture using measurement fusion was the most suitable. In the dissertation from Kaempchen, it is argued that a feature-level fusion architecture has better tracking and classification performance than a low-level and a high-level architecture [139]. In [75], Darms compares centralized (low-level) and decentralized (high-level) fusion architectures based on their loss of information, accuracy, bandwidth, processor load, and modularity. Based on the comparison, a feature-level architecture was developed as a compromise. Herpel et al. focuses on comparing a low-level with a high-level fusion architecture in simulation for urban and motorway traffic and concludes that the low-level fusion architecture consistently performs better in terms of accuracy and detection reliability [124]. More recently, in [308], low-level, feature-level and high-level fusion architectures were compared and a high-level sensor data fusion architecture was proposed for highly automated driving applications.

In this section, low-level, feature-level, high-level and hybrid fusion architectures will be compared to one another based on their sensor encapsulation, bandwidth, accuracy, complexity, modularity, sensitivity to errors and bias and practicality. Table 2.1 gives a qualitative comparison between the different types of architectures, each aspect of which will be discussed in further detail.

Sensor Encapsulation

An important feature in a sensor data fusion architecture for automotive applications is sensor encapsulation, where senor data can be abstracted and encapsulated, such that

Table 2.1: Comparison of sensor data fusion architectures (+: positive, −: negative, +/−: neutral).

	Low-Level	Feature-Level	High-Level	Hybrid
Sensor encapsulation	−	+/−	+	+/−
Bandwidth	−	+/−	+/−	+/−
Accuracy	+	+/−	+/−	+
Complexity	−	+/−	+/−	+/−
Modularity	−	+/−	+	+/−
Sensitivity to errors/bias	−	−	+/−	+/−
Practicality	−	+/−	+	+/−

the sensor specific knowledge is not necessary for further processing. In [200], Naab talks about the idea of a virtual sensor, which can consist of any arbitrary sensor data processing, consisting of one or more sensors, but is abstracted to an interface such that it appears to be just one sensor.

The low-level architecture tends to be the most difficult to encapsulate when different types of sensors are used due to the fact that raw sensor data must be uniquely handled in a centralized fusion algorithm depending on the type of sensor. Feature-level fusion architecture improves on this by pre-processing raw sensor data and extracting features, based on a global object model, which are then transmitted to the centralized fusion algorithm. The extracted features, if well designed, can be encapsulated from any sensor specific knowledge. The high-level architecture, however, is the best in terms of sensor encapsulation, since each sensor processes as much of its data as possible, including filtering over time, before transmitting it to the fusion module as a complete object.

Bandwidth

Another big concern in sensor data fusion, especially where many sensors are involved, is the bandwidth required to transmit the data to the fusion modules and to the applications which require the data. Most work in the literature agrees that a low-level architecture has the highest demand on bandwidth [75, 124, 140, 167]. The feature-level architecture attempts to improve the demand on bandwidth by transmitting features, which have already been processed by the sensor such that not every single piece of data must be transmitted [140, 167]. The high-level architecture also requires less bandwidth than the low-level architecture, due to the fact that far less objects are transmitted than sensor measurements or features. However, it may not be much of an improvement over the feature-level architecture due to the fact that the amount of data representing already-filtered objects will be larger than feature data, due to the extra information gained through filtering the data over time.

Accuracy

Depending on the application, accuracy is very important in automotive environment perception. The least amount of information is lost in a low-level fusion architecture, where sensor data is processed in its purest form, thereby resulting in the most accurate fusion. Many fusion algorithms are optimal if raw sensor data is processed and synchronized

at the fusion module [16]. When processing raw sensor data, no model assumptions are made and data between sensors and over time are independent [75], which improves accuracy. Accuracy in a feature-level architecture is quite similar, but model assumptions may decrease the accuracy. The most challenging architecture in terms of accuracy is the high-level fusion architecture, since data between sensors may be correlated, processed in an unknown manner and model assumptions are made, which may not be consistent from one sensor to another. However, it has been shown that some algorithms in a high-level fusion architecture are able to produce similar results as those of a low-level architecture [311, 313].

Complexity

It is important to consider the complexity of a sensor data fusion architecture. Different applications may allow for different amounts of complexity in a system. Typically, for specific and unique applications, such as the ones usually found in the aerospace industry, a complex system may be necessary to obtain a precision otherwise not possible. But in automotive applications, where cost is a very important factor, simpler systems are usually favored in order to make it possible to bring new technology to the market. In order to keep costs down and development efficient, a sensor data fusion architecture should retain a low degree of complexity so that it is scalable and useable for many applications with different types of sensors, which all may be separately developed.

For surround environment perception, a low-level sensor data fusion architecture has the largest degree of complexity, as the central filtering algorithm must be able to deal with measurement data from many different types of sensors, observing different areas of the vehicle. A feature-level architecture, depending on how it is implemented, may increase the system's complexity by distributing certain tracking functions, such as association, to the sensors, requiring feedback of already-fused object information. Hybrid architectures may also end up being complex, as they attempt to combine aspects of both low-level and high-level architectures. A high-level architecture tends to have the simplest form, but the methods required for fusing the object data may be complexer due to the fact that filtered object data instead of sensor measurement data is used, which can result in correlation problems between the data. Overall, all sensor data fusion architectures have some degree of complexity, which needs to be properly addressed.

Modularity

For future ADAS, where sensor data will have to be shared among multiple applications, modularity is of great importance. A modular sensor data fusion architecture also allows the same algorithms to be used with different types of sensors and sensor configurations, which could be a cost effective method of offering different driver assistance systems in different vehicles, all based on the same underlying software and hardware architecture.

A low-level fusion architecture does not scale very well due to the fact that raw sensor data must flow to a single central fusion unit. The raw sensor data may come in various formats, depending on the sensor type and its measuring principle, requiring an adaptation of the fusion module for every new type of sensor measurement. Feature-level fusion architectures improve on this by extracting features, which correspond to a specific object model, before processing at the central fusion unit. However, as seen in [70, 71], features

may still be tied very closely to the type of sensor used. Previous work has also concluded that low-level architectures have a reduced modularity and scalability compared to high-level architectures [124]. Additionally, the fact that a central fusion unit is used requires that it bear the burden of most of the processing in the overall system architecture.

Compared to a low-level or feature-level architecture, a high-level architecture is quite modular. Sensor data is abstracted at a much higher level, simplifying the interface between the sensors and the fusion unit. This also distributes the processing of sensor measurements among the sensor themselves, leading to a more efficient use of computational resources in the overall system. However, model and filter properties used by the sensors may be drastically different and could affect the fusion result. Previous work mentions that some applications may even need to use different filtering properties [75], though concrete examples proving this theory are not given. It is apparent that high-level fusion is a more modular architecture and better suited for serving multiple driver assistance applications. The disadvantage is that less is known about the details of the filtering algorithms used in the sensors and certain applications may have to be adjusted to use less than optimal object data.

Sensitivity to Errors and Bias

Most sensors in environment perception only have the ability to measure position with a high degree of accuracy. A velocity and acceleration estimate are usually derived from the position over time. In a low-level or feature-level fusion architecture, sensor data from one time step to the next may come from different sensors, which can introduce errors in the derived velocity and acceleration states. Errors due to sensor bias or time jitter may occur between sensors. The following expression shows how velocity is a function of time, position and potential errors when the position data from one time step to the next are from different sensors:

$$v(k) = \frac{x^i(k) + b^i(k) - (x^j(k-1) + b^j(k-1))}{t_k^i - t_{k-1}^j + t_{k-1}^{\text{jitter}}} \tag{2.1}$$

where $x^i(k)$ is the position measurement from sensor i at time t_k^i, $x^j(k)$ is the position measurement from sensor j at time t_{k-1}^j, $b^i(k)$ is the bias from sensor i at t_k^i, $b^j(k-1)$ is the bias from sensor j at t_{k-1}^j and t_{k-1}^{jitter} is the jitter in the time stamps between the two sensors. The bias between the two sensors and the time jitter affect the accuracy of the derived states. It is possible to minimize the effect of the time jitter by modeling it as an uncertainty, as done in [219]. The jitter between sensor time stamps can also be minimized by using various techniques to determine a precise time stamp of sensor data [127, 294].

In a high-level sensor data fusion architecture, the derived velocity and acceleration states are calculated over time from the same sensor. The velocity error due to the sensor bias is eliminated because the bias from the same sensor between measurement time steps is approximately the same, $b^i(k) \approx b^i(k-1)$. Time jitter can also be neglected since a sensor's internal cycle time is known very precisely, so $t_{k-1}^{\text{jitter}} \approx 0$. Reformulating (2.1) such that the derived states are estimated from a single sensor, as would be the case in a

high-level fusion architecture, results in the elimination of the bias and jitter terms:

$$v(k) = \frac{x^i(k) + b^i(k) - (x^i(k-1) + b^i(k-1))}{t_k^i - t_{k-1}^i + t_{k-1}^{\text{jitter}}} = \frac{x^i(k) - x^i(k-1)}{t_k^i - t_{k-1}^i} \tag{2.2}$$

Using a high-level fusion architecture reduces the effect due to sensor bias errors and timing jitter in the derived states. However, in a high-level fusion architecture, precise timing of sensor data and the minimization of bias is still desired for accurate state estimation fusion.

Practicality

Theory and practice can sometimes be quite a world apart and therefore it is important to consider the practicality of a fusion architecture in the context of the automotive industry and ADAS. Issues such as supplier networks, vehicle communication architecture, cost, ability to offer various configurations, etc. are all quite relevant when choosing an architecture for combining sensor data in a practical automotive application. The issue of practicality must also be considered for future mass-production driver assistance applications, where the practical constraints on what can be implemented and eventually sold to customers is much more restricted than in research applications.

In [200], it is mentioned that original equipment manufacturers (OEMs) require the use of several sensors from different sensor manufacturers in order to realize all driver assistance systems. In the future, it will be necessary to combine sensor data for multiple applications in one general architecture, otherwise known as the environment model. This requires strict definitions for sensor interfaces in the architecture for future driver assistance systems. In [75], the use of standard interfaces in the fusion architecture is also designated as an important factor for being able to use sensors from different manufacturers in a single architecture. It is much easier to define a standard interface at an object-level than it is to do so for raw sensor measurements or features, since they can vary greatly between sensors. From an interface standpoint, a high-level architecture with an object interface is better suited in practice.

In practice, OEMs are usually the integrators of a complete driver assistance system [200]. This would also favor a high-level architecture, where the details of sensor measurement processing is left to the sensor manufacturers themselves, simplifying the integration process and thereby reducing costs and increasing modularity.

Practicality may end up being the most important factor in choosing a sensor data fusion architecture that has the potential of serving future driver assistance systems in mass production, as factors such as cost, complexity, and business models may simply take precedence over the fact that the alternative may solely improve accuracy by a few centimeters, a fact that may be irrelevant when realizing most applications.

Another issue to consider is the validation processes required under ISO 26262 [129, 256] for the functional safety of road vehicles. The applied architecture for sensor data fusion may play a deciding role in the chosen architecture for more complex ADAS and automated driving systems.

2.2 Proposed Modular Sensor Data Fusion Architecture

This thesis proposes a novel modular sensor data fusion architecture, designed to meet the needs for future ADAS and highly automated driving application while supporting many such applications in one general, unified architecture [308]. From the previously described architectures, the proposed architecture is derived from a high-level sensor data fusion architecture, with some aspects of a hybrid sensor data fusion architecture. The idea is to keep the architecture simple, highly modular and maintain the highest level of abstraction possible with strictly defined interfaces.

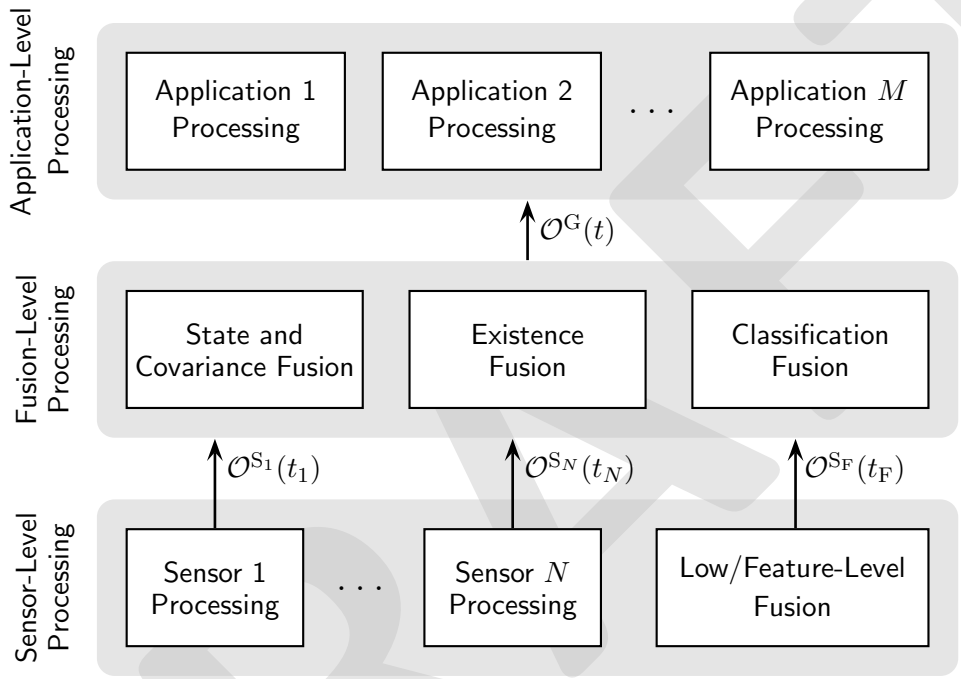


Figure 2.5: Proposed modular sensor data fusion architecture, consisting of three levels of sensor data processing: sensor-level, fusion-level and application-level.

The proposed architecture is organized into three levels of sensor data processing: sensor-level, fusion-level and application-level (Figure 2.5). The main idea is that each level has a specific function and communicates with the next level using a simplified and defined protocol, based on the object model. Each level in the architecture produces an object list that adheres to the defined object model, described in Section 2.2.1, which serves as the input for further processing at the next level. Information about the data and its source is completely abstracted at each level, where the object model contains all of the necessary information, in a general format, required for further processing at the next level. This generalized format allows for the architecture to be completely modular, due to the fact that sensor-specific information is completely eliminated and abstracted. Therefore, as long as sensors adhere to the defined communication protocol using the object model, it is possible to add, remove and exchange sensors as required for a specific configuration, without the need to make any changes to the rest of the architecture or the implemented fusion algorithms.

The remainder of this section will describe the object model and the details of the processing flow and function of each level in the architecture. The remainder of this thesis

will focus on the algorithms required to completely realize and make use of the object model at each level in the architecture.

2.2.1 Object Model

Revisiting (1.8), the goal is to estimate an object list $\mathcal{O}(t)$ at time t based on the previous estimate of the object list and any new sensor information. Before continuing, it is important to define the interface for an object list $\mathcal{O}$. This thesis suggest an object interface which is complete in terms of describing an object's state, detection quality and type attributes. The interface between the different levels of the proposed sensor data fusion architecture is defined by an object list, $\mathcal{O}$, of the perceived objects in the vehicle's environment

$$\mathcal{O} = \{O_1, O_2, \ldots, O_N\} \tag{2.3}$$

where O_i is a a single object in the list, defined as

$$O_i = \{\hat{\mathbf{x}}, \mathbf{P}, \hat{\mathbf{d}}, \mathbf{d}_{\sigma^2}, p(\exists\mathbf{x}), \mathbf{c}, \mathbf{f}\} \tag{2.4}$$

where $\hat{\mathbf{x}}$ is the estimated state vector, $\mathbf{P}$ is the state covariance matrix, $\hat{\mathbf{d}}$ is the estimated object dimension vector, $\mathbf{d}_{\sigma^2}$ the dimension uncertainty vector, $p(\exists\mathbf{x})$ is the probability of existence, $\mathbf{c}$ is the classification vector and $\mathbf{f}$ is the feature vector. The set $\{\hat{\mathbf{x}}, \mathbf{P}\}$, consisting of the state estimate and its error covariance, is commonly referred to as a *track*. This term will be used extensively throughout this thesis, particularly in Chapter 4.

The state vector, $\mathbf{x}$, defines the object's location and dynamic information relative to the host vehicle's coordinate system, as shown in Figure 2.6. The state vector for the object model is defined in this thesis as

$$\mathbf{x} = \begin{bmatrix} x & y & v_x & v_y & a_x & a_y & \psi & \dot{\psi} \end{bmatrix}' \tag{2.5}$$

where x and y are the position of the detected object's geometrical center, v_x and v_y the absolute velocity, a_x and a_y the absolute acceleration, ψ the orientation (or yaw angle) and $\dot{\psi}$ the yaw angle rate. The object's dimension vector

$$\mathbf{d} = \begin{bmatrix} l & w \end{bmatrix}' \tag{2.6}$$

defines a two-dimensional rectangular model for each object, consisting of a length, l, and a width, w. For a three-dimensional object model, the dimension vector could be extended to also include an object height. The dimension uncertainty vector $\mathbf{d}_{\sigma^2}$ contains the variances of the dimension vector $\mathbf{d}$.

This type of rectangular object model is able to represent all dynamic objects which are relevant in driver assistance applications. It is important to note, however, that the defined object model can vary and should be extended if a new sensor is available that can detect a new aspect of an object, such as height, for example. It is not necessary that the full state vector be determined by all of the sensors, as in some cases, this may not be technically possible. Therefore, each sensor communicates only a subset of the states of the complete state vector. The information about which sensor states a sensor can detect can be saved in a configuration file or can be transmitted directly with each new object list.

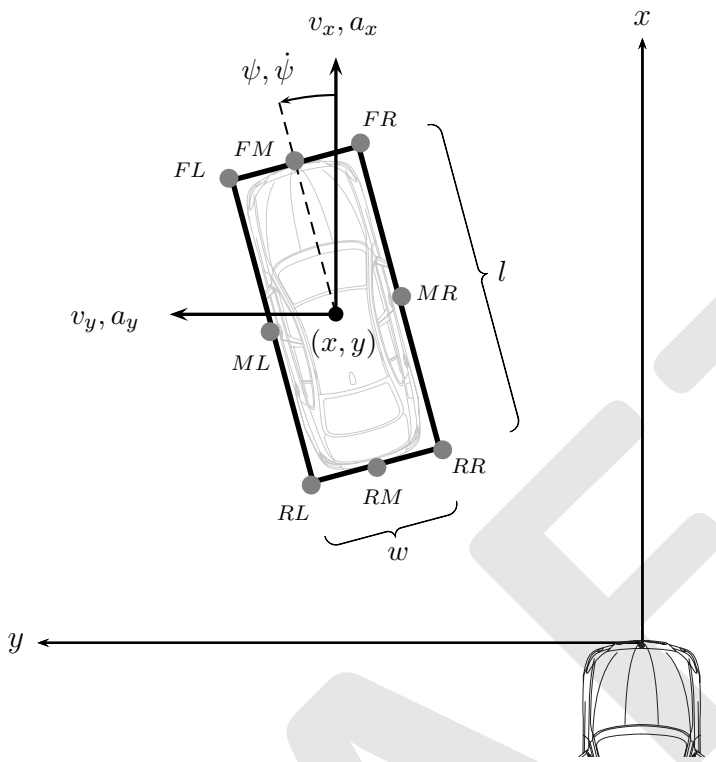


Figure 2.6: Planar rectangular object model depicted with the states and features.

In addition to the position and dynamic information, eight features of the object model are defined and are given by

$$\mathbf{f} = \begin{bmatrix} FL & FR & RL & RR & FM & RM & ML & MR \end{bmatrix}' \tag{2.7}$$

where the features correspond to a corner or a side of the perceived object. For example, RL is the rear-left corner of an object, RM is the rear-middle side, FL is the front-left and so on. The middle features, such as RM or ML, define that the side of an object in general is observed and does not specifically mean that the exact mid-point of the side is observed. The feature information is necessary in order to connect the state vector information, in particular the position, to a larger rectangular object model. In automotive applications, objects cannot be modeled by a simple point-targets, therefore such a geometrical object model is required. In order to allow for the proper fusion of objects from different sensors, it is necessary that the position defined in the state vector is related to one of the defined features of the object. Note that other geometrical models are also possible, such as circles or polylines. As with the state vector, the feature vector should be extended if new information can be detected, such as height. This thesis, however, focuses on the planar rectangular geometrical model, as this model suffices for detecting relevant dynamic objects in the vehicle's environment, particularly in highway scenarios.

An object's classification is defined through a vector of classification likelihoods, $\mathbf{c}$, where the classes are defined as

$$\mathbf{c} = \begin{bmatrix} C_{\text{Car}} & C_{\text{Truck}} & C_{\text{Motorcycle}} & C_{\text{Bicycle}} & C_{\text{Pedestrian}} & C_{\text{Stationary}} & C_{\text{Other}} \end{bmatrix}' \tag{2.8}$$

Note that the classification vector must be normalized.

The details on how an object model is estimated for each level in the architecture will be described and evaluated throughout this thesis.

2.2.2 Sensor-Level

The first level of the proposed architecture is the sensor-level. At this level, each sensor used for environment perception processes its data independently and parallel to all of the other sensors. Each sensor then produces a local object list, depending on the sensor's capabilities, as described in the previous section.

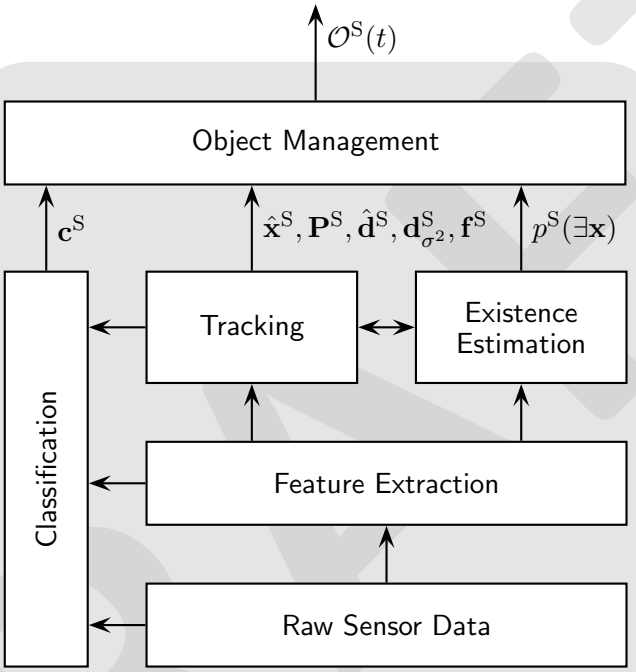


Figure 2.7: Typical sensor data processing flow of a single sensor at the sensor-level of the proposed sensor data fusion architecture.

A common data processing structure at the sensor-level is shown in Figure 2.7. In this example, the raw sensor data is pre-processed through a feature extraction algorithm. The extracted features, which may correspond to the feature vector, $\mathbf{f}^{\mathrm{S}}$, of the local object model, is then tracked over time using common filtering and tracking algorithms [25] to generate the object's state vector, $\mathbf{x}^{\mathrm{S}}$, and state covariance matrix, $\mathbf{P}^{\mathrm{S}}$, as well as an object's dimensions, $\mathbf{d}^{\mathrm{S}}$, and the dimension uncertainty, $\mathbf{d}^{\mathrm{S}}_{\sigma^2}$. Parallel to state vector estimation, the object's existence probability, $p^{\mathrm{S}}(\exists \mathbf{x})$, and classification probability vector, $\mathbf{c}^{\mathrm{S}}$, are also determined.

However, in some cases, it may be beneficial to use a low-level or feature-level fusion architecture between certain sensors. For example, for forward-facing sensors, a feature-level fusion between a camera and radar sensor may provide a more reliable object list, due to the camera's superior object classification performance. In other cases, a low-level fusion architecture may make sense for practical reasons, such as cost, when the same type of sensor from the same manufacturer is used several times. In the cases where such an architecture is desirable, the sensor-level processing for a group of sensors may consist of a

low-level or feature-level fusion processing structure, as described in Section 2.1.1. In the proposed fusion architecture, the low-level or feature-level fusion is then considered as a virtual sensor, as illustrated in the lower-right of Figure 2.5. Through the defined general object list model, how the object data is produced is completely abstracted, therefore, it does not matter if the object list from the sensor-level is generated from a single sensor or from a low-level or feature-level fusion of a small group of sensors.

2.2.3 Fusion-Level

At the fusion-level of the proposed sensor data fusion architecture, a central fusion module exists for the purpose of generating a sensor-independent global object list, $\mathcal{O}^{\mathrm{G}}$, by combining all of the sensor-level object lists. The data processing flow at the fusion-level is shown in Figure 2.8.

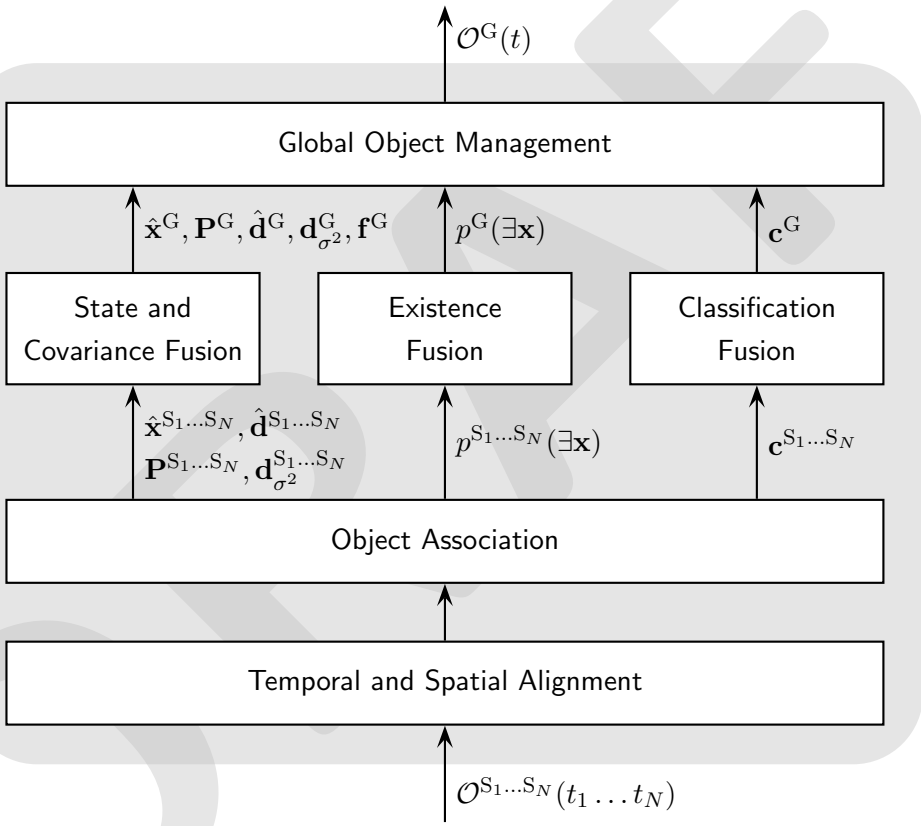


Figure 2.8: Object list data processing flow at the fusion-level for the proposed sensor data fusion architecture.

The first step for the fusion-level data processing is to temporally and spatially align the sensor-level object lists, $\mathcal{O}^{\mathrm{S}_1 \ldots \mathrm{S}_N}(t_1 \ldots t_N)$, from all of the sensors. This includes synchronizing all object lists to the global object list and putting all of the object data in a common coordinate system. Once this is accomplished, the object lists from all of the sensors are associated with one another to determine which objects from different sensors represent the same object in reality. After object association, the state and covariance, the existence probability and classification for associated objects are fused together into a single global object list. The fusion process produces a new, global object list, $\mathcal{O}^{\mathrm{G}}$,

which represents all of the objects in the vehicle's environment that the sensors were able to observe, independent of any sensor specific knowledge. Therefore, sensor knowledge is completely abstracted in the global object list.

The following chapters in this thesis will mainly focus on the algorithms necessary for the alignment, association and fusion of sensor data at the fusion-level, as described above.

2.2.4 Application-Level

At the application-level, the global object list, $\mathcal{O}^{\mathrm{G}}$, is available to any driver assistance application that requires object detection for its functionality. Combined with other environment perception modules, such as lane detection, digital maps, host vehicle localization and orientation, etc., an application can apply its own specific situation assessment algorithms and then trigger a warning, change a state, control an actuator or do whatever is necessary to realize the application. Several applications can run in parallel and use the same global object list that is generated within the proposed sensor data fusion architecture. This also allows the architecture to be highly modular, because applications can be added and modified at no additional cost to the overall object detection environment perception architecture.

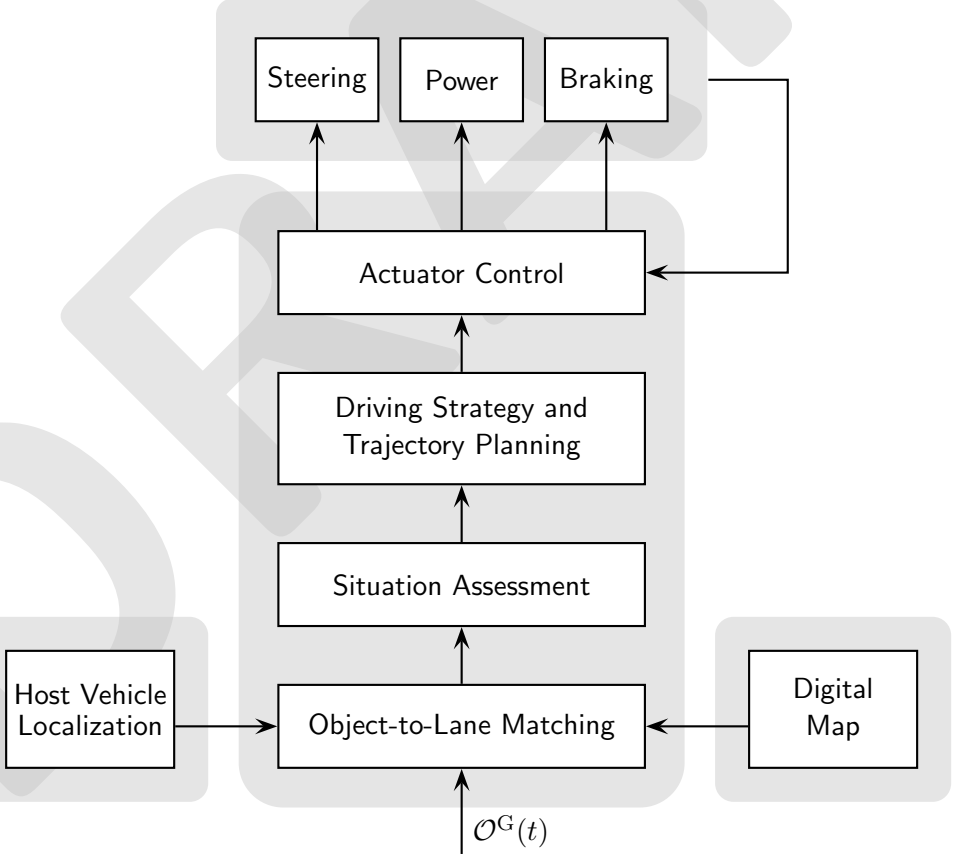


Figure 2.9: Data processing at the application-level for highly automated driving on highways.

An example of an ADAS developed during this thesis is highly automated driving on highways, a subset of which is the emergency stop assistant. In this application, the vehicle is capable of automated driving in highway situations, including fully autonomous lane

change maneuvers and decision making. The situation assessment and decision making module for highly automated driving and other advanced driver assistance systems was developed in [5–8], [314, 317, 318]. The data processing flow for realizing highly automated driving on highways is illustrated in Figure 2.9. In addition to the global object list, a digital map of the highway environment and the localization of the host vehicle must be known. The host vehicle localization algorithms are able to determine the global position of the vehicle within to centimeter accuracy [224]. Using a highly-accurate digital map, such as the ones developed in [224] and [324], the host-vehicle and the detected objects from the global object list are matched onto lanes in the digital map. Based on the results of this object-to-lane matching, a situation assessment algorithm determines which objects are relevant and how critical the current highways situation is [6]. Objects are also predicted in order to allow for an early reaction to certain situation, such as lane changes [317, 318]. The situation assessment is then used to make decisions about the vehicle's driving strategy and generates trajectories for lateral and longitudinal vehicle guidance, even during the driver take-over request phase [316]. The parameters for these trajectories are then applied to control algorithms which alter the vehicle's steering, throttle and braking, therefore fully automatically moving the vehicle on the highway.

Other driver assistance applications can also benefit from the global object list. Active cruise control (ACC), for example, would simply need to select the relevant object in front of the host vehicle from the global object list in order to perform its function. Similarly, a lane change warning or blind spot monitoring application would need to find the objects which may make it difficult for the host vehicle to perform a lane change maneuver, and if such an object is found, a warning can be given to the driver. The advantage of the proposed modular sensor data fusion architecture is that one unified architecture for object recognition, within the environment model, by means of the global object list, can service many driver assistance and active safety applications and functions.

Discussion

Future drive assistance systems will require multiple sensors for various applications. This will require an architecture for combining, or fusing, data from many sensors. In this chapter, an overview of possible sensor data fusion architecture configurations was presented. Based on the qualitative comparison of the architectures, a novel sensor data fusion architecture was developed, which has a high potential for meeting the requirements of future driver assistance systems, in particular highly automated driving applications.

Low-level and feature-level fusion architectures are widely known to offer the best accuracy in terms of state estimation and classification performance. However, their practical disadvantages due to bandwidth, complexity and modularity make them unsuited for more complex ADAS applications, where many sensors from different manufacturers with different measuring principles are necessary. A high-level sensor data fusion architecture is quite modular and scales really well as the sensor configuration becomes more complex. However, more effort is required in the development of the fusion algorithms. Inaccuracies may also occur when sensors process their data with widely different model and filter parameters. Another option is to develop a hybrid architecture, which combines sensor data at both a low and high level.

In Section 2.2, a novel sensor data fusion architecture, based on a high-level architecture,

was proposed. The goal of the proposed architecture was to keep a simple, but strict, interface between interacting modules in the architecture. A planar rectangular object model, which is able to represent all relevant dynamic objects for automotive environment perception, is used as the interface between the three different levels of the architecture in this thesis. The interface is completely abstracted from any sensor-specific information, as generic quality measures are used to represent different sensors' performance. The architecture is split into three simple levels: sensor, fusion and application. The sensor-level is where each sensor independently processes its data in order to produce an object list, as defined by the object model. At the sensor-level, however, low-level or feature-level fusion may also occur, making the proposed architecture a part hybrid fusion architecture. The fusion-level combines the sensor-level object lists using various fusion algorithms for the state and covariance, probability of existence and classification in order to generate a single unified global object list. This global object list can then by used by any number of application-level driver assistance applications for further interpretation. Communication using the object model between the levels of the proposed architecture reduces bandwidth and complexity while increasing modularity and practicality for automotive applications.

The algorithms required to realize the proposed sensor data fusion architecture will be described in the chapters that follow. The effectiveness of the proposed architecture and its algorithms will be demonstrated through simulation and with a prototype vehicle with real sensor data, designed for highly automated driving. The goal in the following chapters is to show that the proposed architecture is able to achieve complete surround-view environment perception using relatively simple vehicle-integrated sensors, while offering similar performance in surround perception as more complex low-level architectures using larger and more capable sensors.

3 Fusion Strategy and Object Association

As shown in the previous chapter, in the proposed high-level sensor data fusion architecture, each sensor produces an object list of the detected objects within its field-of-view. The problem that presents itself at the fusion level is how to combine sensor object data from different sensors together. Apart from the actual process of data fusion, other problems need to be considered. First, the object data must be spatial and temporally aligned to a common reference frame. Second, in which order and how sensor data is fused together to generate fused global objects must be considered. Finally, it must be determined which objects from different sensors actually represent the same object in reality, therefore providing the association between objects with which to carry out fusion. This chapter aims to present solutions to each of these problems.

3.1 Data Alignment

Before it is possible to associate object data with one another it is necessary to align the object data into a common coordinate and spatial frame. In this section, spatial and temporal alignment of object data is described.

3.1.1 Spatial

Object data from different sensors must first be put into a common spatial coordinate frame. The vehicle coordinate system, as defined in DIN 70,000 [83], is used as a common global coordinate system for local object detection, with positive x defined in the forward longitudinal direction and positive y in the lateral left direction. The origin of the coordinate system in this thesis was chosen as the vehicle's emblem on the hood. Other common origins, such as the middle of the rear-axis are also common. The problem of spatial alignment is that of transforming an object defined in a sensor's coordinate system to that of the common vehicle coordinate system, where the sensor coordinate system in a planar environment is translated relative to the vehicle coordinate system with δ_x and δ_y and rotated by θ, as shown with a rearward facing sensor in Figure 3.1. In non-planar environments, translation and rotation amongst a third dimension may also be necessary.

An object in a sensor's coordinate system is transformed into the common global vehicle coordinate system with

$$\begin{bmatrix} \mathbf{x}^{\text{veh}} \\ 1 \end{bmatrix} = \mathbf{H}_{\text{sensor}\rightarrow\text{veh}} \begin{bmatrix} \mathbf{x}^{\text{sensor}} \\ 1 \end{bmatrix} \tag{3.1}$$

where $\mathbf{H}_{\text{sensor}\rightarrow\text{veh}}$ is the transformation matrix between the sensor coordinate system and the common global vehicle coordinate system. The state of an object, $\mathbf{x}$, is that of the object model interface, as defined in Section 2.2.1. Given the two-dimensional translation,

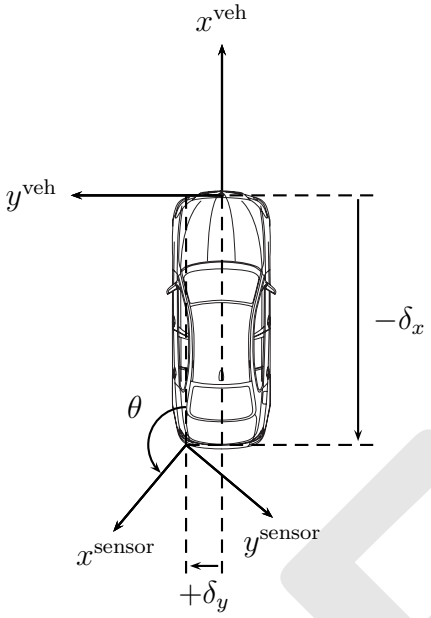


Figure 3.1: Two-dimensional translation (δ_x and δ_y) and rotation (θ) of a sensor's coordinate system relative to the common global vehicle coordinate system.

δ_x and δ_y, and the rotation about the vertical axis, θ, between the sensor and common coordinate system gives the following transformation matrix:

$$\mathbf{H}_{\text{sensor}\to\text{veh}} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 & 0 & 0 & 0 & 0 & \pm\delta_x \\ \sin\theta & \cos\theta & 0 & 0 & 0 & 0 & 0 & 0 & \pm\delta_y \\ 0 & 0 & \cos\theta & -\sin\theta & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sin\theta & \cos\theta & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos\theta & -\sin\theta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sin\theta & \cos\theta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \theta \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.2}$$

The sensor-to-vehicle transformation matrix, $\mathbf{H}_{\text{sensor}\to\text{veh}}$ rotates the position, velocity and acceleration vectors into the common global vehicle coordinate system along with the proper relative translation and rotation of the coordinate systems.

3.1.2 Temporal

Once in a common spatial coordinate system, objects that are to be associated with one another must also be put into a common temporal frame. This is accomplished by using a kinematic model to predict the state of an object to some time in the future. Given that a sensor took a measurement of an object and updated its state at time $k-1$ and needs to be temporally aligned to time k for data association and fusion, then the standard Kalman filter prediction formulation can be used to predict the object's state and error covariance

forward in time:

$$\hat{\mathbf{x}}(k|k-1) = \mathbf{F}(k,k-1)\hat{\mathbf{x}}(k-1|k-1) + \mathbf{B}(k)\mathbf{u}(k) \tag{3.3}$$

$$\mathbf{P}(k|k-1) = \mathbf{F}(k,k-1)\mathbf{P}(k-1|k-1)\mathbf{F}(k,k-1)' + \mathbf{Q}(k) \tag{3.4}$$

where $\hat{\mathbf{x}}(k-1|k-1)$ is the object state at $k-1$, $\hat{\mathbf{x}}(k|k-1)$ is the predicted state, $\mathbf{u}(k)$ is the control vector, $\mathbf{B}(k)$ is the control transformation matrix, $\mathbf{F}(k,k-1)$ is the kinematic model, $\mathbf{Q}(k)$ is the process noise matrix and $\mathbf{P}(k|k-1)$ is the object's predicted error covariance matrix. In moving vehicle applications, the control vector is used to compensate for the host vehicle's movement in a given time interval. In this thesis, a constant-velocity kinematic model is used for temporal alignment of an object's state (see Section 4.1 for more information on kinematic models). More complex kinematic models, such as a constant turn-rate model could also be used, which would result in the extended Kalman filter prediction formulation for a non-linear model.

3.2 Fusion Strategy

It is important to consider the manner in which sensor-level object data is processed at the fusion-level in order to create global objects. The method of processing object data for fusion will be referred to as the *fusion strategy* in this thesis. Different fusion strategies require different types of fusion algorithms in order to properly create a global track. These different algorithms all have their pros and cons in terms of accuracy and practicality. When deciding which fusion strategy is best for a specific application, it is important to consider that sensors can be asynchronous to one another and possibly deliver their data out-of-sequence with respect to the other sensors in the system. Additionally, each sensor's field-of-view may be quite different, observing different areas, such that a single object is not always observed by all of the sensors. In this section, two typical fusion strategies found in the literature [68, 87],[311, 313] are described: sensor-to-sensor and sensor-to-global fusion strategies.

3.2.1 Sensor-to-Sensor

In a high-level sensor data fusion architecture, the most common fusion strategy is a sensor-to-sensor fusion strategy, also commonly called track-to-track fusion with no memory [268]. In a sensor-to-sensor fusion strategy, sensor-level objects are directly fused together at the fusion-level at pre-defined fusion intervals. In Figure 3.2, an example of a sensor-to-sensor fusion strategy is shown with two asynchronous sensors, where the fusion time is synchronized to the data arrival time of sensor j.

In the example shown, when data from sensor i arrives at time $k_{\mathrm{G}}-3$, it is buffered until data arrives from sensor j at $k_{\mathrm{G}}-2$, which is the fusion processing time. At this time, the buffered data from sensor i is temporally aligned by predicting the object to $k_{\mathrm{G}}-2$, at which time the predicted data from sensor i is directly fused together with the data from sensor j to create a global object. If there are more than two sensors in the sensor configuration, then object data from the sensors is sequentially fused together, i.e. sensor 1 fuses with sensor 2, the result of which is then fused with sensor 3, which is then fused with sensor 4, etc. until all of the sensor data in the buffer since the previous fusion is processed. Note that at each fusion cycle, the global object is newly created, without

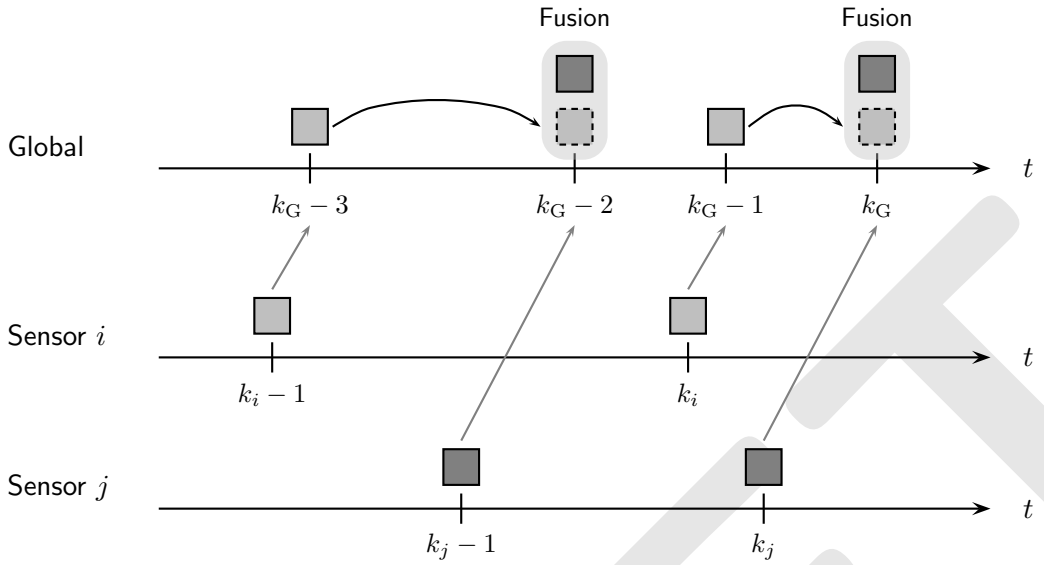


Figure 3.2: Sensor-to-sensor fusion strategy with asynchronous sensors. The local sensor data is buffered and fused together once data from all of the sensors has arrived.

any information from the previously fused global object. Hence, a sensor-to-sensor fusion strategy is memory-less, since the global tracks are unaware of the fusion result from the previous fusion cycle.

A sensor-to-sensor fusion strategy is quite popular in systems with a simple sensor configuration with well synchronized sensors. Many track-to-track fusion algorithms exist for the processing of data using a sensor-to-sensor fusion strategy (see Appendix A). The sensor-to-sensor fusion strategy, as described above, has been widely used in automotive perception applications [100, 126, 174, 262].

3.2.2 Sensor-to-Global

Another method of processing sensor-level object data at the fusion-level is to use a sensor-to-global fusion strategy, also commonly referred to as track-to-track fusion with memory [270]. In this configuration, sensor-level object data is processed as it arrives and is directly fused into a global object which is maintained over time at the fusion-level. An example of a sensor-to-global fusion strategy with two sensors is depicted in Figure 3.3.

In the above example, the sensor-level object data, such as that from sensor i at $k_G - 3$, is directly fused into a globally maintained object at the fusion-level. When data from sensor j arrives at $k_G - 2$, the global object is predicted to this arrival time, where the sensor-level data from sensor j is then similarly fused directly into the global track. The advantage of such a fusion strategy is that sensor data is processed immediately as it arrives, without the need for buffering sensor data and waiting for the pre-defined fusion processing intervals. Since a global object is maintained over time at the fusion-level, data from the previous fusion is not lost when new sensor data arrives. The sensor-to-global fusion strategy is also quite flexible to all sorts of sensor configurations, due to the fact that sensor data is not fused directly together and is processed completely independent of one another. This allows the sensor-to-global fusion strategy to support a "plug-and-play" type of sensor configuration, where sensors can be added or removed at ease, without effecting

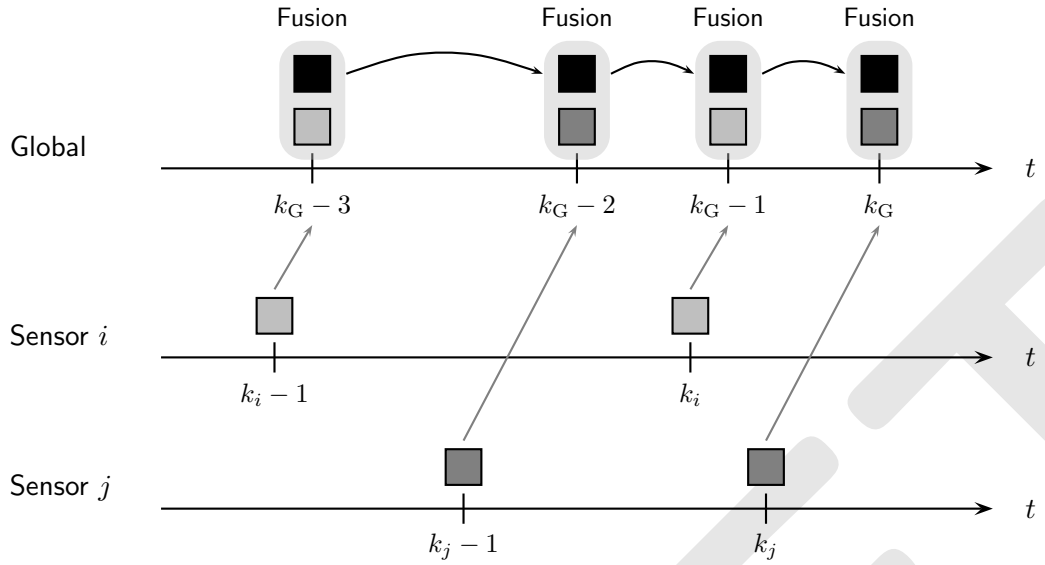


Figure 3.3: Sensor-to-global fusion strategy. The local sensor data is directly fused into a global track, independent of the other sensors.

the algorithms at the fusion-level.

Track-to-track fusion algorithms and their pros and cons in a sensor-to-global fusion strategy are described in Chapter 4. A common method of implementing a sensor-to-global fusion strategy is to simply process the sensor-level object data as if they were measurements to a Kalman filter algorithm, an approach that is quite common in automotive applications [174, 182]. However, this leads to problems, as the sensor-level object data is correlated over time, as will be described in Section 4.2. Otherwise, a sensor-to-global fusion strategy has been quite uncommon in automotive applications and has only recently been considered as a means of fusing sensor data for surround environment perception [311, 313].

3.3 Association

A vital and challenging problem in object tracking is the problem of data association. In the proposed high-level sensor data fusion architecture, data association occurs between already-filtered objects from various sensors. In the literature, data association at this level is typically called *track-to-track association* [20, 21, 214, 269] and has been applied to some automotive applications [101, 126]. In a sensor-to-global fusion strategy, as described in the previous section, object association occurs between sensor-level objects and global objects at the fusion-level. The problem boils down to deciding which objects from a specific sensor represent the same object at the fusion-level. If no such object exists, meaning that an association with an already existing object was not possible, then the sensor-level object is initialized as a new global fusion-level object.

The problem of object association is depicted in Figure 3.4. In this example, a sensor-level local object, O_i^{S}, is to be associated with a fusion-level global object, O_j^{G}. The error covariance, or uncertainty in the state, for the position is depicted with the dashed ellipses. The global object shows a larger lateral uncertainty than the sensor-level object, but has

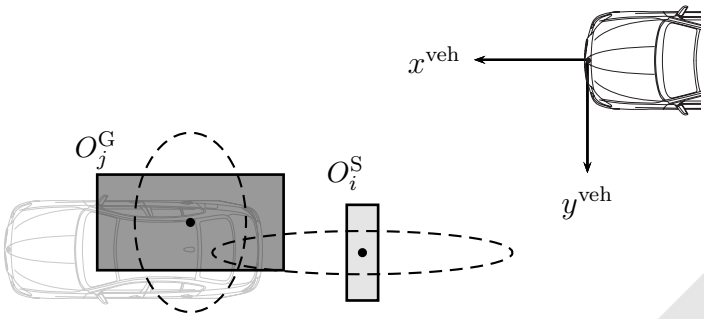


Figure 3.4: Problem of object association between a fusion-level global object (O_j^{G}) and a sensor-level local object (O_i^{S}).

a smaller longitudinal uncertainty. This is a typical situation in sensor data fusion, where sensor performance is complementary to one another.

Object association can be formulated using two hypotheses H_0 and H_1. If a fusion-level object O_j^{G} and a sensor-level object O_i^{S} represent the same object in reality, then evaluating H_0 shows that the values of O_j^{G} and O_i^{S} are the same, or their difference is zero:

$$H_0 : O_j^{\text{G}} - O_i^{\text{S}} = 0. \tag{3.5}$$

Therefore, H_0 represents a true association, whereas the opposing hypothesis H_1 represents a failed association:

$$H_1 : O_j^{\text{G}} - O_i^{\text{S}} \neq 0. \tag{3.6}$$

The goal of object association is to test which hypothesis, given two objects, is more likely.

In this section, novel algorithms will be presented for solving the problem of object association in a multi-object and multi-sensor environment with the planar rectangular object model. In particular, a novel approach is presented for dealing with various geometrical constellations of objects, where not necessarily the same point on the two objects is directly used in data association, as is typical of state-of-the-art approaches in the literature.

3.3.1 Architecture

For point-targets, such as applicable in aerospace applications, the object association problem has been widely studied and many different algorithms exist [16, 18, 40]. In such cases, the problem is simpler due to the fact that the object's dimensions are insignificant compared to the distance the object is from the sensor. In automotive application, this is not the case since object dimensions and their distances from the host vehicle can be of the same order of magnitude. In order to properly detect and represent objects in automotive surround environment perception, a rectangular object model was introduced in Section 2.2.1. A key element of the object model for association is the feature vector $\mathbf{f}$, which is a boolean vector describing which corner and side features of the object were observable. Sensor-level tracking algorithms do not track the geometrical center of an object, as this has been shown to produce poor performance [139, 216, 239, 255], but rather track a specific feature of an object. It is common that the tracked feature can change over time and its choice is vital in successful data association [12, 139, 239]. Additionally, depending

on the sensor configuration, an object may only be partially observed by different sensors due to their restricted field-of-view.

Object association at the fusion-level between two objects consists of three steps: feature selection, state vector association and geometrical association. As with the traditional association algorithms, a quantitative value describing the quality of the association between two objects is calculated, where an extended version of the Mahalanobis distance will be used (Section 3.3.3). Before calculating the association between two objects, the objects and their feature-constellation between one another must be examined such that the proper feature is selected for state vector association (Section 3.3.2). In some cases, a common feature for association may not be available, such that only one dimension of the features in the rectangular object model is used in state vector association; association with the other dimension is solved using a simple geometrical association algorithm (Section 3.3.4). The association result between all fusion-level objects and new sensor-level objects are calculated and entered into an association matrix $\mathbf{A}$, where a single entry in the matrix is denoted as $a_{i,j}$. This association matrix is then solved in order to find a global unique association between objects (Section 3.3.6). The overall processing flow for object association is shown in Figure 3.5.

3.3.2 Feature Selection

As mentioned in the previous section, successful association depends on a correct feature selection based on the object model presented in Section 2.2.1. Let a feature be described by $\mathbf{p}_\mathrm{f}$ where $\mathbf{p}$ is the Cartesian coordinate of the feature and f denotes the feature on the rectangular object model. A feature $\mathbf{p}_\mathrm{f}$ is defined in both vehicle and object coordinate systems, $\mathbf{p}_\mathrm{f}^\mathrm{veh}$ and $\mathbf{p}_\mathrm{f}^\mathrm{obj}$, respectively, where the origin of the object coordinate system is the geometrical middle of the rectangular object model. A feature in the object coordinate system is transformed into the vehicle coordinate system using

$$\mathbf{p}_\mathrm{f}^\mathrm{veh} = \mathbf{H}_{\mathrm{obj}\to\mathrm{veh}}\mathbf{p}_\mathrm{f}^\mathrm{obj}. \tag{3.7}$$

where

$$\mathbf{H}_{\mathrm{obj}\to\mathrm{veh}} = \begin{bmatrix} \cos\psi & -\sin\psi & x \\ \sin\psi & \cos\psi & y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

is the transformation matrix from the object to the vehicle coordinate system and $\mathbf{p}_\mathrm{f}^\mathrm{obj} = \begin{bmatrix} x_\mathrm{f}^\mathrm{obj} & y_\mathrm{f}^\mathrm{obj} & 1 \end{bmatrix}'$. Depending on the feature to be transformed, $x_\mathrm{f}^\mathrm{obj}$ and $y_\mathrm{f}^\mathrm{obj}$ are chosen according to Table 3.1 using the length, l, and width, w, of the object from its estimated dimension vector $\hat{\mathbf{d}}^\mathrm{obj}$. Furthermore, in some cases, as will be described later in this section, only a single dimension, either x or y, is necessary for association. Such a reduced feature is denoted as $\mathbf{p}_{\mathrm{f},x}$ or $\mathbf{p}_{\mathrm{f},y}$.

The remainder of this section describes the different possible feature constellations, shown in Figure 3.6, between two objects to be associated with one another. Depending on the feature constellation, the position of the two objects to be associated are transformed to the proper feature for association. State vector association, described in Section 3.3.3, is then performed on the transformed feature, the object's orientation and its dynamics. If necessary, in the case of a feature constellation where only a single dimension of the

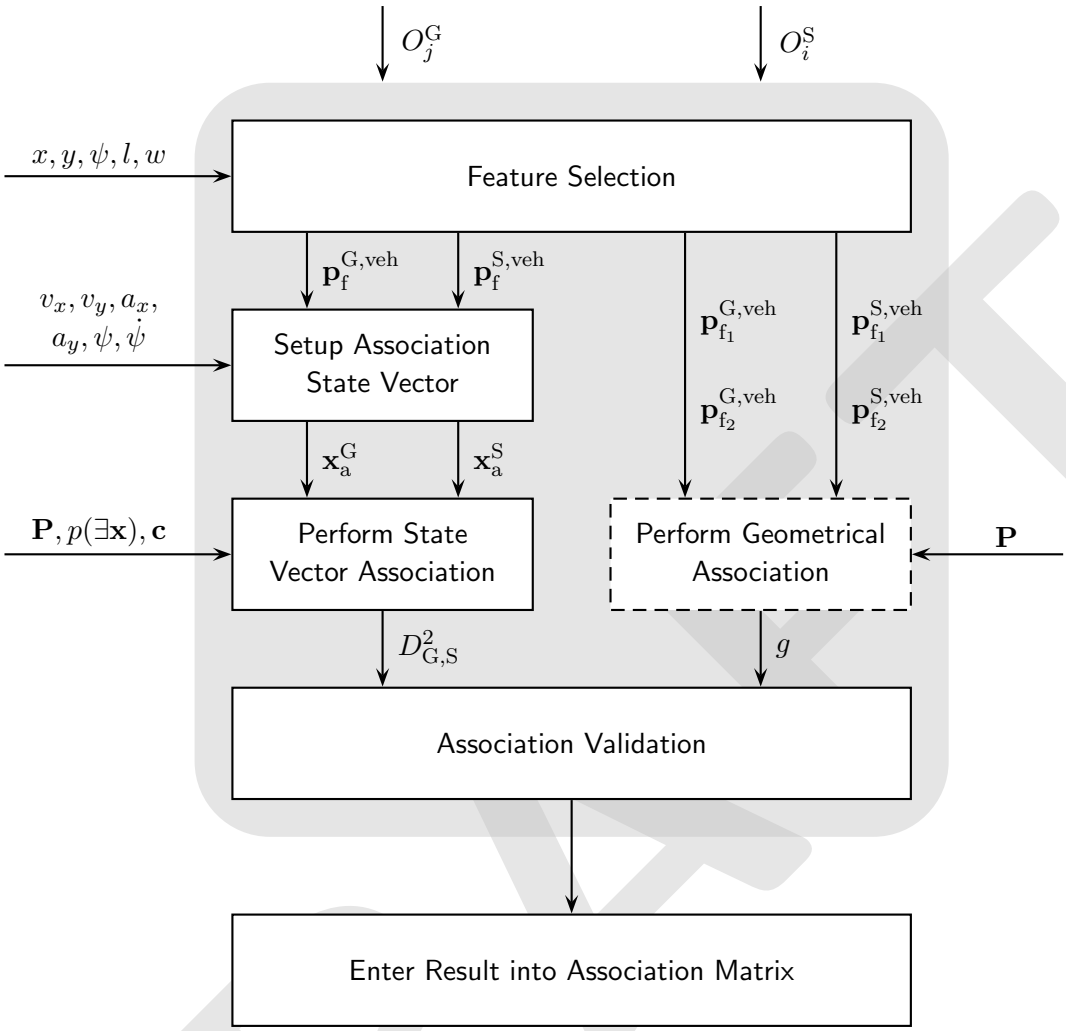
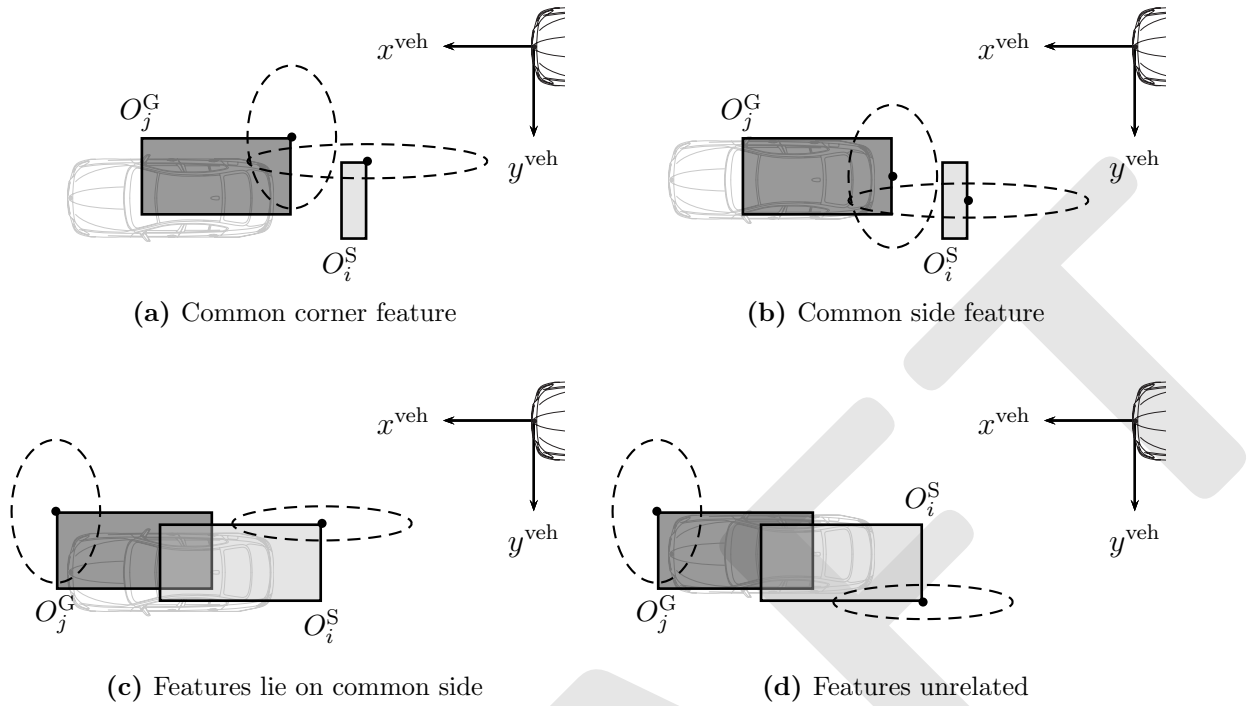


Figure 3.5: Processing flow of object association at the fusion-level, where a sensor-level object O_i^{S} is tested for association to a global object O_j^{G}.

Table 3.1: Feature coordinates of the rectangular object model in the object coordinate system.

Feature	$x_{\mathrm{f}}^{\mathrm{obj}}$	$y_{\mathrm{f}}^{\mathrm{obj}}$
$\mathbf{p}_{RM}^{\mathrm{obj}}$	$-l/2$	0
$\mathbf{p}_{RL}^{\mathrm{obj}}$	$-l/2$	$w/2$
$\mathbf{p}_{RR}^{\mathrm{obj}}$	$-l/2$	$-w/2$
$\mathbf{p}_{FM}^{\mathrm{obj}}$	$l/2$	0
$\mathbf{p}_{FL}^{\mathrm{obj}}$	$l/2$	$w/2$
$\mathbf{p}_{FR}^{\mathrm{obj}}$	$l/2$	$-w/2$
$\mathbf{p}_{ML}^{\mathrm{obj}}$	0	$w/2$
$\mathbf{p}_{MR}^{\mathrm{obj}}$	0	$-w/2$



(a) Common corner feature

(b) Common side feature

(c) Features lie on common side

(d) Features unrelated

Figure 3.6: Different feature constellations between two objects to be associated. Each constellation type requires a slightly different variation in the association method.

position is used in state vector association, an additional geometrical association step is performed, described in Section 3.3.4.

-

Common Corner Feature

The simplest feature constellation between two objects is if they are tracking the same *corner* feature, as shown in Figure 3.6a. In this case, the position coordinates of both objects is transformed to the feature and complete state vector association is performed, without the additional geometrical association step. In the example shown in Figure 3.6a, the transformation $\mathbf{p}_{RR}^{\text{veh}} = \mathbf{H}_{\text{obj}\to\text{veh}}\mathbf{p}_{RR}^{\text{obj}}$ for both objects is carried out.

Common Side Feature

A bit more complicated is when both objects are being tracked by the same *side* feature, illustrated in Figure 3.6b. A side feature is not as definite of a representation as a corner feature, and therefore precaution must be taken when considering such a feature. In theory, each sensor should robustly identify the best tracked feature for an object, such that ideally association with a common side feature is the same as with a common corner feature. If a sensor is not able to actually observe a side feature, which may be the case with occluded objects, then the sensor should not identify use feature as the tracked feature.

However, in practice, sensors occasionally misidentify the best feature and identify a side feature in situations where in reality the side feature is not well observed. Therefore, an additional plausibility check is made when associating with a side feature. If the side,

for example the width in Figure 3.6b, is not fully observed, then the feature can vary depending on how well the side is observed between the two objects. In order to determine the accuracy of the common side feature, the consistency of the side on which the feature lies is tested with

$$|w_j^{\mathrm{G}} - w_i^{\mathrm{S}}| < \sigma_{w_j^{\mathrm{G}}} + \sigma_{w_i^{\mathrm{S}}} \tag{3.9}$$

where the width is used in this example. If the difference between the sides of the two objects to be associated, in this case the width, is small enough with respect to the uncertainties, then it is assumed that the common side feature is accurate enough to carry out complete state vector association, as done with a common corner feature. In Figure 3.6b, this is the case and therefore the transformation $\mathbf{p}_{RM}^{\mathrm{veh}} = \mathbf{H}_{\mathrm{obj}\to\mathrm{veh}}\mathbf{p}_{RM}^{\mathrm{obj}}$ for both objects is carried out, without the additional geometrical association step.

If the side fails the test in (3.9), then the state vector is transformed such that the stable coordinate of the feature (for RM the stable coordinate is x, for example) is carried into the state vector association algorithm using the reduced feature transformation $\mathbf{p}_{RM,x}^{\mathrm{veh}} = \mathbf{H}_{\mathrm{obj}\to\mathrm{veh},x}\mathbf{p}_{RM,x}^{\mathrm{obj}}$. The uncertain coordinate, in this example y, is then associated with the help of the side, in this example the width, w, using the geometrical association algorithm.

Features Lie on Common Side

Another complicated feature combination between two objects is when each object tracks and observes a different feature, but the features lies on a common side, as depicted in Figure 3.6c where the features both lie on the right side of the length. In this case, there is no other choice but to use the reduced feature transformation for the common coordinate, in this case y, where object O_j^{G} is transformed with $\mathbf{p}_{FR,y}^{\mathrm{veh}} = \mathbf{H}_{\mathrm{obj}\to\mathrm{veh},y}\mathbf{p}_{FR,y}^{\mathrm{obj}}$ and object O_i^{S} with $\mathbf{p}_{RR,y}^{\mathrm{veh}} = \mathbf{H}_{\mathrm{obj}\to\mathrm{veh},y}\mathbf{p}_{RR,y}^{\mathrm{obj}}$. This common coordinate in the position is then used in state vector association, whereas the uncommon coordinate is associated with the geometrical association algorithm along the x-axis of the object using the lengths of the two objects.

Features Unrelated

The most uncommon combination of features during association is when the features of the objects to be associated are completely unrelated, as shown in Figure 3.6d. This case usually occurs when the tracked and observed feature of an object from a sensor or the global fusion module is misidentified. However, the association algorithm must still be able to cope with such a situation.

In this case, a heuristic approach is chosen for determining the most likely feature constellation. This begins with a two-dimensional state vector association test of all four corner features; if a common corner feature is found, where the association result of one or more common features lies below a certain threshold, then the minimum common feature result is used for complete state vector association. If this fails, then the common coordinate of the sides are tested separately using the reduced feature transformation, once for x and once for y, where the result of the reduced state vector association on the single coordinates is tested against a threshold. If this also fails, then the objects are considered unassociated.

3.3.3 State Vector

In state vector association, also known as *track-to-track association* in the literature, two vectors, assumed to be normally distributed, are to be tested for association. For object association, the state vector for association consists of a feature position (or reduced feature if only a single coordinate is considered), the orientation and dynamics of the object, such that the state vector for association, $\mathbf{x}_\mathrm{a}$, in the best case is

$$\mathbf{x}_\mathrm{a} = \begin{bmatrix} x_\mathrm{f}^\mathrm{veh} & y_\mathrm{f}^\mathrm{veh} & v_x & v_y & a_x & a_y & \psi & \dot{\psi} \end{bmatrix}'. \tag{3.10}$$

If a reduced feature transformation is used in the feature selection process, then $x_\mathrm{f}^\mathrm{veh}$ or $y_\mathrm{f}^\mathrm{veh}$ is simply omitted. Similarly, if two objects to be associated do not contain all of the dynamic information due to the fact that not all sensors are able to estimate all dynamic features of an object, then only the common information between the two objects is used in state vector association. The length and width of the object is not considered in state vector association and is only used in the geometrical association step, if necessary.

Revisiting (3.5) and (3.6), the same hypotheses are tested using the association state vector in (3.10), where if the true association state vectors of an object are identical, then state vector association is successful

$$H_0 : \mathbf{x}_\mathrm{a}^\mathrm{G} - \mathbf{x}_\mathrm{a}^\mathrm{S} = 0 \tag{3.11}$$

and if they are not identical, then state vector association has failed

$$H_0 : \mathbf{x}_\mathrm{a}^\mathrm{G} - \mathbf{x}_\mathrm{a}^\mathrm{S} \neq 0. \tag{3.12}$$

The simplest approach in testing these hypotheses is to use what's called *rectangular gating* [40], where the difference of each state in the two state vectors is compared to a constant threshold in order to determine if the two state estimates represent the same object. This approach, however, fails to take into consideration covariance terms of the state estimate's error covariance. Another simple approach is to compare the Euclidean distance for position, velocity and acceleration to a threshold. With this approach, the different degrees of uncertainty in the longitudinal and lateral directions is ignored.

Mahalanobis Distance Data Association

The most common method used in tracking applications is the statistical distance, or Mahalanobis distance, which is a generic statistical distance between two normally distributed random vectors [16, 18, 40, 163].

The squared statistical distance between two association state vectors from a global object O_j^G and a sensor object O_i^S is the squared difference of the two vectors, normalized by their error covariances and cross-covariance (note that for simplicity, the subscripts i and j for the specific object in the object list is omitted in the following):

$$d_\mathrm{G,S}^2 = \left(\hat{\mathbf{x}}_\mathrm{a}^\mathrm{G} - \hat{\mathbf{x}}_\mathrm{a}^\mathrm{S}\right)' \mathbf{S}^{\mathrm{G,S}^{-1}} \left(\hat{\mathbf{x}}_\mathrm{a}^\mathrm{G} - \hat{\mathbf{x}}_\mathrm{a}^\mathrm{S}\right) \tag{3.13}$$

where $\mathbf{S}^\mathrm{G,S}$ is the innovation covariance between the two state estimates:

$$\mathbf{S}^\mathrm{G,S} = \mathbf{P}^\mathrm{G} + \mathbf{P}^\mathrm{S} - \mathbf{P}^\mathrm{G,S} - \mathbf{P}^\mathrm{G,S}. \tag{3.14}$$

The squared statistical distance $d^2_{\mathrm{G,S}}$ is χ^2-distributed with $n_{\mathbf{x}_\mathrm{a}} = \dim(\mathbf{x}_\mathrm{a})$ degrees of freedom. Hypothesis H_0 is accepted if the squared statistical distance falls below a gate threshold, G:

$$d^2_{\mathrm{G,S}} \leq G \tag{3.15}$$

The gate threshold G is chosen such that the statistical test

$$P(d^2_{\mathrm{G,S}} > G \mid H_0) = \alpha \tag{3.16}$$

is fulfilled. In other words, G is chosen such that there is a probability of α that the squared statistical distance is greater than G under the condition that H_0 is true. Under this formulation, choosing a G too small will result in more misassociations. The gate threshold G is obtained from the inverse cumulative distribution function of the χ^2-distribution with $n_{\mathbf{x}_\mathrm{a}}$ degrees of freedom at $1-\alpha$:

$$G = F^{-1}_{\chi^2_{n_{\mathbf{x}_\mathrm{a}}}}(1-\alpha) \tag{3.17}$$

A typical value of α in tracking applications is 0.05. The statistical distance is a common approach used for object-to-object [42, 100, 126, 187, 263] and measurement-to-object [139, 182, 191, 194, 262] association in automotive perception applications.

Several methods exist for improving the robustness of object association using the statistical distance based on the object quality. In [40], the statistical distance is extended by adding a logarithmic term dependent on the innovation covariance

$$D^2_{\mathrm{G,S}} = d^2_{\mathrm{G,S}} + 2\ln\sqrt{|\mathbf{S}^{\mathrm{G,S}}|}. \tag{3.18}$$

This has the effect of penalizing an association between objects with large error covariances, such as when an object has just been initialized or the object is unreliable.

Extending the Mahalanobis Distance with Attribute Information

In this section, a novel approach is introduced which extends the statistical distance with attribute information, such that existence probability and classification information can be considered in the object association process.

The statistical distance can be extended using the objects' probability of existences, $p^{\mathrm{G}}(\exists\mathbf{x})$ and $p^{\mathrm{S}}(\exists\mathbf{x})$ (see Chapter 5). From the χ^2 inverse cumulative distribution function, a term based on the existence probabilities are added onto the statistical distance

$$d^2_{\mathrm{G,S}}(p(\exists\mathbf{x})) = F^{-1}_{\chi^2_{n_{\mathbf{x}_\mathrm{a}}}}(1-p^{\mathrm{G}}(\exists\mathbf{x})) + F^{-1}_{\chi^2_{n_{\mathbf{x}_\mathrm{a}}}}(1-p^{\mathrm{S}}(\exists\mathbf{x})) \tag{3.19}$$

which results in the extended statistical distance

$$D^2_{\mathrm{G,S}} = d^2_{\mathrm{G,S}} + 2\ln\sqrt{|\mathbf{S}^{\mathrm{G,S}}|} + d^2_{\mathrm{G,S}}(p(\exists\mathbf{x})). \tag{3.20}$$

The statistical distance is only slightly increased for existence probabilities close to 1, whereas for small existence probabilities, the statistical distance is significantly increased. This has the effect of reducing associations between objects with low existence.

In addition to the existence probability, the classification between the objects should also be considered in the association process. In [21], a sufficient statistics method is used parallel to the association of the kinematic state. Here, a novel approach, published in

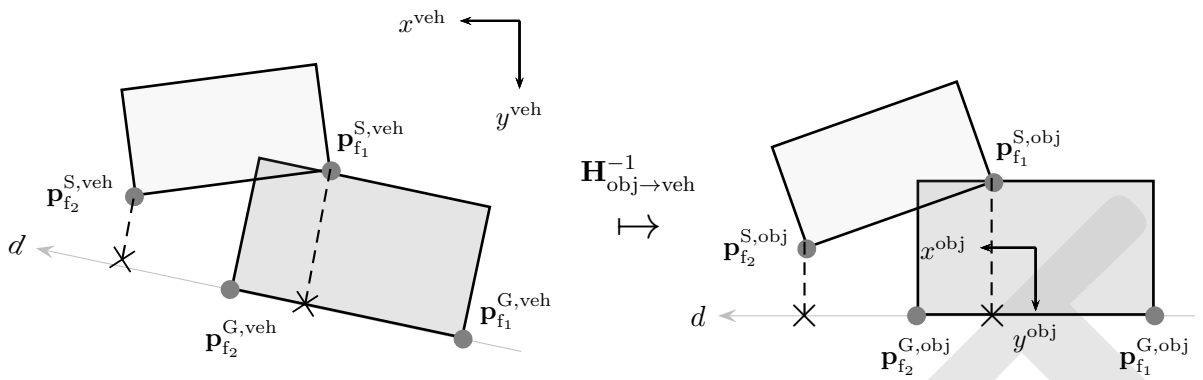


Figure 3.7: Geometrical association using the global object's length, where association occurs in the global object's coordinate system using the object to vehicle coordinate transformation..

[309], using the cosine similarity is presented here in order to determine the similarity between two classification vectors

$$\text{sim}(\mathbf{c}^{\text{G}}, \mathbf{c}^{\text{S}}) = \cos(\theta) = \frac{\mathbf{c}^{\text{G}} \cdot \mathbf{c}^{\text{S}}}{\|\mathbf{c}^{\text{G}}\| \|\mathbf{c}^{\text{S}}\|} \tag{3.21}$$

where $\mathbf{c}^{\text{G}}$ and $\mathbf{c}^{\text{S}}$ are the normalized classification vectors of the two objects. The evaluated function $\text{sim}(\mathbf{c}^{\text{G}}, \mathbf{c}^{\text{S}})$ results in a value between 0 to 1 due to the fact that the classification vectors can never have negative values. The similarity measure of the classification vectors is then also added to the statistical distance using a χ^2 inverse cumulative distribution function, such that the new and final extended statistical distance results in

$$D_{\text{G,S}}^2 = d_{\text{G,S}}^2 + 2\ln\sqrt{|\mathbf{S}^{\text{G,S}}|} + d_{\text{G,S}}^2(p(\exists\mathbf{x})) + F_{\chi^2_{n_{\mathbf{x}_\text{a}}}}^{-1}(1 - \text{sim}(\mathbf{c}^{\text{G}}, \mathbf{c}^{\text{S}})). \tag{3.22}$$

With this formulation, objects whose classification vectors differ will result in larger values added to the statistical distance. This helps in the association of objects which are spatially and dynamically close, but have a different classification. This can happen in highway scenarios when, for example, a motorcycle drives closely next to a car and is even more relevant when considering urban environments, where different types of objects coexist in a much denser environment.

3.3.4 Geometrical

The geometrical association step, as mentioned in the previous sections, is necessary when state vector association does not occur with the same feature from the two objects to be associated. In these cases, a single common dimension can be identified for which state vector association is performed, for example the length, if both features lie along the length of the object's rectangular model. The other dimension is then associated using a heuristic algorithm in order to determine if the two objects' rectangular models overlap with one another, which is called geometrical association in this thesis.

Consider the example in Figure 3.7: the global object in dark gray is longitudinally closer to the host vehicle than the sensor object in light gray. The sensor object seems to be an "extension" of the global object, where the global object has been tracked with feature $\mathbf{p}_{\text{f}_1}^{\text{G,veh}}$ (in this case equivalent to feature RL) and the sensor object is detected and

Algorithm 3.1 Line overlap test for geometrical association.

if $d(\mathbf{p}_{\mathrm{f}_1}^{\mathrm{S,veh}}) - \sigma_d^{\mathrm{S}} > d(\mathbf{p}_{\mathrm{f}_2}^{\mathrm{G,veh}}) + \sigma_d^{\mathrm{G}}$ **then**
    $g \leftarrow 0$
else if $d(\mathbf{p}_{\mathrm{f}_2}^{\mathrm{S,veh}}) + \sigma_d^{\mathrm{S}} < d(\mathbf{p}_{\mathrm{f}_1}^{\mathrm{G,veh}}) - \sigma_d^{\mathrm{G}}$ **then**
    $g \leftarrow 0$
else
    $g \leftarrow 1$
end if
return g

tracked with feature $\mathbf{p}_{\mathrm{f}_2}^{\mathrm{S,veh}}$ (equivalent to feature FL), resulting in the feature constellation depicted in Figure 3.6c where the features lie on the same side, in this case the length of the object. Such a situation occurs quite commonly with longer objects or with objects which are transitioning from one sensor's field-of-view to another's. The goal is to determine a Boolean g where $g = 1$ if the geometrical association determines a successful overlap of the objects' geometry.

Geometrical association begins with choosing the two features that lie on the common side, but have an uncommon dimension. The two features describing the length or width of the global object are denoted as $\mathbf{p}_{\mathrm{f}_1}^{\mathrm{G,veh}}$ and $\mathbf{p}_{\mathrm{f}_2}^{\mathrm{G,veh}}$. A 1-dimensional coordinate system d is defined along the side of the global object, on which geometrical association is calculated. In order to simplify the geometrical association calculation, the relevant features of the sensor objects, $\mathbf{p}_{\mathrm{f}_1}^{\mathrm{S,veh}}$ and $\mathbf{p}_{\mathrm{f}_2}^{\mathrm{S,veh}}$, are transformed into the global object's coordinate system with

$$\mathbf{p}_{\mathrm{f}_n}^{\mathrm{S,obj}} = \mathbf{H}_{\mathrm{obj}\to\mathrm{veh}}^{-1} \mathbf{p}_{\mathrm{f}_n}^{\mathrm{S,veh}} \tag{3.23}$$

where $\mathbf{H}_{\mathrm{obj}\to\mathrm{veh}}^{-1}$ is the inverse object to vehicle coordinate transformation matrix for the global object. The features for the global object are directly calculated as per Table 3.1. The result of such a transformation from the host vehicle coordinate system into the global object's coordinate system is depicted in Figure 3.7.

Given the global and sensor object's features in the global object's coordinate system allows for a simple projection of the features coordinates onto the 1-dimensional coordinate system d, where the projection is simply the the x or y coordinate, depending if d is parallel to the length or width, respectively, of the global object. The projected feature points onto the line d are denoted as $d(\mathbf{p}_{\mathrm{f}_n}^{\mathrm{G,obj}})$ for the global object and $d(\mathbf{p}_{\mathrm{f}_n}^{\mathrm{S,obj}})$ for the sensor object.

In addition to the feature's location on d, the uncertainty of the objects' dimension in form of the standard deviation σ_d is considered, where $d \in \{w, l\}$ refers to the standard deviation of the width or the length, respectively. Using the location of the features along the line d and the uncertainty of the objects' dimension, a line overlap test according to Algorithm 3.1 is performed in order to determine if geometrical association is successful. The line overlap test algorithm checks if there is an overlap between the projection of the outermost and innermost features of the objects' on the line along the relevant side of the global object. If this is the case, the algorithm returns $g = 1$ for successful association and $g = 0$ if unsuccessful.

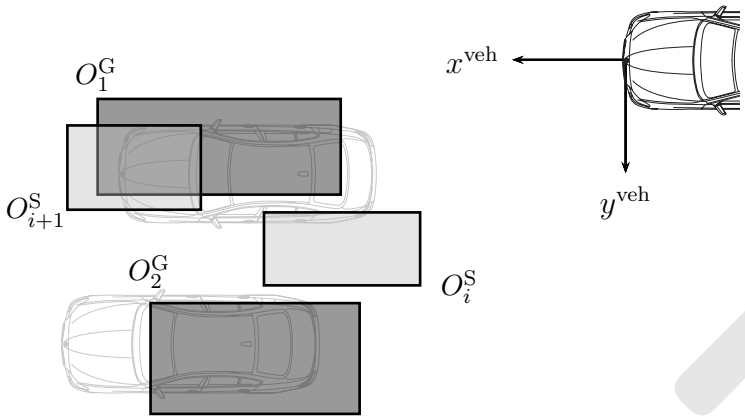


Figure 3.8: Multi-object association problem: which global object should be associated with the sensor-level object O_i^{S}?

3.3.5 Association Validation

The results of both state vector and geometrical association are validated, where both association methods must produce a positive result, before being entered into the association matrix. The result of state vector association, $D_{i,j}^2(k)$, is validated against a threshold G such that a positive association results when the following condition, as in (3.15), is met:

$$D_{i,j}^2(k) \leq G \tag{3.24}$$

where G is chosen such that

$$G = F_{\chi^2_{n_{\mathbf{x}_\mathrm{a}}}}^{-1}(1-\alpha). \tag{3.25}$$

An α value of 0.05 is typically used.

Geometrical association is simply validated by checking the Boolean value g, where $g = 1$ is a positive association and $g = 0$ is a negative association.

If both state vector association and geometrical association succeed, the value $D_{i,j}^2(k)$ obtained during state vector association is entered into the association matrix:

$$\mathbf{A}_{i,j} = D_{i,j}^2(k). \tag{3.26}$$

3.3.6 Multi-Object Association

In automotive environment perception applications, many objects, or targets, are detected by a single sensor at any given time, leading to the problem of deciding which sensor-level objects should be assigned to which fusion-level global objects. Consider the situation shown in Figure 3.8. Here it is unclear whether the sensor-level object, O_i^{S}, should be associated with global object O_1^{G} or O_2^{G}. Simply taking the association with the minimal statistical distance may not suffice if other sensor-level objects exist which could also be associated and assigned. Additionally, the rear part of one of the vehicles is observed with the sensor-level object O_{i+1}^{S}, which raises the question, should the global object O_1^{G} be assigned to the sensor-level object O_i^{S} or $O_{i+1}^{\mathrm{S}i}$ or only O_{i+1}^{S}? In this section, an algorithm for solving this association problem is presented.

An object association matrix, denoted as $\mathbf{A}$, is usually generated for solving the object assignment problem [40]. The columns of the matrix represent the N global fusion-level objects and the rows of the matrix are the M sensor-level objects, where the matrix is filled with the statistical distance between the pairs of objects represented by that specific row and column:

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,j} & \cdots & a_{1,N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{i,1} & a_{i,2} & \cdots & a_{i,j} & \cdots & a_{i,N} \\ a_{i+1,1} & a_{i+1,2} & \cdots & a_{i+1,j} & \cdots & a_{i+1,N} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{M,1} & a_{M,2} & \cdots & a_{M,j} & \cdots & a_{M,N} \end{bmatrix}. \tag{3.27}$$

where

$$a_{i,j} = D^2_{i,j}(k). \tag{3.28}$$

The gating procedures described in the previous section is used to generate a sparsely filled association matrix, where two pairs of objects which did not pass the gating test are given an empty value in the association matrix. Following the example in Figure 3.8, this could end up looking something similar to

$$\mathbf{A} = \begin{bmatrix} - & - & \cdots & - \\ \vdots & \vdots & \vdots & \vdots \\ a_{i,1} & a_{i,2} & \cdots & - \\ a_{i+1,1} & - & \cdots & - \\ \vdots & \vdots & \ddots & \vdots \\ - & - & \cdots & - \end{bmatrix}. \tag{3.29}$$

In this example the sensor-level object $O_i^{\mathrm{S}_i}$ falls within the gate of global objects O_1^{G} and O_2^{G}, and all other association pairs did not pass the gating test. The assignment problem now needs to decide which global objects, O_1^{G} or O_2^{G}, really belong to which sensor-level object.

The most common family of algorithms for solving the assignment problem are nearest neighbor algorithms, which will be used here. These algorithms attempt to find the best one-to-one assignment between the rows and columns of an association matrix. A good overview of different types of association algorithms, including the more complex probabilistic variations, can be found in [40]. The auction algorithm [37, 40] will be described in this section as the algorithm of choice for solving the association problem.

The auction algorithm begins with creating a cost matrix $\mathbf{C}^a$ out of the association matrix $\mathbf{A}$, where the cost of an element is

$$c_{i,j} = 2G - a_{i,j}. \tag{3.30}$$

This cost value represents the "worth" of a specific element in the association matrix with respect to the gating threshold G, such that statistical distances closer to the threshold equal the threshold itself. Elements in $\mathbf{A}$ which did not pass the gating threshold are assigned the value 0. Additionally, a second cost matrix $\mathbf{C}^b$ with dimensions $M \times M$ is defined such that

$$\mathbf{C}^b = G\,\mathbf{I} \tag{3.31}$$

where this matrix represents the fact that if no assignment of a sensor-level object can be made to a global object, then the sensor-level object is assigned to itself, i.e. should be used to initiate a new global object. The complete cost matrix $\mathbf{C}$ is a concatenation of $\mathbf{C}^a$ and $\mathbf{C}^b$, resulting in

$$\begin{aligned}\mathbf{C} &= \left[\begin{array}{c|c} \mathbf{C}^a & \mathbf{C}^b \end{array}\right] \\ &= \left[\begin{array}{cccccc|cccccc} c_{1,1} & c_{1,2} & \cdots & c_{1,j} & \cdots & c_{1,N} & G & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ c_{i,1} & c_{i,2} & \cdots & c_{i,j} & \cdots & c_{i,N} & 0 & \cdots & G & 0 & \cdots & 0 \\ c_{i+1,1} & c_{i+1,2} & \cdots & c_{i+1,j} & \cdots & c_{i+1,N} & 0 & \cdots & 0 & G & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{M,1} & c_{M,2} & \cdots & c_{M,j} & \cdots & c_{M,N} & 0 & \cdots & 0 & 0 & \cdots & G \end{array}\right] \end{aligned} \tag{3.32}$$

where the dimensions of the matrix are now $\dim(\mathbf{C}) = \{M, N+M\}$. Two additional data structures are needed for the auction algorithm. The first is a vector $\mathbf{p}$ with $\dim(\mathbf{p}) = N+M$ which keeps tracks of the current bid price for an object in a given column, where p_i refers to an element in $\mathbf{p}$ and p_{i_j} is the bid price in $\mathbf{p}$ due to the fact that the sensor-level object i was assigned to global object j. The second is a vector $\mathbf{r}$ with $\dim(\mathbf{r}) = M$ which contains for each element r_j an assignment of a sensor-level object to a global object, including the additional global objects in the extended cost matrix $\mathbf{C}$ that represent a self-assignment. The vector $\mathbf{p}$ is initialized with a zero vector and $\mathbf{r}$ is initialized with an invalid value to signify that no assignments have been made.

In order to find the best 1-to-1 assignment between sensor-level objects and global objects, the auction algorithm is carried out iteratively with the following steps:

1. Choose the first element r_j in $\mathbf{r}$ with an invalid value for which no assignment has yet been made.

2. For the sensor-level object j find an assignment to a global object i with the maximum benefit, where the benefit is calculated using

$$a_{i,j} - p_{i_j} = \max_i \{a_{i,j} - p_i\} \tag{3.33}$$

3. Assign the sensor-level object j to global object i_j such that $r_j = i$. If global object i was previously assigned to a different element in $\mathbf{r}$, then remove this assignment.

4. Calculate the new bid price for the assignment of i to j using

$$p_{i_j} = p_{i_j} + y_j + \epsilon \tag{3.34}$$

   where y_j is the difference between the best and the second best benefit from step 2 and ϵ is a tolerance value for the deviation of the algorithm to the optimal solution.

5. Return to step 1 until an assignment has been made for all elements in $\mathbf{r}$.

The output of the auction algorithm is the assignment vector $\mathbf{r}$ containing an assignment for each element r_j to a global object i. This assignment vector is then used to perform the fusion of the objects' state and covariance, existence and classification, as will be described in the following chapters.

Discussion

The process of how objects from sensors are fused together at the fusion-level and the association of sensor-level objects to global fusion-level objects is the critical first step in a high-level fusion architecture. Fusion is best performed in a common coordinate system, usually a vehicle-relative coordinate system. The fusion strategy, whether sensor-to-sensor or sensor-to-global, determines if fusion occurs synchronously at pre-defined fusion intervals or asynchronously as sensor data arrives at the fusion module. In this thesis, a sensor-to-global fusion strategy is chosen due to the fact that the fusion-level global object list is kept as up-to-date as possible, which is critical for certain driver assistance systems which need to react quickly to a changing environment in order to react properly to critical situations.

Object association is achieved using a combination of the traditional track-to-track association algorithms using a statistical distance and a geometrical association step in order to extend the traditional algorithms to the rectangular object model. Care must be taken in choose the proper feature for state vector association, as the wrong feature may lead to undesired association results. A new approach that extends the statistical distance with the existence probability and classification vectors was presented, which further improves association in complex environments. It is assumed that a 1-to-1 assignment can be made between sensor-level objects and global objects. This assumption, however, may not in all cases hold, as some sensors may detect more than one object for a single object in reality. In essence, this is an error of the tracking algorithms at the sensor-level, but could potentially be compensated for at the fusion-level. For sensor-level tracking, probabilistic association algorithms such as Probabilistic Data Association (PDA) or Joint Probabilistic Data Association (JPDA) [18, 40] or one of their integrated variations Integrated Probabilistic Data Association (IPDA) or Joint Integrated Probabilistic Data Association (JIPDA) are sometimes used in automotive applications [170, 182, 191, 194], especially for sensors which generate a lot of clutter, such as a radar, or in feature-level fusion algorithms. The algorithms presented in this thesis could be extended to also consider such probabilistic association methods, where the merging and splitting of objects can be directly modeled. It was found, however, that simple heuristic methods for merging larger objects, such as trucks, work well in practice. Despite this weakness in the presented association algorithms, a 1-to-1 assignment of objects works quite well in practice for highway scenarios, but may need to be reevaluated when considering high-level fusion in urban environments.

The following chapters will present algorithms for the fusion of two objects which were assigned to one another, as presented in this chapter. Fusion takes place in three steps, where the state and covariance, existence probability and classification of an object are each fused separately using different algorithms.

4 State and Covariance

The object model consists of a state estimate, $\hat{\mathbf{x}}$ and its error covariance $\mathbf{P}$, which contain the position, velocity, acceleration, orientation, length and width of the detected object. Together, $\hat{\mathbf{x}}$ and $\mathbf{P}$ is called a *track*. The error covariance matrix is composed of variances and covariance which represent the error of the state estimate. In a multi-sensor, multi-target tracking application, the problem of how to track all of the objects, taking into consideration all of the sensor data, is non-trivial. Many textbooks have covered this topic extensively with a strong aerospace application [16, 25, 40]. In this chapter, algorithms for track estimation within the proposed high-level sensor data fusion architecture at the sensor and fusion-level are described.

4.1 Sensor-Level Processing with Tracking Algorithms

In a high-level sensor data fusion architecture, each sensor must produce an object list, $\mathcal{O}^{\mathrm{S}}(t)$, part of which is the state estimate and its error covariance. This section will briefly describe multi-target tracking algorithms which are typically used in automotive environment perception applications at the sensor-level to detect other relevant objects, such as vehicles, bicycles, pedestrians, etc. Since tracking algorithms have been extensively studied in the literature, only a short overview is given, where the details will be left to the cited works.

Tracking at the sensor-level in an automotive environment is usually accomplished in the follow steps: sensor measurement, feature extraction, data association, filtering and track management. The process of generating an object list at the sensor-level is shown in Figure 4.1. A sensor makes a measurement of the environment using its hardware. This raw sensor data is then pre-processed using feature extraction modules, which extract features of relevant objects from the raw sensor data. The feature extraction step is different for every sensor and its performance is crucial for reliable object detection at the sensor-level. The extracted features are used as the input to the tracking and filtering algorithms in order to produce tracked objects over time. After a filter update, track management creates new tracks or deletes old ones depending on the detection and association results. Details to these different steps will be described in the following sections.

Such tracking algorithms have been extensively used in automotive applications for driver assistance systems. Most tracking systems in automotive applications have focused on the detection and tracking of objects in the frontal area of the vehicle using laser scanners [93, 107, 108, 157, 162, 216, 255], radar [183] and mono or stereo cameras [120, 121]. Such tracking systems have also been used to realize low-level or feature-level sensor data fusion architectures, where sensor data from many sensors are fed into a single, centralized tracking algorithm [78, 139, 168, 211].

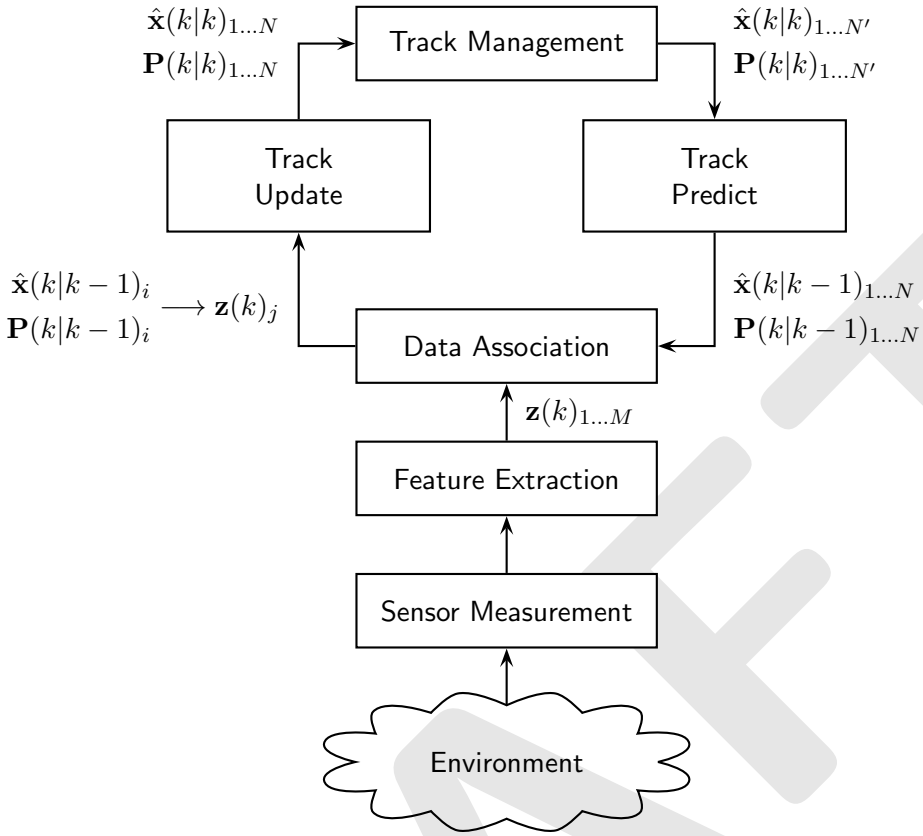


Figure 4.1: Typical multi-target tracking processing flow at the sensor-level for producing a state estimate and its error covariance with sensor measurements.

4.1.1 Feature Extraction

Unlike in aerospace applications, objects cannot be modeled as simple point-targets in automotive environment perception. Objects such as vehicles and trucks are relatively large when in range of a vehicle's sensors. Therefore, the dimensions of an object play a vital role in accurate and reliable tracking algorithms. Using the object model and its feature description, as described in Section 2.2.1, sensors must extract the best observable feature, or sometimes called a reference point, which is then used as the point in space with which the tracking algorithms estimate the object's state and error covariance.

With laser scanner sensors, for example, the laser range image, or point cloud, are clustered into groups using segmentation or clustering algorithms to create object hypotheses, or measurements, for the tracking algorithms [78, 93, 125, 148, 157, 178, 185]. Similar clustering methods are used with radar [78, 146] and stereo camera [158] sensors. Some approaches use an intermediate environment representation before segmentation and clustering, such as occupancy grids [11, 67] or a simplified polar representation such as stixels [217], as opposed to working directly with the sensor's data. After clustering, line or shape extraction algorithms can be used in order to extract the general shape and orientation of the object [11, 53, 178, 180]. From this information, possible observed features, as defined by the object model, are extracted and handed over to a tracking algorithm, as shown in Figure 4.1.

Such feature-based or model-based tracking algorithms are common in automotive ap-

plications and have been extensively studied [31, 77, 78, 140, 216, 239, 289, 306]. For reliable and accurate tracking of objects in the automotive environment, such a model-based approach is necessary, since the dimensions of an object are large compared to the range of an object from a sensor. More recently, occupancy grid methods have been used in order to accumulate the shape of an object using a local object grid map such that feature extraction and shape estimation can be improved [11, 12, 240, 241]. However, such feature and model-based approaches complicates object detection for sensors and, therefore, many different approaches to solve this problem exist.

4.1.2 Data Association

In multi-target tracking, the problem of deciding which measurements belong to which tracks is called *data association*. After data association, filtering algorithms, such as the Kalman filter, can update their tracks with the assigned measurements. Since sensors do not generate perfect measurements of all of the real objects in the environment without false detections, the problem of data association is non-trivial and has been widely studied in the literature [16, 40]. Data association between objects at the fusion-level in a high-level sensor data fusion architecture is covered in Chapter 3.

The most basic method of data association attempts to find the best 1-to-1 assignment between the new measurements and existing tracks. For each track, gating methods are used to find possible measurements for a track. The simplest method is the nearest neighbor approach, which simply assigns the best measurement within a gate to a track. In some cases, a measurement may fall within the gate of more than one track. In order to find the best global solution of the assignment problem, a global nearest neighbor algorithm can be used [16, 40], where the global assignment problem can be solved using the auction algorithm [188]. These basic association techniques have their limits when sensors produce a significant amount of clutter measurements or noise. To solve this problem, probabilistic methods for data association have been developed. The Probabilistic Data Association (PDA) algorithm allows a track to be updated using a weighted sum of all of the measurements within the track's gate [16, 23, 26, 40]. The problem of assigning a single measurement to more than one track was solved using Joint Probabilistic Data Association (JPDA) [16, 40, 64, 65]. Simultaneous estimation of state and existence lead to the development of the Integrated Probabilistic Data Association (IPDA) and JPDA algorithms [197, 199], where the works from Mählisch and Munz have extensively applied these algorithms in automotive environments [170, 191, 194]. The previously described PDA methods are all based on the fact that a single target only produces a single measurement. The generalized probabilistic data association (GPDA) algorithm was presented in [237] and also applied in [1] and solves this problem by loosing this restriction and allows for multiple measurements to originate from a single target, which is common in automotive applications.

4.1.3 Filtering

Tracking on object over time is accomplished using a filter which is able to estimate the state of an object using noisy measurement data. The most popular method for filtering in tracking systems is the Kalman filter [142], which sequentially estimates an object's state over time with the assumption that the state and the input measurement can be

modeled using a normal distribution. Shortly after its introduction in 1960, the Kalman filter quickly found an application in estimating trajectories to the moon and the relative position and velocity between spacecraft in the Apollo space program [176]. Since then, it has become one of the most widely used and studied algorithms for state estimation and tracking.

Since the Kalman filter is such an important statistical estimation tool, the filter and its equations will be shortly described here. For further details on the Kalman filter, many sources are available [25, 244, 290, 305]. As shown in Figure 4.1, filtering is accomplished in a two-step process: predict and update. During state prediction, the previous state at $k-1$ is predicted forward to the time of the newly arrived measurement data, k, using an appropriate model (see Section 4.1.5 for an overview of kinematic models). The Kalman filter prediction equations are

$$\hat{\mathbf{x}}(k|k-1) = \mathbf{F}(k,k-1)\hat{\mathbf{x}}(k-1|k-1) + \mathbf{B}(k)\mathbf{u}(k) \tag{4.1}$$

$$\mathbf{P}(k|k-1) = \mathbf{F}(k,k-1)\mathbf{P}(k-1|k-1)\mathbf{F}(k,k-1)' + \mathbf{Q}(k) \tag{4.2}$$

where $\hat{\mathbf{x}}(k-1|k-1)$, $\mathbf{P}(k-1|k-1)$ is the previous state estimation and error covariance, $\mathbf{F}(k,k-1)$ is the state transition matrix, $\mathbf{u}(k)$ is the control vector, $\mathbf{B}(k)$ is the control transformation matrix, $\mathbf{Q}(k)$ is the process noise covariance and $\hat{\mathbf{x}}(k|k-1)$, $\mathbf{P}(k|k-1)$ is the predicted state estimation and error covariance. In automotive applications, the host vehicle's motion from $k-1$ to k can be modeled as the control in order to estimate the absolute velocity of detected objects. The process noise matrix, $\mathbf{Q}(k)$, models the uncertainty in the state transition model used. After the prediction step, the Kalman filter algorithm updates the state and covariance with new measurement data from time k:

$$\mathbf{S}(k) = \mathbf{H}(k)\mathbf{P}(k|k-1)\mathbf{H}(k)' + \mathbf{R}(k) \tag{4.3}$$

$$\mathbf{K}(k) = \mathbf{P}(k|k-1)\mathbf{H}(k)'\mathbf{S}(k)^{-1} \tag{4.4}$$

$$\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}(k|k-1) + \mathbf{K}(k)\left[\mathbf{z}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k|k-1)\right] \tag{4.5}$$

$$\mathbf{P}(k|k) = \left[\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)\right]\mathbf{P}(k|k-1) \tag{4.6}$$

where $\mathbf{z}(k)$ is the new measurement, $\mathbf{H}(k)$ is the measurement-to-state space transformation matrix, $\mathbf{S}(k)$ is the innovation covariance, $\mathbf{R}(k)$ is the measurement covariance, $\mathbf{K}(k)$ is the Kalman gain, $\mathbf{I}$ is the identity matrix and $\hat{\mathbf{x}}(k|k)$, $\mathbf{P}(k|k)$ is the updated state and covariance. The state and covariance is updated in a weighted fashion by means of the Kalman gain, which weighs the predicted covariance with the innovation covariance.

The Kalman filter algorithm as presented above makes various assumptions. First, the uncertainties represented by the error covariance, measurement covariance and process noise covariance are assumed to be normally distributed. Second, the state transition and measurement models are assumed to be linear. Variations of the Kalman filter exist that loosen these assumptions. The extended Kalman filter is able to estimate a non-linear state transition and measurement models by linearizing a non-linear function at a given point [25, 244, 290, 305]. Another solution to solving the state estimation problem for non-linear models is to make use of the unscented transformation, resulting in the unscented Kalman filter [134, 135]. The idea behind the unscented Kalman filter is to propagate so-called extracted sigma-points from a normally distributed random variable through a non-linear function and then reconstructing the resulting normal distribution. If the state

or measurements cannot be accurately modeled using a normal distribution, Monte Carlo methods of state estimation exist, such as the particle filter, which are able to estimate states of any arbitrary distribution, but with additional computational costs [244, 266].

4.1.4 Track Management

After every filter update, the creation of new tracks and the deletion of old ones is referred to as track managements. Basic track management techniques are described in [40], where a track score, depending on whether a track received an association or not, can be used to delete tracks. New tracks are usually created from unassociated measurements and later verified using the track score upon further measurement associations. These simple heuristics work well in most tracking applications.

Track management is simplified when using IPDA or Joint Integrated Probabilistic Data Association (JIPDA) algorithms, where track existence is also estimated. With these algorithms, track management is achieved by thresholding a track's existence probability in order to delete or confirm tracks. Track management using an existence probability will also be used in thesis at the fusion-level and is described in further detail in Chapter 5.

Another problem in track management is how to model uncertain associations and possible merging and splitting of tracks. The Multiple Hypothesis Tracking (MHT) filter attempts to solve these problems by modeling multiple hypothesis for tracked objects when association conflicts exist, with the hope that subsequent data will resolve any uncertainties [40, 41, 226]. The most likely hypothesis for each track is then the generated output for a particular sensor. Application of the MHT algorithm for automotive environments has been demonstrated in [154, 155, 262].

More advanced algorithms have been developed in order to model the number of targets in the environment, including their creation and deletion, together with the traditional state estimation. One such family of algorithms is based on random set theory and Finite Set Statistics (FISST), introduced by Mahler [164], where all aspects of multi-object tracking are unified in a single, consistent framework. An algorithm which builds upon this framework is the Probability Hypothesis Density (PHD) filter, also introduced by Mahler [165] and further extended in [283]. FISST based tracking methods, including the PHD filter, have been successfully applied to tracking applications in automotive environments [154–156, 166, 177].

4.1.5 Kinematic Models

The prediction of dynamic objects in tracking algorithms depends on a kinematic model. The choice of kinematic model directly effects the accuracy and reliability of tracking performance, as an inaccurate model will lead to misassociations and therefore poor performance in state estimation (especially the dynamic states such as velocity and acceleration) and reliable track management. It is therefore vital to make the correct assumptions about the movement of objects in the desired environment. An overview of most of the kinematic models for tracking applications can be found in [238] and the popular tracking works of [18, 25, 40].

The most common and simplest model is that of constant velocity or constant acceleration model. In this model, a kinematic state, either the velocity or acceleration is

assumed to be constant as the object moves forward in time. Such a model usually decouples state estimation in a given coordinate system, where, for example, longitudinal state estimation and lateral state estimation are independent of one another. Such models work well if properly parametrized or if the time-intervals between measurement updates are sufficiently small. In automotive applications, particularly with the state estimation of vehicles, a constant turn rate model is more appropriate, as it more closely approximates the dynamics of a vehicle. In a constant turn rate model, it is assumed that the turn rate of the vehicle is constant and the vehicle moves in space about a circle whose radius is proportional to the turn rate. This leads to a state estimation where the longitudinal and lateral directions are coupled through this rotation, such that the velocity and acceleration must be estimated along the axis of the vehicle and not in the tracking reference frame.

A typical problem in automotive environments is that different types of dynamics objects are to be detected, such as vehicles, pedestrians, animals, etc., where a single kinematic model does not properly describe the motion of all types of objects. Tracking performance can in this case be improved by using an Interacting Multiple Model (IMM) filter, introduced in [44]. The IMM filter estimates several kinematic models for a single track simultaneously and combines the results probabilistic, such that the result of the filter with the most probable kinematic model is applied to the final state estimation of the track. An example of successful application of the IMM filter in automotive environments can be found in [141]. Many approaches also combine the IMM approach with other tracking algorithms, as done in [45], where the IMM filter is combined with the JPDA algorithm. Similarly, the MHT approach can also be combined with IMM, as demonstrated in [41].

4.2 Correlation and Sequence of Sensor Data

In a high-level sensor data fusion architecture, the data to be fused is already-filtered object data. If two sensors observe the same object at the sensor-level using the algorithms described in the previous section, then the object data from each of the respective sensors may be correlated. This fact was first considered by Bar-Shalom in [15] and is further discussed in his book [16].

Consider the fact that two sensors i and j observe the same object and each produce object data at time k in the form of a state vector, $\hat{\mathbf{x}}^i(k|k)$ and $\hat{\mathbf{x}}^j(k|k)$, and covariance matrix, $\mathbf{P}^i(k|k)$ and $\mathbf{P}^j(k|k)$. If the object data from these two sensors is correlated, then the cross-covariance matrix, $\mathbf{P}^{ij}(k|k)$, does not equal zero:

$$\begin{aligned}\mathbf{P}^{ij}(k|k) &\triangleq \mathrm{E}\left[\left(\mathbf{x}(k)-\hat{\mathbf{x}}^i(k|k)\right)\left(\mathbf{x}(k)-\hat{\mathbf{x}}^j(k|k)\right)'\right] \\ &\neq 0\end{aligned} \tag{4.7}$$

The source of correlation depends on how object data is fused at the fusion-level. It may either be due to the common process noise or due to the common information history in the object data.

In addition to the correlation problem, sensor data may also arrive out-of-sequence at the fusion-level, which leads to the problem of how to properly temporally fuse all of the sensor data, while at the same time integrating all of the data as it arrives at the fusion module.

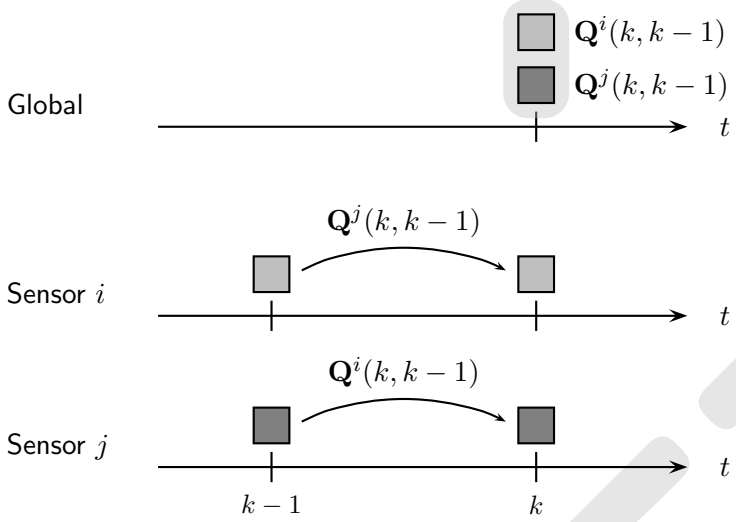


Figure 4.2: The process noise from sensors i and j, $\mathbf{Q}^i(k,k-1)$ and $\mathbf{Q}^j(k,k-1)$, are both in the fused global object.

4.2.1 Process Noise

One of the sources of correlation between object data is due to the common process noise. Filtered object data usually follows a kinematic model, such as the ones described in Section 4.1.5. In order to model the uncertainty in an ideal kinematic model, a process noise is added onto the kinematic states of the object. When object data from two sensors have observed the same object, the process noise was added separately by each sensor. During the fusion of these two sensors, the process noise in the object data exists twice, whereas for a single observed object it should only be added once. Figure 4.2 shows how the process noise from sensor i and sensor j from time $k-1$ to k, $\mathbf{Q}^i(k,k-1)$ and $\mathbf{Q}^j(k,k-1)$, are both included into the fused global object, even though the process noise should only be included once for every object.

In order to compensate for the common process noise during fusion, the cross-covariance matrix between two sensors must be calculated. The following shows two methods, one optimal and one approximate, for calculating the cross-covariance matrix.

Cross-Covariance Recursion

An optimal method for calculating the cross-covariance matrix due to the common process noise between two sensors exists and was first presented by Bar-Shalom in [15] and discussed further in [16]. The optimal method is a recursion, which uses the previously calculated cross-covariance matrix and sensor-level filtering parameters:

$$\mathbf{P}^{ij}(k|k) = \left[\mathbf{I} - \mathbf{K}^i(k)\mathbf{H}^i(k)\right]\left[\mathbf{F}(k,k-1)\mathbf{P}^{ij}(k-1|k-1)\mathbf{F}(k,k-1)' + \right. \\ \left. + \mathbf{Q}(k,k-1)\right]\left[\mathbf{I} - \mathbf{K}^j(k)\mathbf{H}^j(k)\right]' \quad (4.8)$$

where $\mathbf{P}^{ij}(k-1|k-1)$ is the previously calculated cross-covariance, $\mathbf{K}^i(k)$ is the Kalman gain from time k for sensor i, $\mathbf{Q}(k,k-1)$ is the discretized continuous-time process noise from time $k-1$ to k, $\mathbf{F}(k,k-1)$ is the state transition matrix from time $k-1$ to k and $\mathbf{H}^i(k)$ is the state-to-measurement transformation matrix for sensor i.

Under the assumptions made in the derivation, the cross-covariance recursion method is optimal. However, the algorithm requires sensor-level parameters from each of the sensors, such as the Kalman gain. In a high-level sensor data fusion architecture, such internal filter information is usually not available at the fusion-level. Therefore, this method may be quite impractical in realistic applications and will not be further considered in this thesis.

Cross-Covariance Estimation

In [16], a method to estimate the cross-covariance matrix between two sensors was presented. As opposed to the recursion method presented in the previous section, the estimation method does not require the internal filter parameters, such as the Kalman gain. The cross-covariance estimation is calculated directly from the covariance matrices of sensors i and j:

$$\mathbf{P}^{ij}(k|k) \approx \rho\sqrt{\mathbf{P}^{i}(k|k) \bullet \mathbf{P}^{j}(k|k)} \tag{4.9}$$

where ρ is a correlation weighting factor, which can be seen as a tuning parameter, and $\bullet$ is the element-wise multiplication operator for the two matrices. For tracking applications, it was found that $\rho = 0.4$ provide good results in filter consistency [16]. Though not optimal, the cross-covariance estimation method provides an approximation of the cross-covariance when internal filter parameters are not available at the fusion-level.

4.2.2 Common Information History

In a sensor-to-global fusion strategy, as described in Section 3.2.1, another source of correlation can occur: correlation due to common information history. The problem is that global objects are maintained over time, which leads to the fact that the full state estimate from a sensor is fused into the global objects at each fusion time.

The problem of correlation due to the common information history is best illustrated using an example. In Figure 4.3, the state estimate from sensor i is fused into the global object twice, first at time $k_i - 1$ and then again at time k_i. At time $k_i - 1$, the state estimate for sensor i contains the measurement data, or information, $\{Z^i\}_0^{k_i-1}$, which includes the set of all measurements for sensor i from time 0 to $k_i - 1$. The information for this sensor up to this time is then fused into the global object, represented by the dashed lines in Figure 4.3. At this point in time, the global object contains the information from sensor i up to time $k_i - 1$:

$$\{Z^{\mathrm{G}}\} = \{Z^i\}_0^{k_i-1} \tag{4.10}$$

At time k_i, the state estimate from sensor i is updated with a new measurement, such that the new state estimate contains the information $\{Z^i\}_0^{k_i}$, represented by the solid white box. This state estimate with the new measurement is now also fused into the global object. Visually, it can be seen from Figure 4.3 that the information from sensor i up to time $k_i - 1$ was fused twice, represented by the dashed lines showing up twice. Mathematically, the sum of the set of measurements also show that this information was fused twice, therefore

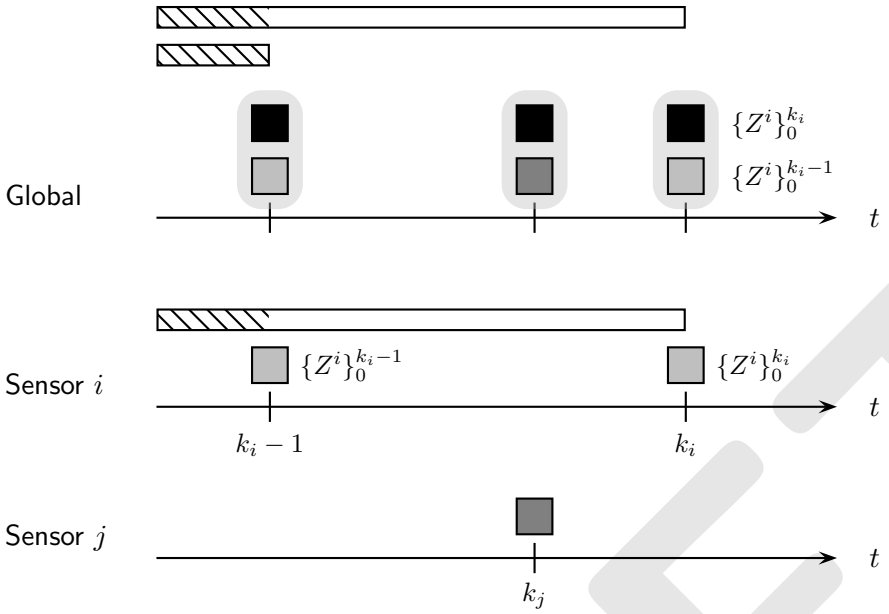


Figure 4.3: Correlation of sensor data due to a common history of previously fused sensor data, where the data from sensor i from time 0 to $k_i - 1$, $\{Z^i\}_0^{k_i-1}$ and depicted by the boxes with the dashed lines, is fused twice into the global object at the fusion-level.

leading to a correlation of measurement data in the global object:

$$\begin{aligned}\{Z^{\mathrm{G}}\} &= \{Z^i\}_0^{k_i} + \{Z^i\}_0^{k_i-1} + \{Z^j\}_0^{k_j} \\ &= \{Z^i\}_{k_i-1}^{k_i} + \{Z^i\}_0^{k_i-1} + \{Z^i\}_0^{k_i-1} + \{Z^j\}_0^{k_j} \\ &= \{Z^i\}_{k_i-1}^{k_i} + 2\{Z^i\}_0^{k_i-1} + \{Z^j\}_0^{k_j}\end{aligned} \tag{4.11}$$

It is obvious that as time goes on, more measurement data will arrive from sensor i and the state estimate with new and old measurement data will be fused into the global object. The more often the sensor-level state estimate gets fused into the global object, the more severe the effect of the correlation due to common information history. Solving this problem is one of the main challenges in track-to-track fusion using a sensor-to-global fusion strategy. In the following sections, it will be shown that some track-to-track fusion algorithms, such as the information matrix fusion algorithm, are able to fuse tracks while compensating for the common information history correlation.

4.2.3 Out-of-Sequence Data

Another problem in an asynchronous and multi-sensor configuration is the sequence of the sensor data at the fusion-level. Each sensor may have a different measurement period and on top that may also have a latency of the measurement data due to internal sensor processing time and communication delays. This could lead to the problem of out-of-sequence sensor data at the fusion-level, which is a significant problem in a sensor-to-global fusion strategy.

The problem of out-of-sequence sensor data is illustrated in Figure 4.4. In the configuration shown, sensor i has a different measurement period and is asynchronous to sensor j. Additionally, sensor i also has a much shorter processing time and communication delay,

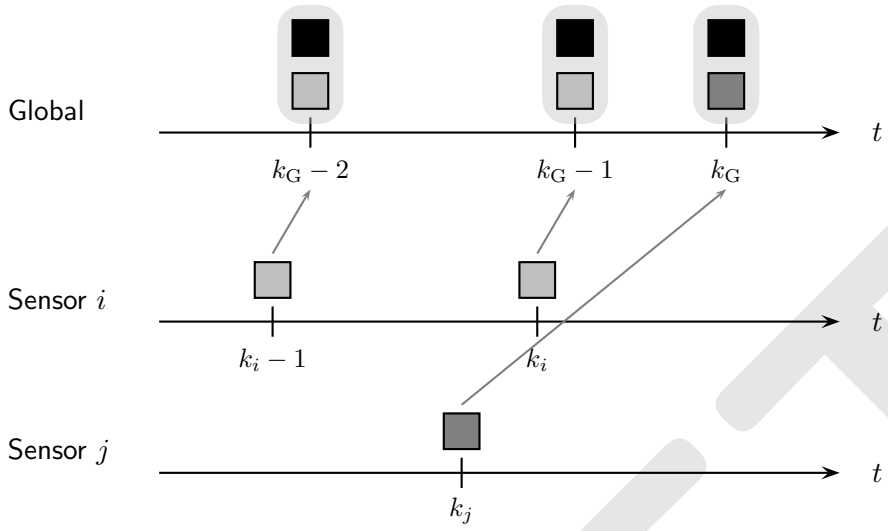


Figure 4.4: Problem of out-of-sequence sensor data at the fusion-level. Sensor j has a much longer processing time and communication delay, resulting that its sensor data arrives out-of-sequence at the fusion level with respect to the data from sensor i.

depicted by the gray arrows to the global fusion-level, than sensor j. The result is that data from sensor i arrive at the fusion-level at time $k_G - 2$ and $k_G - 1$ and data from sensor j arrives at time k_G. If a sensor has a large latency, such as sensor j in Figure 4.4, it may arrive out-of-sequence, leading to the problem of how to fuse an older object, in this case the object originating at time k_j, into a newer global object, which resulted from data from sensor i at time k_i.

It is important that the information from the out-of-sequence sensor is properly incorporated into the global object, as it can improve the accuracy of the global state estimation. At the same time, it is important that the data from the potentially less accurate sensor i is incorporated into the global object immediately after arrival at the fusion-level, in order to keep the global object list as up-to-date as possible, eliminating any buffering of data. This can be quite critical in automotive safety applications, where crash probabilities are calculated using fused sensor data [189].

At the sensor-level, out-of-sequence data can be handled using an adapted version of the Kalman filter, which is able to integrate such out-of-sequence data [19, 22, 24, 227]. This Kalman filter can also be used at the fusion-level, but does not compensate for common information history correlation, as will be shown in the following sections. Some track-to-track fusion algorithms, in particular the information matrix fusion algorithm, however, are able to cope with out-of-sequence data, sometimes even without any special modifications.

4.3 Track-to-Track Fusion with the Common State

The fusion of two already-filtered tracks is called *track-to-track fusion* in the literature and these family of algorithms will be considered in this section. For the states in a track that are common between the global object and the sensor-level object, as per the feature selection process in the association step (see Section 3.3), point-target track-to-track fusion algorithms can be used. The uncommon states are either derived from the geometrical

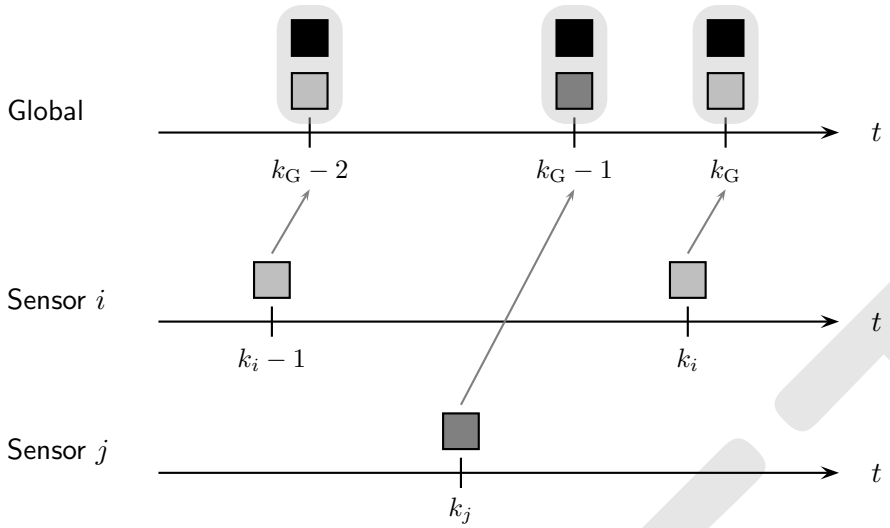


Figure 4.5: Example asynchronous fusion situation for integrating the sensor-level data from sensor i at time k_i into the global track at time k_G.

fusion algorithm or directly replace the state in the global state vector. The track-to-track fusion algorithms must be able to deal with asynchronous and possibly out-of-sequence sensors, and must do so in a sensor-to-global fusion strategy, as described in Section 3.2.2. This limits the choice of track-to-track fusion algorithms that have been widely used in the literature, since many algorithms are based on a sensor-to-sensor fusion strategy. For a comparison of synchronous track-to-track fusion algorithms, see Appendix A. Synchronous track-to-track fusion algorithms have been successfully implemented in automotive applications [42, 98, 100, 126, 160, 174, 187, 263].

This section gives an overview of different algorithms that have been proposed and can be adapted for asynchronous track-to-track fusion. Figure 4.5 shows a typical sensor configuration for asynchronous sensor-to-global fusion, where sensor i and j are asynchronous and have different latencies to the fusion-level to be fused with the global track. The notation in the following subsections describing the algorithms will be based on that of Figure 4.5, where each algorithm will be shown to fuse the data from sensor i at time k_i to the global track at time k_G, which is the arrival time of the data from sensor i. Note that all temporal alignment in the following algorithms is carried out using the predict step of the Kalman filter with a basic constant velocity kinematic model with a proper process noise, as described in Section 3.1.2. The effect of the kinematic model for small time steps for data alignment has little effect on the fusion results and therefore a simple kinematic model is used. For convenience, the notation $\hat{\mathbf{x}}_a$ will be omitted in the follow sections; it is always assumed that track-to-track fusion occurs with the common state vector between the sensor-level object and the global object.

4.3.1 Adapted Kalman Filter

The most obvious and common method for realizing track-to-track fusion in a sensor-to-global fusion strategy is to simply use a global Kalman filter and treat the sensor-level tracks as the measurement inputs. In this case, the typical Kalman filter equations are adapted such that the sensor-level tracks from sensor i at time k_i are used to update the

global Kalman filter at k_{G}. Since the assumption that the input measurements to the Kalman filter are independent of one another is violated, which is the case for sensor-level tracks, this algorithm is called the *adapted Kalman filter*. The sensor-level tracks are in fact correlated with one another over time due to their common information history, as described in Section 4.2.2.

As with the measurement-to-track Kalman filter, in a track-to-track Kalman filter, the global state estimate must be predicted forward in time from the previous track update to the measuretime time k_i of the track from sensor i, such that $k_{\mathrm{G}} = k_i$. This is accomplished using the standard Kalman filter predict equations

$$\hat{\mathbf{x}}^{\mathrm{G}}(k_{\mathrm{G}}|k_{\mathrm{G}}-1) = \mathbf{F}(k_{\mathrm{G}}, k_{\mathrm{G}}-1)\hat{\mathbf{x}}^{\mathrm{G}}(k_{\mathrm{G}}-1|k_{\mathrm{G}}-1) \tag{4.12}$$

$$\mathbf{P}^{\mathrm{G}}(k_{\mathrm{G}}|k_{\mathrm{G}}-1) = \mathbf{F}(k_{\mathrm{G}}, k_{\mathrm{G}}-1)\mathbf{P}^{\mathrm{G}}(k_{\mathrm{G}}-1|k_{\mathrm{G}}-1)\mathbf{F}(k_{\mathrm{G}}, k_{\mathrm{G}}-1)' + \mathbf{Q}(k_{\mathrm{G}}, k_{\mathrm{G}}-1) \tag{4.13}$$

The predicted global track is then updated with the sensor-level track using the adapted Kalman filter update formulation

$$\mathbf{S}(k_{\mathrm{G}}) = \mathbf{P}^{\mathrm{G}}(k_{\mathrm{G}}|k_{\mathrm{G}-1}) + \mathbf{P}^{i}(k_i|k_i) \tag{4.14}$$

$$\mathbf{K}(k_{\mathrm{G}}) = \mathbf{P}^{\mathrm{G}}(k_{\mathrm{G}}|k_{\mathrm{G}-1})'\mathbf{S}(k_{\mathrm{G}})^{-1} \tag{4.15}$$

$$\hat{\mathbf{x}}^{\mathrm{G}}(k_{\mathrm{G}}|k_i) = \hat{\mathbf{x}}^{f}(k_{\mathrm{G}}|k_{\mathrm{G}-1}) + \mathbf{K}(k_{\mathrm{G}})\left[\hat{\mathbf{x}}^{i}(k_i|k_i) - \hat{\mathbf{x}}^{\mathrm{G}}(k_{\mathrm{G}}|k_{\mathrm{G}-1})\right] \tag{4.16}$$

$$\mathbf{P}^{\mathrm{G}}(k_{\mathrm{G}}|k_i) = \left[\mathbf{I} - \mathbf{K}(k_{\mathrm{G}})\right]\mathbf{P}^{\mathrm{G}}(k_{\mathrm{G}}|k_{\mathrm{G}-1}) \tag{4.17}$$

In case of an out-of-sequence sensor, such that the sensor-level track arrives at a time $k_i < k_{\mathrm{G}}$, where k_{G} is the time of the last update for the global track, then an out-of-sequence track-to-track fusion algorithm can be used. Out-of-sequence measurement algorithms are widely known for the normal measurement-to-track Kalman filter [19, 22, 24, 227] and have been used in automotive perception applications [321–323]. In [208], the Bar-Shalom retrodiction algorithm from [19] is adapted to the problem for a track-to-track fusion Kalman filter. Using this method, the global track is retrodicted to the measurement time of the sensor-level track at k_i using

$$\mathbf{S}(k_{\mathrm{G}}) = \mathbf{P}^{\mathrm{G}}(k_{\mathrm{G}}|k_{\mathrm{G}-1}) + \mathbf{P}^{s}(k_s|k_s) \tag{4.18}$$

$$\eta(k_{\mathrm{G}}) = \hat{\mathbf{x}}^{s}(k_{\mathrm{G}}|k_{\mathrm{G}}) - \hat{\mathbf{x}}^{\mathrm{G}}(k_{\mathrm{G}}|k_{\mathrm{G}}-1) \tag{4.19}$$

$$\mathbf{P}_{vv}(k_{\mathrm{G}}, k_i|k_{\mathrm{G}}) = \mathbf{Q}(k_{\mathrm{G}}, k_i) - \mathbf{Q}(k_{\mathrm{G}}, k_i)\mathbf{S}(k_{\mathrm{G}})^{-1}\mathbf{Q}(k_{\mathrm{G}}, k_i) \tag{4.20}$$

$$\mathbf{P}_{xv}(k_{\mathrm{G}}, k_i|k_{\mathrm{G}}) = \mathbf{Q}(k_{\mathrm{G}}, k_i) - \mathbf{P}^{\mathrm{G}}(k_{\mathrm{G}}|k_{\mathrm{G}-1})\mathbf{S}(k_{\mathrm{G}})^{-1}\mathbf{Q}(k_{\mathrm{G}}, k_i) \tag{4.21}$$

$$\hat{\mathbf{x}}^{\mathrm{G}}(k_i|k_{\mathrm{G}}) = \mathbf{F}(k_i, k_{\mathrm{G}})\left[\hat{\mathbf{x}}^{\mathrm{G}}(k_{\mathrm{G}}|k_{\mathrm{G}}) - \mathbf{Q}(k_{\mathrm{G}}, k_i)\mathbf{S}(k_{\mathrm{G}})^{-1}\eta(k_{\mathrm{G}})\right] \tag{4.22}$$

$$\begin{aligned}\mathbf{P}^{\mathrm{G}}(k_i|k) = \mathbf{F}(k_i, k_{\mathrm{G}})\big[&\mathbf{P}^{\mathrm{G}}(k_{\mathrm{G}}|k_{\mathrm{G}}) + \mathbf{P}_{xv}(k_{\mathrm{G}}, k_i|k_{\mathrm{G}}) - \mathbf{P}_{xv}(k_{\mathrm{G}}, k_i|k_{\mathrm{G}}) + \\ &- \mathbf{P}_{xv}(k_{\mathrm{G}}, k_i|k_{\mathrm{G}})'\big]\mathbf{F}(k_i, k_{\mathrm{G}})'\end{aligned} \tag{4.23}$$

where $\mathbf{S}(k_{\mathrm{G}})$ is the innovation covariance at k_{G}, $\eta(k_{\mathrm{G}})$ is the innovation at k_{G}, $\mathbf{P}_{vv}(k_{\mathrm{G}}, k_i|k_{\mathrm{G}})$ is the covariance of the retrodicted process noise, $\mathbf{P}_{xv}(k_{\mathrm{G}}, k_i|k_{\mathrm{G}})$ is the cross-covariance between the process noise and the state, $\hat{\mathbf{x}}^{\mathrm{G}}(k_{\mathrm{G}}|k_{\mathrm{G}}-1)$, $\mathbf{P}^{\mathrm{G}}(k_{\mathrm{G}}|k_{\mathrm{G}-1})$ is the state and covariance of the predicted global track before the last sensor-level track update, $\hat{\mathbf{x}}^{s}(k_{\mathrm{G}}|k_{\mathrm{G}})$, $\mathbf{P}^{s}(k_s|k_s)$ is the state and covariance of the last sensor-level track used to update the global track, regardless of whether it was the same sensor as the out-of-sequence sensor, $\mathbf{F}(k_i, k_{\mathrm{G}})$ is the backward prediction where $\mathbf{F}(k_i, k_{\mathrm{G}}) = \mathbf{F}(k_{\mathrm{G}}, k_i)^{-1}$ and

$\mathbf{Q}(k_\mathrm{G}, k_i)$ is the process noise between the global track and the measurement time of the out-of-sequence track. After retrodiction, the global track is updated with the out-of-sequence sensor-level track with

$$\mathbf{S}(k_i) = \mathbf{P}^\mathrm{G}(k_i|k) + \mathbf{P}^i(k_i|k_i) \tag{4.24}$$

$$\mathbf{P}_{xz}(k_\mathrm{G}, k_i|k_\mathrm{G}) = \left[\mathbf{P}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G}) - \mathbf{P}_{xv}(k_\mathrm{G}, k_i|k_\mathrm{G})\right] \mathbf{F}(k_i, k_\mathrm{G})' \tag{4.25}$$

$$\mathbf{W}(k_\mathrm{G}, k_i) = \mathbf{P}_{xz}(k_\mathrm{G}, k_i|k_\mathrm{G})\mathbf{S}(k_i) \tag{4.26}$$

$$\hat{\mathbf{x}}^\mathrm{G}(k_\mathrm{G}|k_i) = \hat{\mathbf{x}}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G}) + \mathbf{W}(k_\mathrm{G}, k_i)\left[\hat{\mathbf{x}}^i(k_i|k_i) - \hat{\mathbf{x}}^\mathrm{G}(k_i|k_\mathrm{G})\right] \tag{4.27}$$

$$\mathbf{P}^\mathrm{G}(k_\mathrm{G}|k_i) = \mathbf{P}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G}) - \mathbf{W}(k_\mathrm{G}, k_i)\mathbf{P}_{xz}(k_\mathrm{G}, k_i|k_\mathrm{G})' \tag{4.28}$$

where $\mathbf{S}(k_i)$ is the innovation covariance for the out-of-sequence track, $\mathbf{P}_{xz}(k_\mathrm{G}, k_i|k_\mathrm{G})$ is the covariance between the retrodicted state and the out-of-sequence track, $\mathbf{W}(k_\mathrm{G}, k_i)$ is the gain for updating the global track and $\hat{\mathbf{x}}^\mathrm{G}(k_\mathrm{G}|k_i)$, $\mathbf{P}^\mathrm{G}(k_\mathrm{G}|k_i)$ is the updated global track, which now includes the information from the out-of-sequence sensor-level track.

It is important to note that simply using a global Kalman filter on already filtered sensor-level tracks completely ignores any correlation between the sensor-level tracks and the global track, due to the fact that the Kalman filter assumes that it is being updated with statistically independent measurements. Additionally, no special consideration of common process noise correlation made. Despite this, the adapted Kalman filter is quite a common method of asynchronous track-to-track fusion and has found its way into several automotive perception applications, such as the ones described in [174, 182]. In Section 4.3.4, however, it will be shown that this method has severe disadvantages.

4.3.2 Covariance Intersection

A popular method for track-to-track fusion that does not require the explicit calculation of the cross-covariance matrix between two estimates is the covariance intersection algorithm. It was introduced by Julier et al. in [132] as a means to obtain a consistent fused estimate and has been successfully used in simultaneous localization and mapping (SLAM) algorithms in [133]. Unlike the adapted Kalman filter algorithm, covariance intersection takes into account the correlation between two estimates. However, the algorithm is typically used to fuse data in a sensor-to-sensor fusion strategy, where the correlation of the estimates is due to the common process noise, as was done in [174]. The algorithm works by interpreting two state estimates in a geometrical fashion, attempting to find a fused estimate that encloses the intersection of the two estimates [132].

The covariance intersection algorithm is modified to be used in a sensor-to-global fusion strategy, where an intersection between a global track at the fusion-level is made with sensor-level tracks, which may have previously already been fused to the global track. Using a convex combination of the global and sensor-level track results in the fused covariance

$$\mathbf{P}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G})^{-1} = \omega\mathbf{P}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G}-1)^{-1} + (1-\omega)\mathbf{P}^i(k_\mathrm{G}|k_i)^{-1} \tag{4.29}$$

where $\mathbf{P}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G}-1)$ is the covariance of the predicted global track, $\mathbf{P}^i(k_\mathrm{G}|k_i)$ is the covariance of the sensor-level track predicted to its arrival time at the fusion-level and $\mathbf{P}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G})$ is the fused covariance. The fused state is then given by

$$\begin{aligned}\hat{\mathbf{x}}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G}) = \mathbf{P}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G})\big[&\omega\mathbf{P}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G}-1)^{-1}\hat{\mathbf{x}}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G}-1) + \\ &+ (1-\omega)\mathbf{P}^i(k_\mathrm{G}|k_i)^{-1}\hat{\mathbf{x}}^\mathrm{G}(k_\mathrm{G}|k_i)\big]\end{aligned} \tag{4.30}$$

where $\hat{\mathbf{x}}^{\mathrm{G}}(k_{\mathrm{G}}|k_{\mathrm{G}}-1)$ is the predicted global track, $\hat{\mathbf{x}}^{\mathrm{G}}(k_{\mathrm{G}}|k_i)$ is the sensor-level state estimate, $\hat{\mathbf{x}}^{\mathrm{G}}(k_{\mathrm{G}}|k_{\mathrm{G}})$ is the newly fused global state estimate and ω is a weighting factor. The weighting factor is determined by some form of optimization of the fused estimate from expression (4.29). A typical method is to minimize the determinate of the fused covariance:

$$\omega = \arg\min \left(\det \left(\mathbf{P}^{\mathrm{G}}(k_{\mathrm{G}}|k_{\mathrm{G}})\right)\right) \tag{4.31}$$

where $\omega \in [0, 1]$. Such a minimization is equivalent to minimizing the Shannon information of the fused Gaussian covariance, such that the most information amount of remaining information from a system is extracted [128]. Geometrically, such an approach finds a fused covariance which maximizes the volume of the intersection of the input covariances [132]. Many variations of the minimization exist, in particular fast covariance intersection [104], which aim to improve real-time computation retaining optimality fast covariance intersection.

In a sensor-to-sensor fusion strategy, the covariance intersection algorithms has been successfully used to fuse sensor data in automotive perception applications in a high-level sensor data fusion architecture [98, 100, 174].

4.3.3 Information Matrix Fusion

The information matrix fusion algorithm was proposed as a means of track-to-track fusion in [63], which differed compared to previous methods that attempted to explicitly calculated the cross-covariance between two tracks. However, the fusion configuration used required that the sensors feedback their fused state estimate to all other sensors in order for the algorithms to be optimal. Recently, the information matrix fusion algorithm has been studied again by Tian and Bar-Shalom for various sensor configurations and compared to the traditional track-to-track fusion algorithms that compute the cross-covariance [268, 270, 271]. In this section, a novel approach to track-to-track fusion for automotive applications using information matrix fusion in a sensor-to-global fusion strategy is presented and has been published in [311, 313].

In order to help explain the information matrix fusion algorithm, Figure 4.6 is used as an example. In Figure 4.6 a global track is updated with track information from sensors i and j at different points in time. At $k_{\mathrm{G}-2}$, track data from sensor i arrives at the fusion module and therefore the global track is initialized as the track from sensor i at k_{i-1}, predicted to the arrival time $k_{\mathrm{G}-2}$. The global track now contains the information from sensor i from 0 to k_{i-1}, denoted as $\{Z^i\}_0^{k_i-1}$. Track data from sensor j arrives at $k_{\mathrm{G}-1}$ and since no data has yet been integrated from sensor j into the global track, no correlation due to the common information is present, and a typical track-to-track fusion algorithms that compensates for the process noise can be used (see Appendix A). The track now contains data from sensors i and j from 0 to k_j, $\{Z^i, Z^j\}_0^{k_j}$. At k_{G}, new track data from sensor i arrives. Since track data from sensor i has already been fused into the global track at $k_{\mathrm{G}-2}$, a correlation due to the common information, $\{Z^i\}_0^{k_i-1}$, between the new track from sensor i and the global track exists. The information matrix fusion algorithm calculates from the previous sensor-level track and the newly arrived sensor-level track the information that is new to the sensor-level track, and only fuses that part into the global track, avoiding any correlation. In the current example, the new information from sensor i is $\{Z^i\}_{k_{i-1}}^{k_i}$, which is the information between the previously arrived track at k_{k-1} and the new updated track

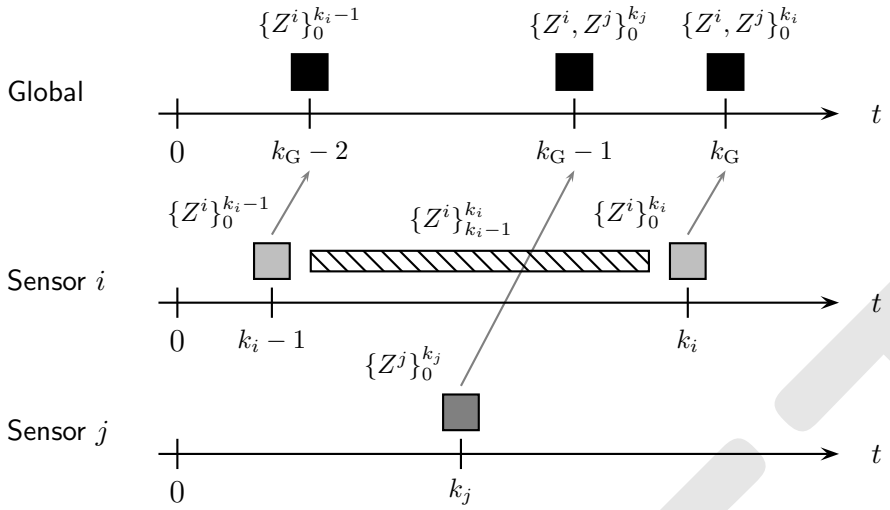


Figure 4.6: Visualization of the information matrix fusion algorithm with a global track and sensor-level tracks i and j, where the information from sensor i between $k_i - 1$ and k_i ($\{Z^i\}_{k_i-1}^{k_i}$) is fused into the global track using at k_G.

at k_i. Using the information form of the covariance, the fused covariance can be calculated by subtracting the old track's information matrix from the new track's and adding this difference to the global track's information matrix:

$$\mathbf{P}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G})^{-1} = \mathbf{P}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G}-1)^{-1} + \left[\mathbf{P}^i(k_\mathrm{G}|k_i)^{-1} - \mathbf{P}^i(k_\mathrm{G}|k_i-1)^{-1}\right] \tag{4.32}$$

where $\mathbf{P}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G})$ is the fused covariance of the global track, $\mathbf{P}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G}-1)$ is the covariance of the previous global track, $\mathbf{P}^i(k_\mathrm{G}|k_i)$ is the covariance of the newly arrived sensor-level track, $\mathbf{P}^i(k_\mathrm{G}|k_i-1)$ is the covariance of the previously fused sensor-level track and the inversion of the covariances is the information matrix. Note that all of the tracks are predicted to the arrival time of the new sensor-level track, such that the information gain ($[\mathbf{P}^i(k_\mathrm{G}|k_i)^{-1} - \mathbf{P}^i(k_\mathrm{G}|k_i-1)^{-1}]$) can be calculated and fused into the global track in its proper place in time. The fused state estimate is then updated with

$$\hat{\mathbf{x}}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G}) = \mathbf{P}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G})\Big(\mathbf{P}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G}-1)^{-1}\hat{\mathbf{x}}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G}-1) + \left[\mathbf{P}^i(k_\mathrm{G}|k_i)^{-1}\hat{\mathbf{x}}^i(k_\mathrm{G}|k_i) + \right.$$
$$\left. - \mathbf{P}^i(k_\mathrm{G}|k_i-1)^{-1}\hat{\mathbf{x}}^i(k_\mathrm{G}|k_i-1)\right]\Big) \tag{4.33}$$

where $\hat{\mathbf{x}}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G})$ is the fused state of the global track, $\hat{\mathbf{x}}^\mathrm{G}(k_\mathrm{G}|k_\mathrm{G}-1)$ is the previous state of the global track, and $\hat{\mathbf{x}}^i(k_\mathrm{G}|k_i)$ and $\hat{\mathbf{x}}^i(k_\mathrm{G}|k_i-1)$ is the newly arrived and previous sensor-level track, respectively. No special algorithm is required for out-of-sequence track information due to the fact that the decorrelation process from (4.32-4.33) is time-aligned. This is similar to the information matrix decorrelation approach in the two-step method of out-of-sequence measurement processing in a Kalman filter [24, 227], hence making the information matrix fusion algorithm an ideal approach for handling out-of-sequence track data. The formulations (4.32) and (4.33) are similar to the information filter, where the information of a measurement is extracted by the subtracting process of the new and previous sensor-level track.

The only disadvantage of the information matrix fusion algorithm is that it requires the previous sensor-level tracks in order to calculate the information gain, which effects the

memory that the algorithm requires. However, as will be shown in the following section, the information matrix fusion algorithm, as presented here, delivers the exact same results as would a centralized Kalman filter algorithm in a low-level sensor data fusion architecture.

4.3.4 Comparison

The previous sections presented several asynchronous track-to-track fusion algorithms that can be used in a sensor-to-global fusion strategy. In this section, these algorithms will be compared to one another with a simulation of an overtaking maneuver. Additionally, the information matrix fusion algorithm will be compared to a track-to-track fusion algorithm in a sensor-to-sensor fusion strategy and a centralized Kalman filter in a low-level sensor data fusion architecture. Note that the comparison will use simulated objects modeled as point-targets and therefore ignore the dimensions of an object in order to focus on the track-to-track fusion algorithms and their results. Furthermore, the problem of data association is omitted in order to focus on the performance of the track-to-track fusion algorithms.

To gauge the performance of the track-to-track fusion algorithms, two metrics are used for evaluation: estimate consistency and estimate error. A state estimate's consistency is a measure of how well the state estimate's error covariance represents the real error of the estimate. An effective way to measure a state estimate's consistency is to use the Normalized Estimation Error Squared (NEES) [25], which normalizes the error of the state estimate to its error covariance at time k

$$\epsilon(k) = \tilde{\mathbf{x}}(k)' \mathbf{P}(k)^{-1} \tilde{\mathbf{x}}(k) \tag{4.34}$$

where $\mathbf{P}(k)$ is the error covariance and the true error of the state estimate is

$$\tilde{\mathbf{x}}(k) = \mathbf{x}(k) - \hat{\mathbf{x}}(k) \tag{4.35}$$

where $\mathbf{x}(k)$ is the true state and $\hat{\mathbf{x}}(k)$ is the state estimate. The NEES is χ^2-distributed with $n_{\hat{\mathbf{x}}}$ degrees of freedom, where $n_{\hat{\mathbf{x}}} = \dim(\hat{\mathbf{x}})$ is the dimension of the state estimate. To obtain a better statistical average of an estimate's consistency, the NEES is average over N Monte Carlo runs of a simulation:

$$\bar{\epsilon}(k) = \frac{1}{N} \sum_{i=0}^{N} \epsilon(k)_i \tag{4.36}$$

where $\bar{\epsilon}(k)$ is χ^2-distributed with $N \cdot n_{\hat{\mathbf{x}}}$ degrees of freedom. An estimate is tested for consistency by determining if $\bar{\epsilon}(k)$ lies within the interval

$$\bar{\epsilon} \in [r_1, r_2] \tag{4.37}$$

The interval $[r_1, r_2]$ is chosen such that the probability

$$P(\bar{\epsilon} \in [r_1, r_2] \mid H_0) = 1 - \alpha \tag{4.38}$$

where H_0 is the hypothesis that the filter is consistent. A typical value for α is 0.05, which in effect yields a 95% confidence interval for the consistency test of a state estimate. The

estimate is considered inconsistent when $\bar{\epsilon} \notin [r_1, r_2]$, which when plotted gives an NEES below r_1 or above r_2, outside the interval.

The other metric for evaluating the performance of the track-to-track fusion algorithms is the error. Consider a state, such as the position in the x-direction, then the error is given by

$$\tilde{x}(k) = x(k) - \hat{x}(k) \tag{4.39}$$

where $x(k)$ is the true position, $\hat{x}(k)$ is the position estimate and $\tilde{x}(k)$ is the error between the two. Defining a two-dimensional position vector, $\mathbf{p}(k) = \begin{bmatrix} x & y \end{bmatrix}'$, then the Root Mean Squared Error (RMSE) of the position over N Monte Carlo runs is given by

$$\text{RMSE}(\tilde{\mathbf{p}}(k)) = \sqrt{\frac{1}{N}\sum_{i=0}^{N} \tilde{x}(k)_i^2 + \tilde{y}(k)_i^2} \tag{4.40}$$

where $\tilde{x}(k)$ is the error in the x-direction and $\tilde{y}(k)$ is the error in the y direction. Similarly, using a two-dimensional velocity vector, $\mathbf{v}(k) = \begin{bmatrix} v_x & v_y \end{bmatrix}'$, the RMSE of the velocity of an object over N Monte Carlo runs is given by

$$\text{RMSE}(\tilde{\mathbf{v}}(k)) = \sqrt{\frac{1}{N}\sum_{i=0}^{N} \tilde{v}_x(k)_i^2 + \tilde{v}_y(k)_i^2} \tag{4.41}$$

where $\tilde{v}_x(k)$ is the velocity error in the x-direction and $\tilde{v}_y(k)$ is the velocity error in the y direction.

In the sections that follow, these two evaluation metrics will be used to evaluate the fusion algorithms.

Simulation Configuration

An overtaking maneuver using a realistic sensor configuration is simulated in order to evaluate the fusion algorithms. A target vehicle is simulated in a two-dimensional environment, traveling at an initial relative velocity of $5\,\text{m/s}$ and at an initial relative position of $x = -75\,\text{m}$ and $y = 0\,\text{m}$ to the host vehicle. The overtaking vehicle's trajectory is simulated by applying an acceleration profile in the longitudinal, x, and lateral, y, directions over time. The trajectory and the acceleration profiles are shown in Figure 4.7. To initiate the overtaking maneuver, the target vehicle accelerates during the interval $t = [1.5, 5.5]\,\text{s}$ with $a_x^{\max} = 1.5\,\text{m/s}^2$. To execute a lane change maneuver, lateral acceleration of $|a_y^{\max}| = 1\,\text{m/s}^2$ is applied between $t = [2.5, 5.5]\,\text{s}$ to move into the overtaking lane and again between $t = [6.5, 9.5]\,\text{s}$ to move back into the original lane to complete the maneuver. At the end of the maneuver, a braking maneuver is simulated between $t = [9.5, 12.5]\,\text{s}$ with $a_x^{\max} = -2\,\text{m/s}^2$.

A realistic surround environment perception sensor configuration is used during the simulation, similar to the one used in the test vehicle described in Appendix 7.1. A sensor in the simulation is described by five parameters: the measurement period, latency, longitudinal measurement standard deviation (σ_x), lateral measurement standard deviation (σ_y) and the measurement interval. The measurement period defines how often the sensor takes a measurement and the latency is the time it takes for the data to arrive at the fusion model after the measurement is taken. The standard deviations describe the accuracy of

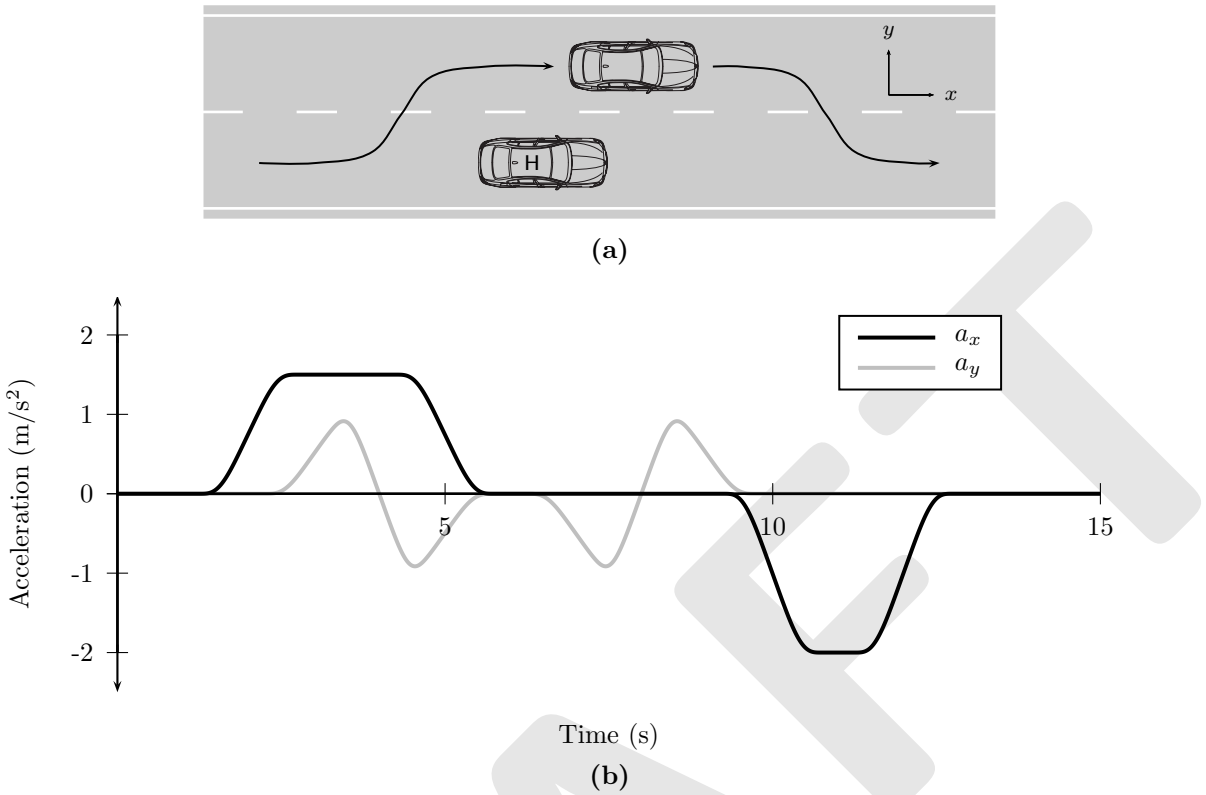


Figure 4.7: Simulated scenario, where the target vehicle overtakes the host vehicle (a) with the acceleration profiles in the x-direction, a_x, and y-direction, a_y, (b), simulating acceleration to overtake the host vehicle, a lane change maneuver and a braking maneuver after overtaking.

the sensor's measurements. The time during which the sensor is able to take a measurement during the simulation is given by the measurement interval, which simulate the sensors' different field of views. The sensor configuration for the simulation is summarized in Table 4.1. Two sensors observe the rear area of the vehicle, two the front and one the side. The pairs of sensors observing the front and the rear have complementary performance in their standard deviations, where the sensor with the slightly better performance has a larger latency and is therefore an out-of-sequence sensor. The sensors in the rear and the front overlap with the sensor from the side, such there is a one-second transition period where both the rear and side, or front and side, sensors observing the overtaking vehicle. There is also a one-second gap, starting at $t = 6\,\mathrm{s}$ where only the side sensor observes the overtaking vehicle.

The simulation is carried out by generating a noisy trajectory, perpetrated by a process noise with spectral density $\tilde{q} = 0.5$. From this trajectory, each sensor takes noisy measurement as per their defined measurement period and interval. Each sensor then process its measurements using a Kalman filter, producing tracks. The tracks, based on the sensor latency, are then sorted according to their arrival time at the fusion level and then processed using a track-to-track fusion algorithm. In addition to the high-level method of processing the sensor data, a low-level method is carried out by sorting the measurements of all of the sensors according to their arrival time and then processing the measurements using a single, centralized Kalman filter.

Table 4.1: Sensor configuration for the simulation of an overtaking maneuver.

Sensor	Rear 1	Rear 2	Side	Front 1	Front 2
Measurement period (ms)	80	60	100	80	60
Latency (ms)	40	150	80	40	150
σ_x (m)	1.50	0.25	1.00	1.50	0.25
σ_y (m)	0.75	1.75	1.50	0.75	1.75
Measurement interval (s)	$[0, 5]$	$[1, 6]$	$[5, 8]$	$[7, 12]$	$[8, 15]$

Table 4.2: Average root mean square error for the position and velocity of the sensor-to-global track-to-track fusion algorithms.

Algorithm	μ, Position (m)	μ, Velocity (m/s)
Information matrix fusion	0.51	1.00
Covariance intersection	0.66	1.23
Adapted Kalman filter	0.68	1.20

Evaluation of Sensor-to-Global Track-to-Track Fusion Algorithms

First, the three asynchronous track-to-track fusion algorithms presented in the previous sections (adapted Kalman filter, covariance intersection and information matrix fusion) are compared to one another. The RMSE of the position and the velocity for the three algorithms for the duration of the simulation is shown in Figure 4.8. The information matrix fusion provides the best performance, having the lowest error in both position and velocity. Though having a smooth error signal, the adapted Kalman filter has the worst error of the three algorithms. The covariance intersection algorithm is slightly better than the adapted Kalman filter, but its error fluctuates and is a bit unstable. The mean RMSE over time for the three algorithms is summarized in Table 4.2. The result clearly shown that the information matrix fusion algorithm offers the best performance.

The consistency of the fused tracks using the three track-to-track fusion algorithms is evaluated using the NEES. The NEES for the three algorithms is shown in Figure 4.9, with the 95% confidence interval depicted by the two horizontal dashed lines. In Figure 4.9a, it can be seen that the consistency of the adapted Kalman filter algorithm diverges immediately and eventually reaches a value of around 80, which is far above the 95% confidence interval upper bound of 6.70. Figure 4.9b zooms in on the confidence interval in order to show the results of the consistency of the covariance intersection and information matrix fusion algorithms. As with the error, the covariance intersection algorithm exhibits instabilities. Only the information matrix fusion algorithm delivers a consistent result for most of the duration of the simulation, being only slightly inconsistent shortly after the first lane change maneuver at around $t = 5\,\mathrm{s}$.

In a probabilistic framework for driver assistance systems, it is important that the state estimates are consistent and that the uncertainty realistically represents the true error of the estimate. The uncertainty information is used at the application-level of the proposed high-level sensor data fusion architecture in order to carry out probabilistic-based situation interpretations for various applications. For a pre-crash application, for example, the state estimate and its error covariance can be used to calculate a crash probability [190]. In

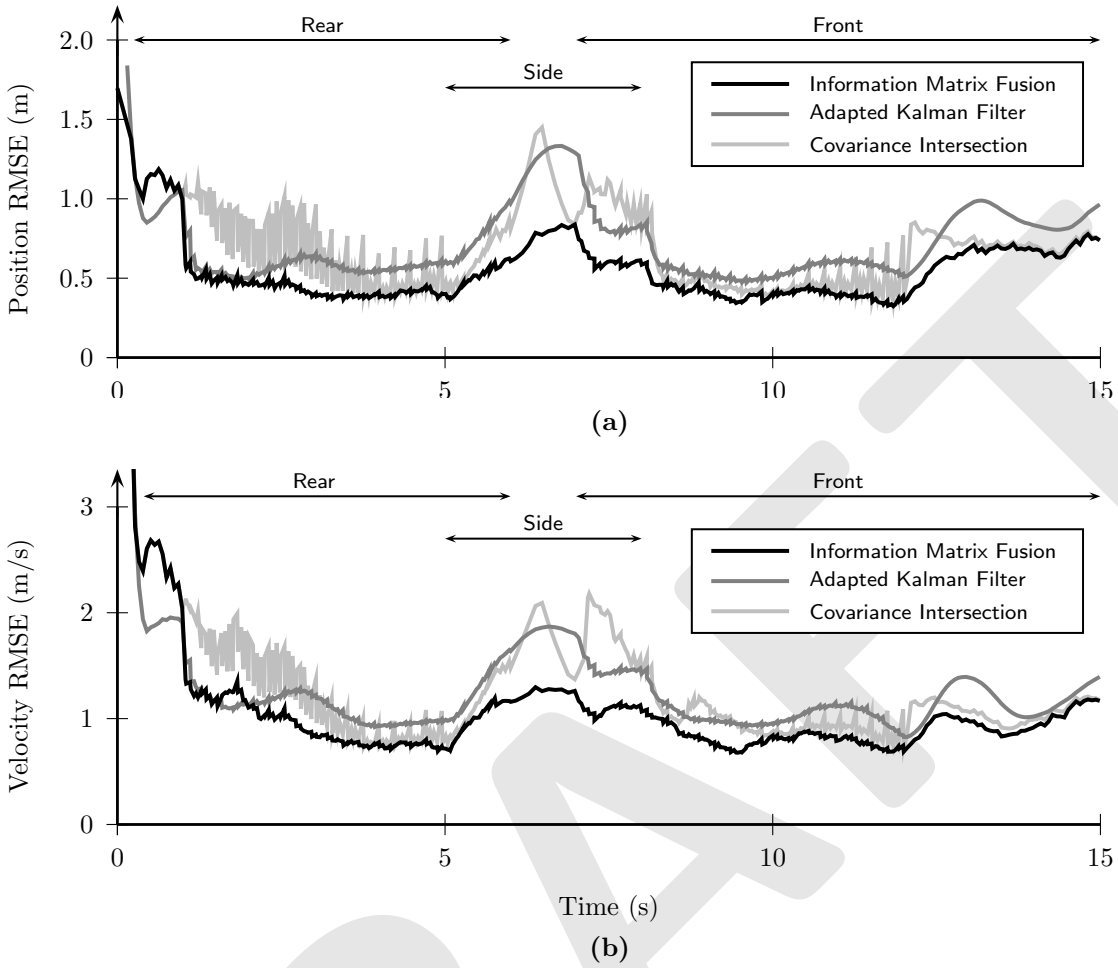
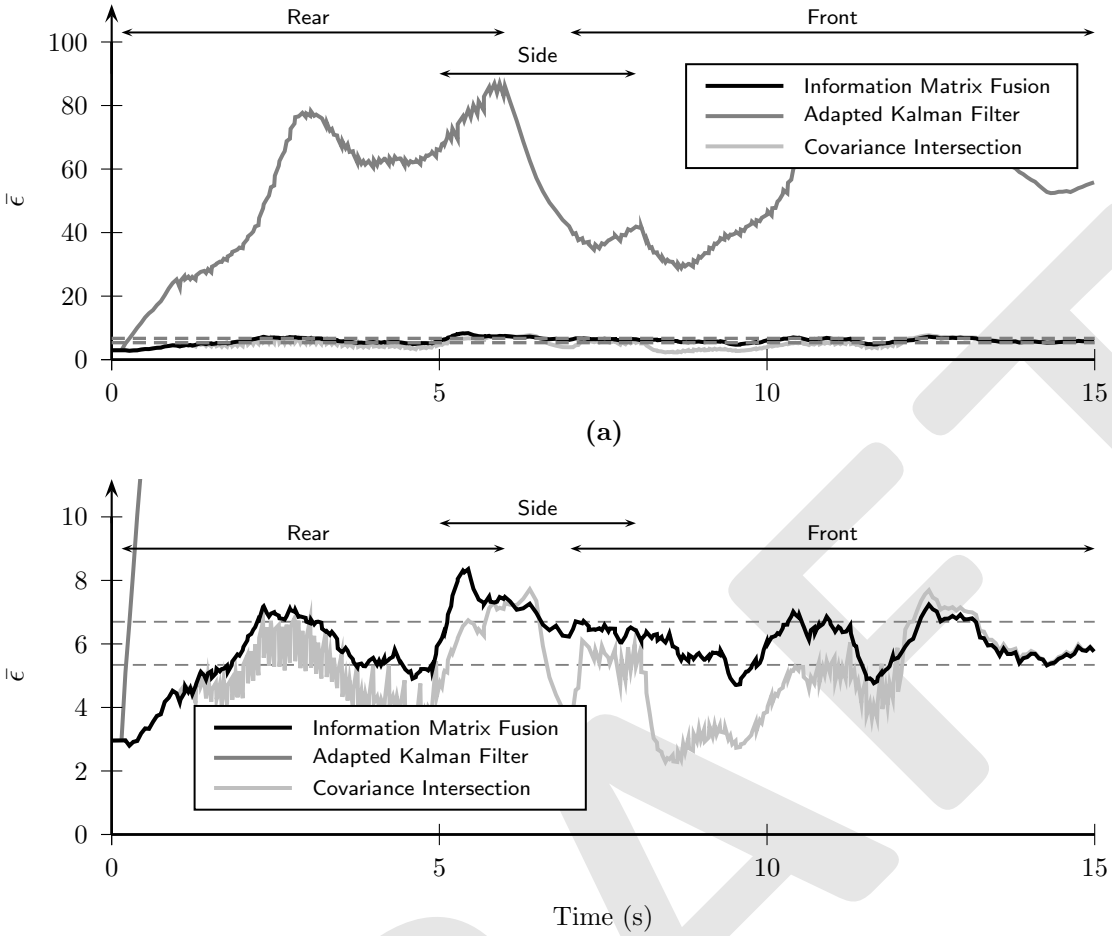


Figure 4.8: The RMSE for the position (a) and velocity (b) of the three evaluated track-to-track fusion algorithms using a sensor-to-global fusion strategy.

highly automated driving applications, the uncertainty in the detect objects can be used to determine worst-case situation prediction in order to guarantee safe and reliable decision on vehicle maneuvers, such as automated lane changes [6, 7].

Evaluation of Fusion Strategies and Architecture

In a sensor-to-global fusion strategy, the information matrix fusion algorithm delivers the best results. In this section, fusion strategies and fusion architectures will be compared to one another. The sensor-to-global fusion strategy is compared to a sensor-to-sensor fusion strategy and a low-level sensor data fusion architecture. The sensor-to-sensor fusion strategy is implemented with a 100 ms fusion cycle, where the buffered sensor data between the current and previous fusion cycle is fused together to create a global track. A new global track is created at each fusion cycle, where the previous global track is not used, hence a memory-less fusion process [268]. A typical synchronous track-to-track fusion algorithm which calculates the cross-covariance (here donated as the *use of cross-covariance* (UoCC) algorithm) between the sensor-level tracks is used, as further described in Appendix A. A low-level fusion architecture is implemented by sorting the measurements from all of the sensors according to their arrival time and processing the measurements using a centralized



(b) Zoom of (a).

Figure 4.9: Consistency test using the NEES of the three evaluated track-to-track fusion algorithms using a sensor-to-global fusion strategy.

Kalman filter algorithm. In case of an out-of-sequence measurement, the retrodiction method of processing the out-of-sequence measurement is used [19, 22].

The root mean square error of the position and the velocity of the different fusion strategies and architectures is shown in Figure 4.10. Two phenomenon occur with the use of cross-covariance algorithm. First, the error momentarily increases, for example at $t = [1.1, 2.3, 4.7]\,\mathrm{s}$, due to the fact that at that particular fusion cycle, no data arrives from the out-of-sequence sensor because its latency is larger than the fusion cycle. The other problem with the use of cross-covariance algorithm is that as the observed vehicle exits the field of view of a sensor, such as at $t = 6\,\mathrm{s}$ and $t = 12\,\mathrm{s}$, the error also increases. Since track-to-track fusion using a sensor-to-sensor fusion strategy is memory-less, data from a sensor that is available at one fusion cycle, but not at the next, is completely lost at the new fusion cycle. The information matrix fusion algorithm actually performs the same as the centralized Kalman filter algorithm in a low-level fusion architecture, typically considered to be the optimal method of fusion. The mean RMSE over the simulation is summarized in Table 4.3. Here it can be seen that the information matrix fusion algorithm actually slightly outperforms the centralized Kalman filter. This is due to the fact that the centralized Kalman filter uses the retrodiction method of out-of-sequence processing,

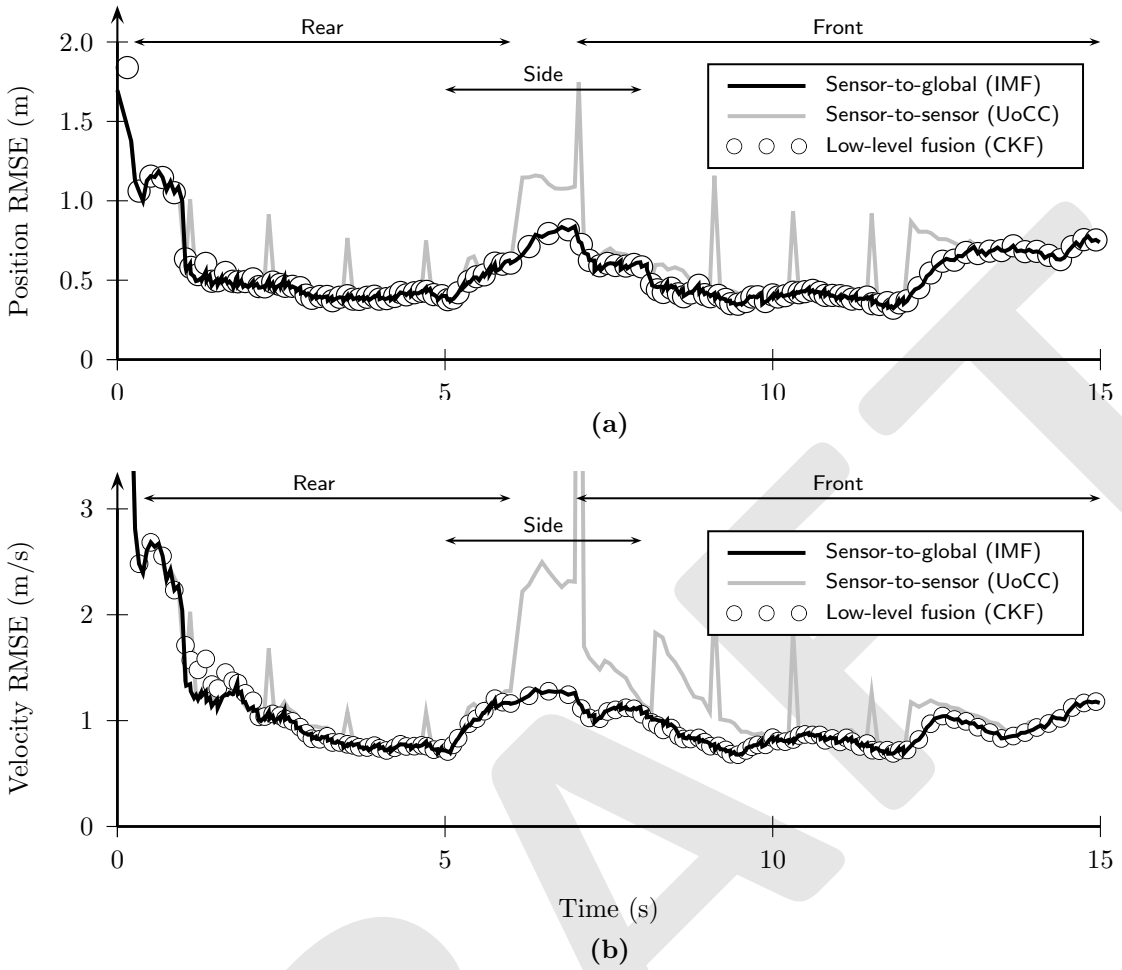


Figure 4.10: The root mean square error (RMSE) of the position (a) and velocity (b) estimations for the different fusion strategies and a level-fusion using a centralized Kalman filter (CMF).

which is only optimal in a 1-lag out-of-sequence situation, which is not the case with the simulated sensor configuration.

The consistency of the different fusion strategies and the low-level architecture is shown in Figure 4.11. Similar results appear for the consistency. The use of cross-covariance algorithm is inconsistent during various portions of the simulation. The information matrix fusion algorithm and the centralized Kalman filter again have the same result in the NEES, showing that the information matrix fusion algorithm performs just as well as a low-level fusion architecture.

Summary

Based on the results from this section, Table 4.4 gives a quick summary of the comparison between the different track-to-track fusion algorithms for various criteria. The information matrix fusion algorithms demonstrated the best accuracy, as shown in Figure 4.8 and Figure 4.10, where the other algorithms tend to have slightly less accurate state estimates. The consistency, presented in Figure 4.9 and Figure 4.9 was also shown to be best with the information matrix fusion algorithm, where the adapted Kalman filter performs particularly poorly due to the accumulation of the correlated information history. All of the

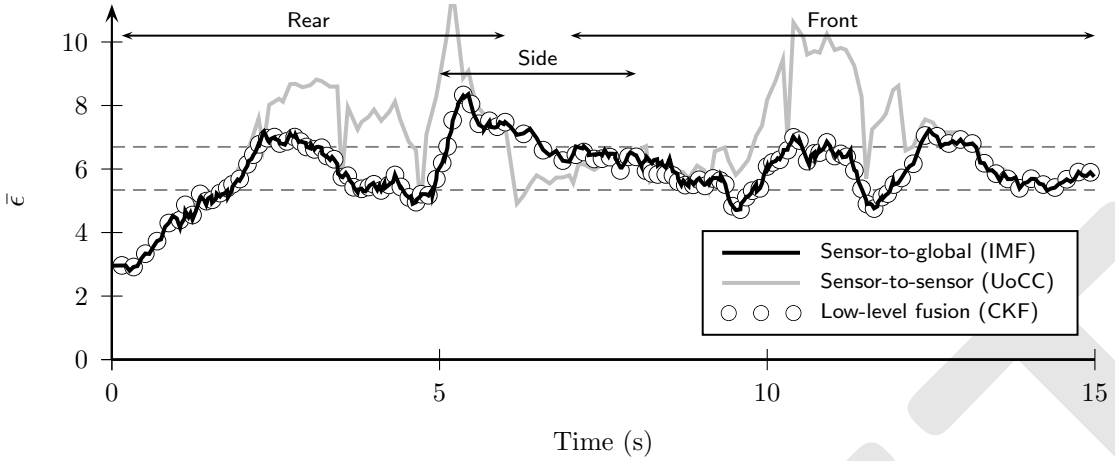


Figure 4.11: The NEES for the sensor-to-global and sensor-to-sensor fusion strategies using the information matrix fusion (IMF) algorithm and the use of cross-covariance (UoCC) algorithms and also a low-level fusion using a centralized Kalman filter (CKF).

Table 4.3: Average position and velocity root mean square error for different fusion strategies and low-level centralized Kalman filter processing.

Fusion Strategy	μ, Position (m)	μ, Velocity (m/s)
Sensor-to-global (IMF)	0.51	1.00
Sensor-to-sensor (UoCC)	0.64	1.32
Centralized Kalman filter	0.52	1.02

algorithms, except for the use of cross-covariance algorithm, are able to process data arriving from sensor in an asynchronous fashion using a sensor-to-global fusion strategy. Due to the nature of the information matrix fusion's decorrelation process, no special modifications are necessary in order to process out-of-sequence data, whereas the other algorithms do not perform optimally unless properly addressed. The computational efficiency of the information matrix fusion algorithm is not as positive compared to the some of the other algorithms due to its reliance on several matrix inversion operations, which can be time consuming to compute on platforms with limited processing capability, particularly for higher dimensional states. Memory is one of the main drawbacks of the information matrix fusion algorithm, as all previous state estimates from all of the input sensors, as well as the global state estimates, need to be saved in order to perform the decorrelation on which the algorithm relies.

4.4 Geometrical Fusion using the Object Model

The geometry of the object is handled differently than the position and dynamical states of the object, hence the separate dimension vector $\mathbf{d}$. This is due to the fact that the geometry cannot be easily modeled with traditional tracking and track-to-track fusion algorithms. This is partially due to the fact that different sensors have different estimation performance regarding the geometry, but more importantly due to the fact that an object's geometry, as observed by a sensor, varies greatly depending on the sensor's position and orientation

Table 4.4: Comparison of track-to-track fusion algorithms (+: positive, −: negative, +/−: neutral).

	Information Matrix Fusion	Covariance Intersection	Adapted Kalman Filter	Use of Cross Covariance
Accuracy	+	+/−	+/−	+/−
Consistency	+	+/−	−	+/−
Asynchronous	+	+	+	−
Out-of-Sequence Data	+	+/−	+/−	−
Computation	+/−	+/−	+	+
Memory	−	+	+/−	−

relative to the observed object over time. Previous work, such as [139], have attempted to estimate the dimensions of an object by including it in the normal state-based filter, where the observability of the dimensions are carefully modeled in the measurement model. The problem is that such models for determining the observability turn out to be quite heuristic and error-prone in practice. Furthermore, the geometry of an object, as opposed to the position and dynamics, is a static property: once confirmed to be fully observed with a high degree of confidence, then the geometry of the object should not change anymore over time, which differs from the idea of state-based estimation filter, where the states of an object are allowed to converge given new measurement information.

In this section, a novel 1-dimensional grid-based algorithm for accumulating information about an object's geometry, in particular its length and width, is presented in order to estimate the fused object's dimension. Furthermore, the relationship between the feature-based association and state vector fusion methods play an important role together with the object's dimensions in order to properly estimate and extract an object's position, as it relates to its dimensions.

4.4.1 Dimension Estimation

Since the dimensions, $\mathbf{d}$, of an object, in the two-dimensional case its length and width, as seen by different sensors at a similar time can vary greatly, for example an object which is partially seen from a sensor with different field-of-views and angle to the object, it is not practical to fuse or filter the dimensions using traditional algorithms. In this section, a novel approach for estimating the dimensions of an object at the fusion-level is presented.

The idea for estimating and updating the dimensions of an object are based on the methods of occupancy grid mapping [266]. For each dimension d, where $d \in \{w, l\}$ for a 2-dimensional object, a 1-dimensional map $\mathbf{m}^d$ with a discretization Δd and maximum size $d_{\max}$ is established, representing the probability that a location along the dimension has been observed by a sensor. Such a dimension map, along with the notation used, is shown in Figure 4.12 for the length and width of a 2-dimensional object.

The goal is to estimate the estimated length or width of the global object at time t

$$p(d^{\mathrm{G}}(t) \mid \{d^{\mathrm{S}_1 \ldots \mathrm{S}_N}\}_0^t) \tag{4.42}$$

conditioned on the length or width estimations from the sensors S_1 to S_N for all observed times 0 to t of the object. A 1-dimensional map $\mathbf{m}^d$ along the dimensions of the object

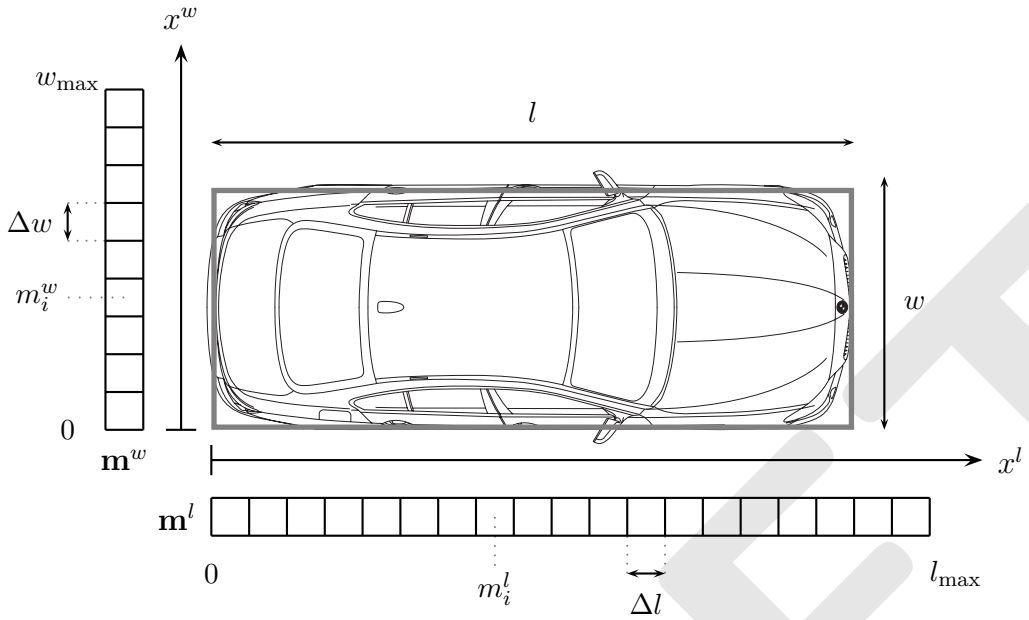


Figure 4.12: Updating the dimensions of an object at the fusion-level using a 1-dimensional grid map along each dimension.

is used to estimate the dimensions, where an element m_i^d represents the probability that the dimension up to that element has been observed by sensor data. The size of such a dimension map is defined by $\dim(\mathbf{m}^d) = \lceil d_{\max}/\Delta d \rceil$ where Δd is the discretization of the map. As in the problem of occupancy grid mapping, it is assumed that each element in the map is independent of the others such that

$$p(\mathbf{m}^d \mid \{d^{\mathrm{S}_1 \ldots \mathrm{S}_N}\}_0^t) = \prod_i p(m_i^d \mid \{d^{\mathrm{S}_1 \ldots \mathrm{S}_N}\}_0^t) \tag{4.43}$$

which allows for the calculation of each element $p(m_i^d \mid \{d^{\mathrm{S}_1 \ldots \mathrm{S}_N}\}_0^t)$ separately. A binary Bayes filter [266] is used to recursively calculate the probability that an element of the dimension map has been observed by sensor data with

$$p(m_i^d \mid \{d^{\mathrm{S}_1 \ldots \mathrm{S}_N}\}_0^t) = \frac{p(m_i^d \mid d^{\mathrm{S}})\, p(d^{\mathrm{S}})\, p(m_i^d \mid \{d^{\mathrm{S}_1 \ldots \mathrm{S}_N}\}_0^{t-1})}{p(m_i^d)\, p(d^{\mathrm{S}} \mid \{d^{\mathrm{S}_1 \ldots \mathrm{S}_N}\}_0^{t-1})}. \tag{4.44}$$

In order to avoid numerical instabilities, the log odds ratio is used to efficiently calculate (4.44) with

$$\begin{aligned} \mathcal{L}_i^d(t) &= \log \frac{p(m_i^d \mid \{d^{\mathrm{S}_1 \ldots \mathrm{S}_N}\}_0^t)}{1 - p(m_i^d \mid \{d^{\mathrm{S}_1 \ldots \mathrm{S}_N}\}_0^t)} \\ &= \log \frac{p(m_i^d \mid d^{\mathrm{S}})}{1 - p(m_i^d \mid d^{\mathrm{S}})} + \log \frac{p(m_i^d \mid \{d^{\mathrm{S}_1 \ldots \mathrm{S}_N}\}_0^{t-1})}{1 - p(m_i^d \mid \{d^{\mathrm{S}_1 \ldots \mathrm{S}_N}\}_0^{t-1})} + \log \frac{1 - p(m_i^d)}{p(m_i^d)} \\ &= \log \frac{p(m_i^d \mid d^{\mathrm{S}})}{1 - p(m_i^d \mid d^{\mathrm{S}})} + \mathcal{L}_i^d(t-1) - \log \frac{p(m_i^d)}{1 - p(m_i^d)} \end{aligned} \tag{4.45}$$

where $p(m_i^d \mid d^{\mathrm{S}})$ is the estimate with current sensor data, $\mathcal{L}_i^d(t-1)$ is the previous estimate for this element in log odds ratio form and $p(m_i^d)$ is the prior probability for the element,

typically chosen to be $p(m_i^d) = 0.5$. The probability of the element m_i^d can be recovered from the log odds ratio form with

$$p(m_i^d \mid \{d^{\mathrm{S}_1 \ldots \mathrm{S}_N}\}_0^t) = 1 - \frac{1}{1 - e^{\mathcal{L}_i^d(t)}}. \tag{4.46}$$

The probability that an element is observed based on the current sensor's estimate of the dimension, $p(m_i^d \mid d^{\mathrm{S}})$, is modeled as a piecewise function where the complementary cumulative distribution function of the normal distribution of the dimension estimate with mean d^{S} and standard deviation σ_d^{S}, weighted by a maximum update probability $P_{d_{\max}}$ is used for $x_i^d \leq d^{\mathrm{S}}$ and the sigmoid function for $x_i^d > d^{\mathrm{S}}$:

$$p(m_i^d \mid d^{\mathrm{S}}) = p_d(x_i^d, d^{\mathrm{S}}, \sigma_d^{\mathrm{S}}) = \begin{cases} P_{d_{\max}} \cdot \left(1 - \frac{P_{d_{\max}}}{2}\left[1 + \operatorname{erf}\left(\frac{x_i^d - d^{\mathrm{S}}}{\sigma_d^{\mathrm{S}}\sqrt{2}}\right)\right]\right) & x_i^d \leq d^{\mathrm{S}} \\ \frac{-p(m_i^d)}{1 + e^{-k(x_i^d - \frac{d^{\mathrm{S}} + d_{\max}}{2})}} + p(m_i^d) & x_i^d > d^{\mathrm{S}} \end{cases} \tag{4.47}$$

where erf() is the error function and k is the steepness of the sigmoid function. An example of this update model is shown in Figure 4.13. To update an element m_i^d, the mid-point value of the cell, x_i^d, is used with the function $p_d(x_i^d, d^{\mathrm{S}}, \sigma_d^{\mathrm{S}})$ in order to calculate the update probability to be used in (4.45). This model of the dimension update function was chosen for several important reason. First, the sensor-level dimension estimate, up to the dimension value itself, is given a confidence larger than the prior probability $p(m_i^d)$, where higher confidence is given at the lower end of σ_d^{S}, therefore resulting in a positive increase of the dimension's grid model. Since the sensor-level dimension estimation is a normal distribution, the complementary cumulative distribution function of the normal distribution is naturally chosen to model this behavior. As the observability of the dimension model is not explicitly modeled, it is assumed that previous observations were fairly correct. Therefore the model after d^{S} should not immediately continue towards 0, as the complementary cumulative distribution function would do, but should retain the prior such that previous observations of the dimension are not negatively influenced. However, in order to counter act temporary false large dimension estimation at the sensor-level, a sigmoid function was chose beyond d^{S} such that a very gradual decline in confidence up to $d_{\max}$ is modeled.

Given the grid-based accumulation of the dimensions of the object, it is desired to extract an estimate of the dimension using the grid $\mathbf{m}^d$. As with the update model, it is assumed that the accumulated grid, in a histogram form, roughly approximates a discrete complementary cumulative distribution function of a normal distrubition, such that

$$p(\mathbf{m}^d \mid \{d^{\mathrm{S}_1 \ldots \mathrm{S}_N}\}_0^t) \approx 1 - F_{d^{\mathrm{G}}}(m_i^d). \tag{4.48}$$

A normal distribution estimating the dimension of the object from the grid can be recovered by taking the finite difference of the discrete cumulative distribution function $F_{d^{\mathrm{G}}}(m_i^d)$, which results in a discrete probability distribution approximating a normal distribution $\mathcal{N}(d^{\mathrm{G}}, \sigma_{d^{\mathrm{G}}}^2)$:

$$p(d^{\mathrm{G}}(t) \mid \{d^{\mathrm{S}_1 \ldots \mathrm{S}_N}\}_0^t) \approx \frac{dF_{d^{\mathrm{G}}}(m_i^d)}{dm_i^d} \sim \mathcal{N}(d^{\mathrm{G}}(t), \sigma_{d^{\mathrm{G}}}^2(t)). \tag{4.49}$$

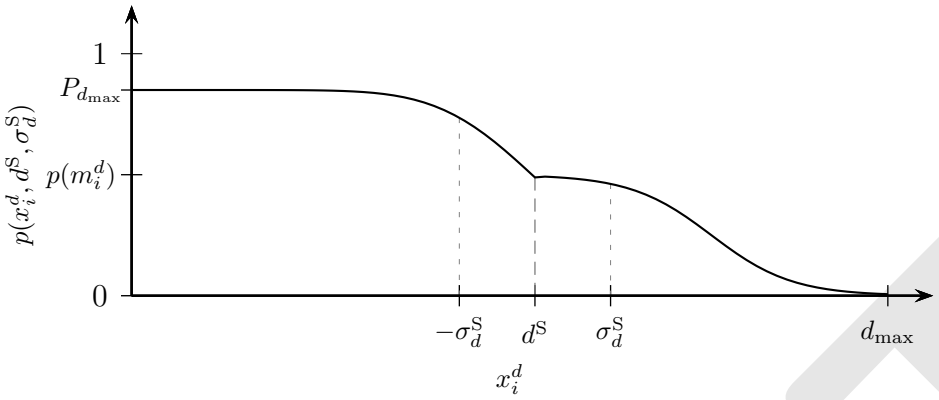


Figure 4.13: Probability function used for updating the dimension grid map with a sensor's current dimension estimate d^{S} and standard deviation σ_d^{S}.

From this discrete probability distribution function, the mean and variance of the dimension can be calculated with

$$d^{\mathrm{G}}(t) = \sum x_i^d p_i \tag{4.50}$$

$$\sigma_{d^{\mathrm{G}}}^2(t) = \sum \left(x_i^d - d^{\mathrm{G}}\right)^2 p_i \tag{4.51}$$

where p_i is the probability of the value at x_i^d from the discrete probability function $p(d^{\mathrm{G}}(t) \mid \{d^{\mathrm{S}_1 \ldots \mathrm{S}_N}\}_0^t)$. The new estimated dimension vector, for the two-dimensional case, is then

$$\hat{\mathbf{d}}(t) = \begin{bmatrix} l^{\mathrm{G}}(t) \\ w^{\mathrm{G}}(t) \end{bmatrix} \tag{4.52}$$

with the new dimension uncertainty vector of

$$\mathbf{d}_{\sigma^2}(t) = \begin{bmatrix} \sigma_{l^{\mathrm{G}}}^2(t) \\ \sigma_{w^{\mathrm{G}}}^2(t) \end{bmatrix}. \tag{4.53}$$

4.4.2 Extract Fused Coordinates

After fusion of the common associated state and the object's geometry, the object's new coordinates, as defined by the object model interface in Section 2.2.1, must be extracted. The new box-middle coordinates of the object are extracted from the fused association state and the newly extracted dimensions of the object. In order to extract the proper coordinates, the type of feature constellation, determined during state vector association (see Section 3.3.2) is used.

For the feature constellation cases "common corner feature" and "common side feature", as described in Section 3.3.2, the selected feature $\mathbf{p}_{\mathrm{f}}^{\mathrm{veh}}$ used during association and fusion is used as the base feature for extracting the new fused box-middle coordinate of the object. In the feature constellation case where the feature lie on a common side, the feature which belongs to the global object and which was used as the origin in geometrical association, $\mathbf{p}_{\mathrm{f_1}}^{\mathrm{G,veh}}$, is used as the base feature.

Once the base feature is identified, a transformation similar to the one described in Section 3.3.2, where in this case the opposite transformation is desired. The vector describing the fused base feature in the object coordinate system is the opposite of that used

in Section 3.3.2, denoted here as $\mathbf{p}_{-\mathrm{f}}^{\mathrm{obj}}$, where the negative sign inverts the direction, such that

$$\mathbf{p}_{-\mathrm{f}}^{\mathrm{obj}} = \begin{bmatrix} -x_{\mathrm{f}}^{\mathrm{obj}} & -y_{\mathrm{f}}^{\mathrm{obj}} & 1 \end{bmatrix}'. \tag{4.54}$$

The object to vehicle transformation matrix also differs in that the base feature's position in the vehicle coordinate system is used in the transformation matrix:

$$\mathbf{H}_{\mathrm{obj}\to\mathrm{veh,f}} = \begin{bmatrix} \cos\psi & -\sin\psi & x_{\mathrm{f}}^{\mathrm{veh}} \\ \sin\psi & \cos\psi & y_{\mathrm{f}}^{\mathrm{veh}} \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.55}$$

By similarly applying (3.8), the box-middle position of the object, $\mathbf{p}^{\mathrm{veh}}$, can be determined with

$$\mathbf{p}^{\mathrm{veh}} = \mathbf{H}_{\mathrm{obj}\to\mathrm{veh,f}}\mathbf{p}_{-\mathrm{f}}^{\mathrm{obj}} \tag{4.56}$$

where the length and the width used in $\mathbf{p}_{-\mathrm{f}}^{\mathrm{obj}}$ is the estimation of the fused length and width, described in the previous section.

Discussion

In a high-level sensor data fusion architecture, the challenge is to fuse the state estimate of already filtered data from several number of sensors. This requires the use of track-to-track fusion algorithms which consider the temporal correlation of data. In this chapter, it was shown that the information matrix fusion algorithm can deliver the same performance in a high-level sensor data fusion architecture as a low-level central Kalman filter approach and is therefore considered the algorithm of choice for track-to-track fusion. However, the algorithm comes with a slight cost in memory, where for each sensor the previous estimate of the state and covariance must be saved in order to calculate the information gain to be fused into the global object. If memory is an important issue in an application, the covariance intersection algorithm also produces favorable results and can be used as a suboptimal track-to-track fusion algorithm in a sensor-to-global fusion strategy.

An often overlooked aspect of fusion algorithms is the resulting covariance estimation. Automated driving applications are showing the trend of relying heavily on probabilistic approaches in situation interpretation, prediction and planning algorithms. Once a probabilistic description of an object is required, it is important to guarantee that the resulting covariance after fusion represents the true error of the estimation as much as possible. An overestimate error covariance may result in later algorithms "trusting" the data too much, which could result in critical situations in automated driving. A far too conservative error covariance could result in many false reactions of an automated driving system which may end up driving far too conservatively than desired. In this chapter, it was shown, through the evaluation of the algorithms using the NEES metric, that the information matrix fusion algorithm results in the most consistent error covariance estimation. The straight-forward fusion approach using a second filtering algorithm at the fusion-level in a high-level fusion architecture has severe consequences on the error covariance, resulting in a far over-estimated result, which could have adverse effects on further algorithms using these results. This approach should be avoided at all costs and an algorithm such as information matrix fusion or covariance intersection should be used instead.

Furthermore, care must be taking in considering the geometrical shape of an object in automotive application. Most track-to-track fusion algorithms in the literature thus far have been developed with an aerospace application in mind, where targets could be considered as point-objects. In the automotive environment, other objects are represented using a rectangular box model, making the choice of and treatment of which point on the box to fuse non-trivial. Additionally, a proper algorithm for fusing the dimension of an object must be used. Both of these issues were resolved in this chapter by considering a parallel state vector and geometrical fusion approach, where based on the feature constellation after association, track-to-track fusion is carried out on on a specific common feature between the sensor and global object and the uncommon features are then treated using a grid map motivated approach for dimension estimation.

In scenarios with more dense traffic with varying types of objects, additional methods may need to be considered in order to extend the track-to-track and geometrical fusion approaches described in this chapter. A tighter coupling between association and fusion could help improve results by extending known JPDA algorithms for sensor-level tracking to the track-to-track fusion problem, since some sensors tend to output multiple objects for certain objects, such as large trucks. This would improve the robustness of fusion-level algorithms against weak sensor-level results. Considering multiple geometrical models or a more general geometrical model, for example local object grids typically used at the sensor-level [11, 12, 240, 241], could also improve the estimation of objects which do not fit well to the rectangular box-model used in this thesis.

5 Existence Probability

In object tracking, it is very important to have a good estimate of the state and the error covariance, as discussed in the previous chapter. In real applications, however, sensor data is far from perfect and false sensor measurements, which do not represent a relevant object for a specific application, may occur. A false sensor measurement which is interpreted by the object tracking algorithms as a real measurement is called a *false positive*. Contrary, a sensor measurement which actually does come from a relevant object, such as a vehicle, is called a *true positive*. In automotive environments, sensors generate a lot of false positives, such as from trees, bushes, guard rails and other sources. Since these are not relevant traffic objects for a driver assistance application, it is desired to filter these objects out.

In essence, whether a detected object is a real and relevant object or if it is a false detection is a measure of the quality of the detected object. In this chapter, the idea of an existence probability is introduced as a means of measuring the quality of detected objects. The basic idea is that each detected object is assigned a probability which represents the existence and relevance of the detected object. Using the existence probability, sporadic detections, such as the ones from trees and bushes, for example, can be filtered out using a threshold. Traditional methods for measuring an object's quality are done using a track score, where a track's quality measure is increased the more often it is observed [40]. The advantages of using an existence probability is that it can be modeled in various ways using different probabilistic methods, allowing for a single generic representation of object quality, even though the methods for calculating the quality may differ between sensors.

In this chapter, algorithms for calculating the existence probability are presented. Section 5.1 presents an algorithm for calculating the existence probability at the sensor-level. The fusion of existence probabilities from many sensors at the fusion-level is presented in Section 5.2. At the application-level, the existence probability can be used in situation interpretation algorithms in order to carry out various driver assistance functionalities.

5.1 Sensor-Level Processing

Many methods exist for producing an existence probability for sensor-level tracking of objects. A popular approach is to use an Integrated Probabilistic Data Association (IPDA) algorithm [198, 199], where data existence estimation is integrated with a probabilistic data association algorithm. This approach was later further developed into the joint integrated probabilistic data association (JIPDA) algorithm [197]. For automotive environment perception, IPDA [169] and JIPDA [120, 121, 170, 172, 191, 194, 195] have been extensively used. Another approach is to more loosely decouple existence probability estimation with state estimation, where a 1-to-1 data association method, such as global nearest neighbor, can be used. Popular methods for calculating a more decoupled existence probability are based on Bayesian formulations. A confidence measure using a generalized Bayes estimator for automotive applications was done in [3].

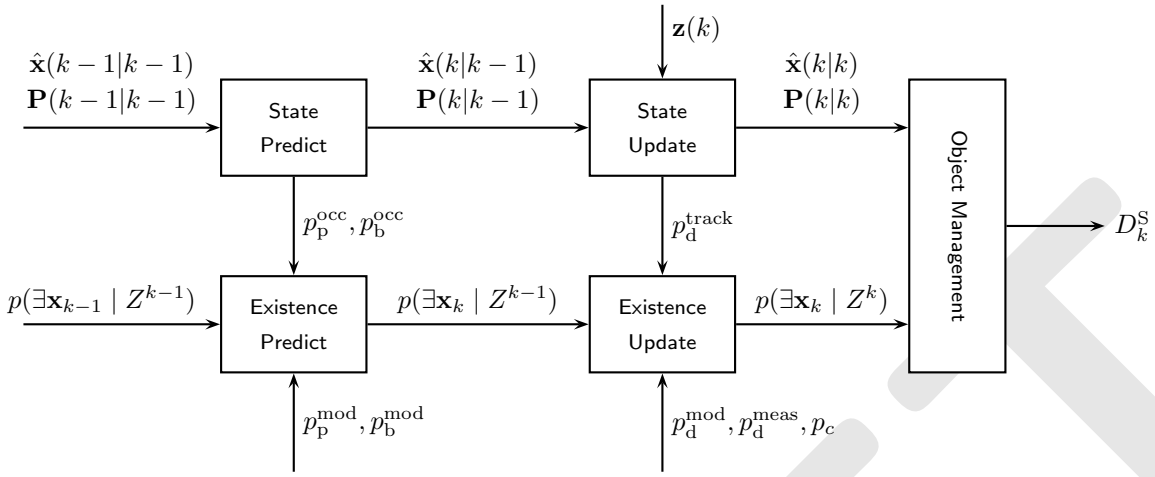


Figure 5.1: Sensor-level processing flow of decoupled existence probability estimation and state estimation.

In this section, a Bayesian approach is used to calculate an object's existence probability in a loosely decoupled fashion from the state estimation, where some aspects of the state estimation are used to improved the existence probability. The approach is kept general in that no parameters are sensor-specific and can be modeled differently for various sensors. An existence probability at time k with measurement information up to time k for an object track $\mathbf{x}(k)$ is denoted as $p(\exists \mathbf{x}_k | Z^k)$, where the probability of non-existence is simply given by the complement $p(\nexists \mathbf{x}_k | Z^k) = 1 - p(\exists \mathbf{x}_k | Z^k)$.

The flow of a Bayesian existence probability estimation, loosely decoupled from the state estimation, is shown in Figure 5.1. Just like in state estimation, an a-priori existence probability is predicted forward in time and then updated depending on the quality of the measurement associated with that particular track or if a measurement was even associated with the track at all. Various parameters, as shown in the figure, influence the prediction and update steps of existence probability estimation. Some of these parameters can be modeled and others come directly from other sources, such as classification or state estimation. The following sections will describe the process of existence probability prediction and updating and how to model or obtain the required parameters.

5.1.1 Existence Prediction

Object existence can be modeled as a two-state Markov process, as presented in [197, 199], where the object at time k can either exist ($\exists \mathbf{x}_k$) or not exist ($\nexists \mathbf{x}_k$). Using the Chapman-Kolmogrov equation, the Markov chain can be transitioned from $k-1$ to k using a probability transition matrix. The probability of object existence is predicted from $k-1$ to k by

$$\begin{bmatrix} p(\exists \mathbf{x}_k \mid Z^{k-1}) \\ p(\nexists \mathbf{x}_k \mid Z^{k-1}) \end{bmatrix} = \begin{bmatrix} p(\exists \mathbf{x} \mid \exists \mathbf{x}) & p(\exists \mathbf{x} \mid \nexists \mathbf{x}) \\ p(\nexists \mathbf{x} \mid \exists \mathbf{x}) & p(\nexists \mathbf{x} \mid \nexists \mathbf{x}) \end{bmatrix} \begin{bmatrix} p(\exists \mathbf{x}_{k-1} \mid Z^{k-1}) \\ p(\nexists \mathbf{x}_{k-1} \mid Z^{k-1}) \end{bmatrix} \tag{5.1}$$

where $p(\exists \mathbf{x} \mid \exists \mathbf{x})$, $p(\exists \mathbf{x} \mid \nexists \mathbf{x})$, $p(\nexists \mathbf{x} \mid \exists \mathbf{x})$ and $p(\nexists \mathbf{x} \mid \nexists \mathbf{x})$ are the state transition probabilities from $k-1$ to k. Substituting $p(\exists \mathbf{x} \mid \exists \mathbf{x}) = p_{\mathrm{p}}(\hat{\mathbf{x}}_{k|k-1})$, $p(\nexists \mathbf{x} \mid \exists \mathbf{x}) = 1 - p_{\mathrm{p}}(\hat{\mathbf{x}}_{k|k-1})$,

$p(\exists \mathbf{x} \mid \nexists \mathbf{x}) = p_{\mathrm{b}}(\mathbf{x}_{k|\hat{k}-1})$ and $p(\nexists \mathbf{x} \mid \nexists \mathbf{x}) = 1 - p_{\mathrm{b}}(\mathbf{x}_{k|\hat{k}-1})$ leads to the evaluation of the following two expressions

$$p(\exists \mathbf{x}_k \mid Z^{k-1}) = p_{\mathrm{p}}(\hat{\mathbf{x}}_{k|k-1}) p(\exists \mathbf{x}_{k-1} \mid Z^{k-1}) + p_{\mathrm{b}}(\hat{\mathbf{x}}_{k|k-1}) p(\nexists \mathbf{x}_{k-1} \mid Z^{k-1}) \tag{5.2}$$

$$p(\nexists \mathbf{x}_k \mid Z^{k-1}) = \left[1 - p_{\mathrm{p}}(\hat{\mathbf{x}}_{k|k-1})\right] p(\exists \mathbf{x}_{k-1} \mid Z^{k-1}) + \left[1 - p_{\mathrm{b}}(\hat{\mathbf{x}}_{k|k-1})\right] p(\nexists \mathbf{x}_{k-1} \mid Z^{k-1}) \tag{5.3}$$

where $p(\exists \mathbf{x}_{k-1} \mid Z^{k-1})$ is the probability that $\mathbf{x}$ exists at $k-1$ given that measurements Z^{k-1} have been integrated, $p(\exists \mathbf{x}_k \mid Z^{k-1})$ is the predicted probability that $\mathbf{x}$ exists at k given that measurements Z^{k-1} have been integrated, $p_{\mathrm{p}}(\hat{\mathbf{x}}_{k|k-1})$ is the persistence probability of the object given the predicted state $\hat{\mathbf{x}}(k|k-1)$ and $p_{\mathrm{b}}(\hat{\mathbf{x}}_{k|k-1})$ is the birth probability of the object given the predicted state $\hat{\mathbf{x}}(k|k-1)$.

The prediction step serves two main purposes for existence probability estimation in practice. First, the persistence probability is usually chosen such that it does not equal 1, guaranteeing that a predicted existence probability is never exactly 1. This models the fact that during prediction, an assumption is made that the object continues to exist, which may not necessarily be true. Second, the existence probability is reduced as an object moves into an area where the persistence probability is low, which allows for proper deletion of tracks that leave the field of view of a sensor. Modeling of the persistence and birth probability are described in Section 5.1.4.

The modeling of existence could also be extended by introducing an extra state in the Markov process which represents that an object exists, but is currently occluded [197, 199]. In the proposed high-level sensor data fusion architecture, the problem of occlusions will be handled at the fusion-level and, therefore, the simple two-state Markov process to model existence is used at the sensor-level.

5.1.2 Existence Update

Existence probability update is achieved using a basic Bayes estimator update formulation, which is based on Bayes rule. For estimation applications, Bayes rule for the update step is usually formulated as follows:

$$p(\theta \mid z) = \frac{p(z \mid \theta)\, p(\theta)}{p(z)} \tag{5.4}$$

where θ is the value to be estimated, $p(\theta)$ is the prior probability of the value, $p(\theta \mid z)$ is the posterior of the value to be estimated after observing a measurement, $p(z \mid \theta)$ is the measurement made from the value to be estimated and $p(z)$ is a normalizing factor. In order to simplify the expression, $p(z)$ is often replaced by a normalizing factor η such that

$$p(\theta \mid z) = \eta\, p(z \mid \theta)\, p(\theta) \tag{5.5}$$

where η ensures that the result of evaluating Bayes rules amongst the value θ and its complement sum to 1.

For existence probability, the value to be estimated, θ and its complement $\overline{\theta}$, is the existence probability, its complement the non-existence and the measurement is the information about the probabilistic nature of the measurement, $\mathbf{z}$, associated with the object, if one was associated at all. From (5.5), this results in

$$p(\theta \mid z) = \eta\, p(z \mid \theta)\, p(\theta) = p(\exists \mathbf{x}_k \mid Z^k) \tag{5.6}$$

$$p(\overline{\theta} \mid z) = \eta\, p(z \mid \overline{\theta})\, p(\overline{\theta}) = p(\nexists \mathbf{x}_k \mid Z^k). \tag{5.7}$$

Measurement Associated

Given that an object is updated with a measurement at k, it is desired to update the existence probability with the quality of the measurement. From expressions (5.6) and (5.7) and given that a measurement $\mathbf{z}_k$ was made and associated with the predicted track $\hat{\mathbf{x}}(k|k-1)$, the existence and non-existence probability update is

$$p(\exists \mathbf{x}_k \mid Z^k) = \eta\, p(\mathbf{z}_k \mid \exists \mathbf{x}_k)\, p(\exists \mathbf{x}_k \mid Z^{k-1}) \tag{5.8}$$
$$p(\nexists \mathbf{x}_k \mid Z^k) = \eta\, p(\mathbf{z}_k \mid \nexists \mathbf{x}_k)\, p(\nexists \mathbf{x}_k \mid Z^{k-1}) \tag{5.9}$$

where $p(\exists \mathbf{x}_k \mid Z^k)$ is the posterior existence probability with information from the measurement set Z^k up to time k, $p(\exists \mathbf{x}_k \mid Z^{k-1})$ is the prior existence probability with the information from the measurement set Z^{k-1} up to time $k-1$, $p(\mathbf{z}_k \mid \exists \mathbf{x}_k)$ is the measurement information about the track at time k and η is the normalizing factor. With $p(\nexists \mathbf{x}_k \mid Z^k)$, the complement of the object's existence, its non-existence, is calculated. The prior existence probability and non-existence, $p(\exists \mathbf{x}_k \mid Z^{k-1})$ and $p(\nexists \mathbf{x}_k \mid Z^{k-1})$, is obtained from the output of the prediction step described in the previous section. The normalizing factor can then be caluculated using

$$\eta = \left[p(\mathbf{z}_k \mid \exists \mathbf{x}_k)\, p(\exists \mathbf{x}_k \mid Z^{k-1}) + p(\mathbf{z}_k \mid \nexists \mathbf{x}_k)\, p(\nexists \mathbf{x}_k \mid Z^{k-1})\right]^{-1} \tag{5.10}$$

and then used to evaluate $p(\exists \mathbf{x}_k \mid Z^k)$ and $p(\nexists \mathbf{x}_k \mid Z^k)$.

The probability that a measurement was made given that the track exists is $p(\mathbf{z}_k \mid \exists \mathbf{x}_k)$, where this probability is proportional to the quality of the measurement associated to the track. This probability can be called the detection probability p_d such that

$$p(\mathbf{z}_k \mid \exists \mathbf{x}_k) = p_\mathrm{d}(k). \tag{5.11}$$

Similarly, $p(\mathbf{z}_k \mid \nexists \mathbf{x}_k)$ is the probability that a measurement was made, even though the track does not exist. This probability can be called the clutter probability p_c, or false positive probability, such that

$$p(\mathbf{z}_k \mid \nexists \mathbf{x}_k) = p_\mathrm{c}(k). \tag{5.12}$$

Rewriting (5.8) and (5.9) with the above definitions yields the final expressions for updating the existence and non-existence probability if a measurement was associated

$$p(\exists \mathbf{x}_k \mid Z^k) = \eta\, p_\mathrm{d}(k)\, p(\exists \mathbf{x}_k \mid Z^{k-1}) \tag{5.13}$$
$$p(\nexists \mathbf{x}_k \mid Z^k) = \eta\, p_\mathrm{c}(k)\, p(\nexists \mathbf{x}_k \mid Z^{k-1}). \tag{5.14}$$

Examples of how to model the parameters $p_\mathrm{d}(k)$ and $p_\mathrm{c}(k)$ is described in Section 5.1.4.

No Measurement Associated

If a measurement was *not* associated with a track at k, then a similar Bayes formulation as from (5.6) and (5.7) is used to update the existence and non-existence of the track with

$$p(\exists \mathbf{x}_k \mid Z^k) = \eta\, p(\overline{\mathbf{z}}_k \mid \exists \mathbf{x}_k)\, p(\exists \mathbf{x}_k \mid Z^{k-1}) \tag{5.15}$$
$$p(\nexists \mathbf{x}_k \mid Z^k) = \eta\, p(\overline{\mathbf{z}}_k \mid \nexists \mathbf{x}_k)\, p(\nexists \mathbf{x}_k \mid Z^{k-1}) \tag{5.16}$$

where $\overline{\mathbf{z}}_k$ represents the probability that no measurement was associated with the object. The normalizing factor in this case is calculated with

$$\eta = \left[p(\overline{\mathbf{z}}_k \mid \exists \mathbf{x}_k)\, p(\exists \mathbf{x}_k \mid Z^{k-1}) + p(\overline{\mathbf{z}}_k \mid \nexists \mathbf{x}_k)\, p(\nexists \mathbf{x}_k \mid Z^{k-1})\right]^{-1} \tag{5.17}$$

The probability that a measurement was not associated conditioned on whether the object exists or does not exist is the complement of the definitions in (5.11) and (5.12) such that

$$p(\overline{\mathbf{z}}_k \mid \exists \mathbf{x}_k) = 1 - p_{\mathrm{d}}(k). \tag{5.18}$$

and

$$p(\overline{\mathbf{z}}_k \mid \nexists \mathbf{x}_k) = 1 - p_{\mathrm{c}}(k). \tag{5.19}$$

Similarly rewriting (5.15) and (5.16) with the above definitions yields the final expressions for updating the existence and non-existence probability if a measurement was not associated

$$p(\exists \mathbf{x}_k \mid Z^k) = \eta\, [1 - p_{\mathrm{d}}(k)]\, p(\exists \mathbf{x}_k \mid Z^{k-1}) \tag{5.20}$$

$$p(\nexists \mathbf{x}_k \mid Z^k) = \eta\, [1 - p_{\mathrm{c}}(k)]\, p(\nexists \mathbf{x}_k \mid Z^{k-1}). \tag{5.21}$$

Again, examples of how to model the parameters $p_{\mathrm{d}}(k)$ and $p_{\mathrm{c}}(k)$ is described in Section 5.1.4.

5.1.3 Generalized Bayes Extension

Traditional Bayes estimation is formulated such that the value to be estimated is updated with only the newest measurement information. Under the Markov assumption, the information from previous measurements is inherently contained in the prior of the value to be estimated. In [3], a generalized Bayes estimation method was introduced, which loosens the restriction that only the newest measurement is used in the update process. With the generalized Bayes formulation, the measurement update can consist up to several consecutive previous measurements, resulting in a filtering effect of the Bayes update.

The update formulation for existence and non-existence probability if a measurement was associated from (5.8) and (5.9) can be extended to a generalized Bayes formulation with

$$p(\exists \mathbf{x}_k \mid Z^k) = \eta\, p(\mathbf{z}_k \mid Z_{k-L}^{k-1}, \exists \mathbf{x}_k)\, p(\exists \mathbf{x}_k \mid Z^{k-1}) \tag{5.22}$$

$$p(\nexists \mathbf{x}_k \mid Z^k) = \eta\, p(\mathbf{z}_k \mid Z_{k-L}^{k-1}, \nexists \mathbf{x}_k)\, p(\nexists \mathbf{x}_k \mid Z^{k-1}) \tag{5.23}$$

where $p(\mathbf{z}_k \mid Z_{k-L}^{k-1}, \exists \mathbf{x}_k)$ and $p(\mathbf{z}_k \mid Z_{k-L}^{k-1}, \nexists \mathbf{x}_k)$ is the measurement update information between time k and $k-L$, where L is the number previous measurements to consider. A simple method for calculating $p(\mathbf{z}_k \mid Z_{k-L}^{k-1}, \exists \mathbf{x}_k)$ and $p(\mathbf{z}_k \mid Z_{k-L}^{k-1}, \nexists \mathbf{x}_k)$ is to simply take the average of the current measurement information and the previous L measurements such that

$$p(\mathbf{z}_k \mid Z_{k-L}^{k-1}, \exists \mathbf{x}_k) = \frac{1}{L+1} \sum_{i=k-L}^{k} p_{\mathrm{d}}(i) \tag{5.24}$$

and

$$p(\mathbf{z}_k \mid Z_{k-L}^{k-1}, \exists \mathbf{x}_k) = \frac{1}{L+1} \sum_{i=k-L}^{k} p_{\mathrm{c}}(i) \tag{5.25}$$

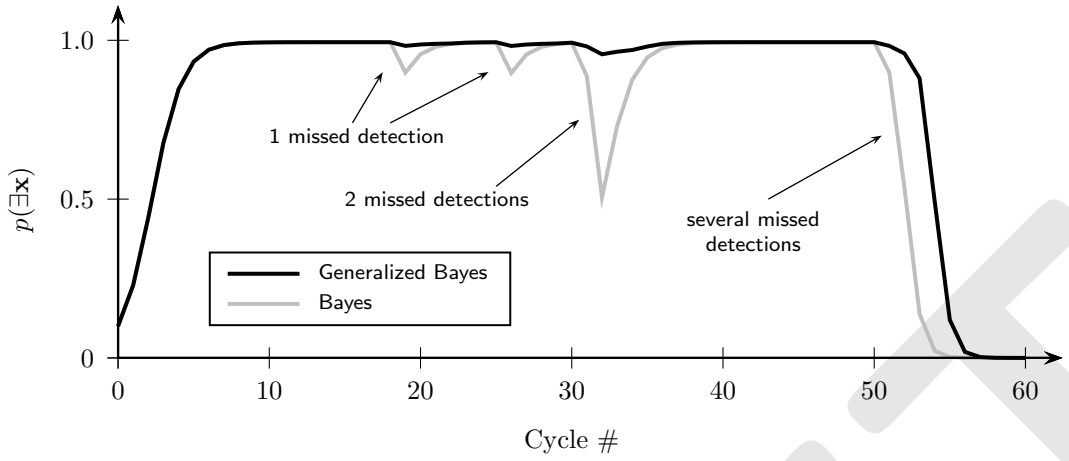


Figure 5.2: Comparison of the generalized Bayes and Bayes update method for existence probability calculation. Generalized Bayes is less sensitive to sporadic missed detections.

where $p_\mathrm{d}(i)$ is the detection probability and $p_\mathrm{c}(i)$ the clutter probability at time i of the summation.

The expressions (5.15) and (5.16) for when a measurement is not associated with the track can be similarly extended such that

$$p(\exists \mathbf{x}_k \mid Z^k) = \eta\, p(\overline{\mathbf{z}}_k \mid Z_{k-L}^{k-1}, \exists \mathbf{x}_k)\, p(\exists \mathbf{x}_k \mid Z^{k-1}) \tag{5.26}$$

$$p(\nexists \mathbf{x}_k \mid Z^k) = \eta\, p(\overline{\mathbf{z}}_k \mid Z_{k-L}^{k-1}, \nexists \mathbf{x}_k)\, p(\nexists \mathbf{x}_k \mid Z^{k-1}) \tag{5.27}$$

where the measurement probabilities are again a summation of the complement of the detection probability

$$p(\overline{\mathbf{z}}_k \mid Z_{k-L}^{k-1}, \exists \mathbf{x}_k) = \frac{1}{L+1} \sum_{i=k-L}^{k} 1 - p_\mathrm{d}(i) \tag{5.28}$$

and clutter probability

$$p(\overline{\mathbf{z}}_k \mid Z_{k-L}^{k-1}, \exists \mathbf{x}_k) = \frac{1}{L+1} \sum_{i=k-L}^{k} 1 - p_\mathrm{c}(i). \tag{5.29}$$

An example of the generalized Bayes update method compared to the normal Bayes update formulation is demonstrated in Figure 5.2. In this example, constant parameters of $p_\mathrm{d}(k) = 0.9$ and $p_\mathrm{c}(k) = 0.3$ were chosen where the window size for the generalized Bayes method was $L = 3$. It can be seen that the generalized Bayes method is far less sensitive to sporadic missed detections. A window size of $L = 3$ seems to be a good compromise between the delay in the existence probability when there are several missed detections and for dealing with sporadic missed detections.

5.1.4 Modeling the Parameters

The previous sections described the predict and update steps for calculating an existence probability. For each of these steps, parameters such as the persistence probability, birth probability, detection probability and clutter probability were mentioned. This section will describe methods for determining these parameters.

Persistence Probability

The persistence probability, p_p is used in the predict step of the process for estimating the existence probability, as described in Section 5.1.1. For aerospace radar applications, as was the case where IPDA and JIPDA were introduced [197, 199], the persistence probability in the Markov process was chosen as a constant, since aerial ground radars have a 360° field of view. In automotive applications, the field of view of a single sensor is much more restricted and therefore the persistence probability can be modeled on the specific viewing properties of the sensor.

The concept of using the sensor's field of view to model the persistence probability in automotive applications was introduced in [169]. The persistence probability has also sometimes been called the survival probability [172]. The total persistence probability consists of a combination of the persistence probability in the polar coordinates

$$p_\mathrm{p}^\mathrm{mod}(r,\phi) = p_\mathrm{p}(r) \cdot p_\mathrm{p}(\phi) \tag{5.30}$$

where the range, r, is in the interval $[r_\mathrm{min}, r_\mathrm{max}]$ and the angle, ϕ, is in the interval $[-\phi_\mathrm{max}, \phi_\mathrm{max}]$ where the minimum and maximum range and angle are given by the sensor's field of view. The persistence probability for a predicted track $\hat{\mathbf{x}}(k|k-1)$ is denoted as $p_\mathrm{p}(\hat{\mathbf{x}}_{k|k-1})$, where r and ϕ are obtained from the position of the state vector.

The persistence probability for the range is modeled as a four-part piecewise function:

$$p_\mathrm{p}(r) = \begin{cases} 0 & r < r_\mathrm{min} \\ p_\mathrm{p}^\mathrm{max} & r_\mathrm{min} \leq r < r_\mathrm{max}(1-m_r) \\ p_\mathrm{p}^\mathrm{max} \cdot \left(e^{\frac{-\log(\alpha)(r-r_\mathrm{max})}{r_\mathrm{max}-r_\mathrm{max}(1-m_r)}} + 1 \right) & r_\mathrm{max}(1-m_r) \leq r \leq r_\mathrm{max} \\ 0 & r > r_\mathrm{max} \end{cases} \tag{5.31}$$

From the minimum range up to a m_r percent part of the maximum range, denoted as the cut-off range (obtained with $r_\mathrm{max}(1-m_r)$), the persistence probability is constant at a maximum persistence probability, $p_\mathrm{p}^\mathrm{max}$. After the cut-off range and up to the maximum range of the sensor, the persistence probability is modeled as a falling inverted exponential, where α is a factor that defines the steepness of the exponential. Similarly, the persistence probability of the angle is modeled as a five-part piecewise function:

$$p_\mathrm{p}(\phi) = \begin{cases} 0 & \phi < -\phi_\mathrm{max} \\ p_\mathrm{p}^\mathrm{max} \cdot \left(e^{\frac{-\log(\alpha)(\phi+\phi_\mathrm{max})}{\phi_\mathrm{max}(1-m_\phi)-\phi_\mathrm{max}}} + 1 \right) & -\phi_\mathrm{max} \leq \phi \leq -\phi_\mathrm{max}(1-m_\phi) \\ p_\mathrm{p}^\mathrm{max} & -\phi_\mathrm{max}(1-m_\phi) < \phi < \phi_\mathrm{max}(1-m_\phi) \\ p_\mathrm{p}^\mathrm{max} \cdot \left(e^{\frac{-\log(\alpha)(\phi-\phi_\mathrm{max})}{\phi_\mathrm{max}-\phi_\mathrm{max}(1-m_\phi)}} + 1 \right) & \phi_\mathrm{max}(1-m_\phi) \leq \phi \leq \phi_\mathrm{max} \\ 0 & \phi > \phi_\mathrm{max} \end{cases} \tag{5.32}$$

where ϕ_max is the maximum viewing angle of the sensor in the positive and negative direction of the sensor forward axis. As with the range, an exponential is used to model the persistence probability on the edges of the field of view up to a m_ϕ percent part of the maximum angle (obtained with $\phi_\mathrm{max}(1-m_\phi)$).

Figure 5.3 shows an example of how the persistence probability can be modeled for a forward facing sensor with a maximum range of 200 m and a maximum angle of ±50° along with a two-dimensional visualization of the persistence probability after the polar parts are multiplied together, as in (5.30). The cut-off range and angle were chose as $m_r = 0.4$ and $m_\phi = 0.3$, respectively, and α was chosen to be 0.01.

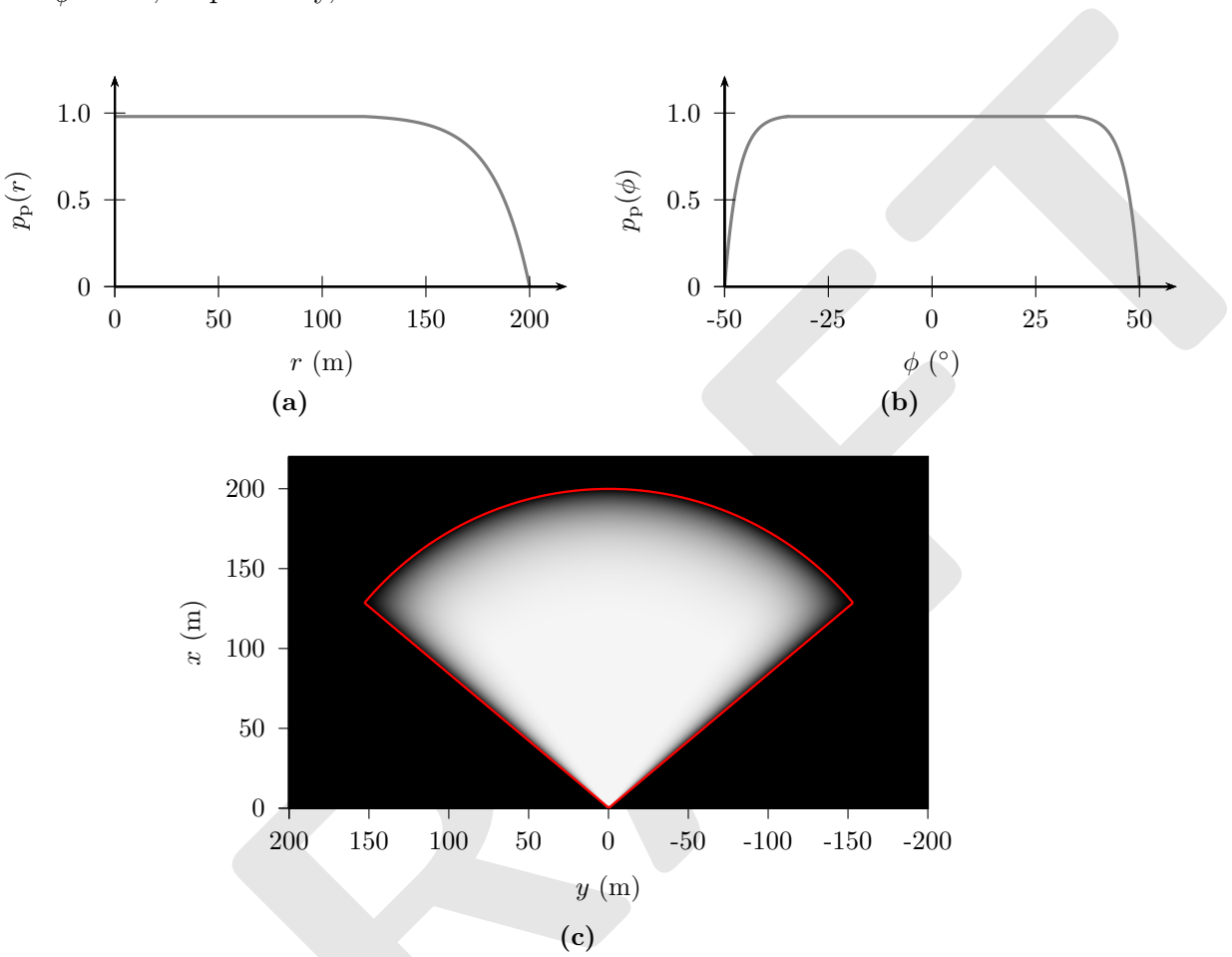


Figure 5.3: Modeling of the persistence probability in polar coordinates of a sensor with a maximum range of 200 m and a maximum angle of ±50°, where (a) shows the modeling of the range, r, (b) shows the modeling of the angle ϕ, and (c) visualizes the combined persistence probability in a Cartesian coordinate system.

The persistence probability can be extended with occlusion information obtained from the predicted object list, $\mathcal{O}^\mathrm{S}(k|k-1)$. Similar to [169], the modeled persistence probability based on the field of view is multiplied with the occlusion persistence probability, $p_\mathrm{p}^\mathrm{occ}(\mathcal{O}^\mathrm{S}(k|k-1))$,

$$p_\mathrm{p}(\hat{\mathbf{x}}_{k|k-1}) = p_\mathrm{p}^\mathrm{mod}(r,\phi) \cdot p_\mathrm{p}^\mathrm{occ}(\mathcal{O}^\mathrm{S}(k|k-1)) \tag{5.33}$$

where the persistence probability in $p_\mathrm{p}^\mathrm{occ}(\mathcal{O}^\mathrm{S}(k|k-1))$ is 0 for locations which are in the "shadow" of an object from $\mathcal{O}^\mathrm{S}(k|k-1)$. Note that using the occlusion persistence probability only makes sense for line-of-sight sensors, such as laser scanners, which are not able detect occluded objects. A sensor which can detect or partially detect occluded objects, such as a radar, should not use the occlusion persistence probability or may need to derive a more complex model representing the sensor's capabilities.

Birth Probability

The birth probability $p_{\mathrm{b}}(\mathbf{x}_{k|\hat{k-1}})$ was introduced in Section 5.1.1 for the existence prediction. Additionally, the birth probability is used to initialize the existence probability for a newly detected object from an unassigned measurement in the environment.

The most simply method for modeling the birth probability is to assume a reasonable constant value, for example $p_{\mathrm{b}}(\mathbf{x}_{k|\hat{k-1}}) = 0.1$. Such a constant should be above the deletion threshold (see Section 5.1.5), but low enough such that the object's existence can be verified over time as the object continues to be observed with measurements. Choosing a constant birth probability is also the most typical approach used in the literature when used in conjunction with IPDA and Joint Probabilistic Data Association (JPDA) [197, 199].

In [169], knowledge about the sensor's field of view and other detected objects is used to determine a more precise birth probability. The birth probability is modeled by taking the gradient of the persistence probability such that at locations where the persistence probability changes is proportional to a higher birth probability. This results in the effect that a high birth probability is chosen at the edges of the sensor's field of view and at the edges of object occlusion.

The birth probability is approached differently in [196], where the assumption is made that new objects cannot be created in the near vicinity of already detected objects with a high existence probability. The idea of a Probability Hypothesis Density (PHD) function are used to derive the probability that an object over a specific region exists. The complement of this probability overall all objects in the environment results in a spatial birth probability for any new object hypothesis.

As shown, there are many options for modeling a birth probability. The various options and their simplicity or complexity should be considered when choosing a birth probability model. The type of sensor being used may also influence the choice. For example, the model with the PHD function presented in [196] works well for sensors which can detect the extend of an object's dimensions. Other sensor, such as a camera, may rely more on a polar model of birth probability, similar to the model presented in [169]. But the simplest method of choosing a constant value can also be effective enough if the rest of the existence probability problem is well formulated.

Detection Probability

From (5.11) and (5.12), the detection probability, $p_{\mathrm{d}}(k)$, represents the probability whether or not a valid measurement for a relevant object was detected and associated. The detection probability influences how fast the existence probability rises and falls when measurements are associated with an object. There are many ways to model this probability and can vary depending on the sensor type. In [196], the detection probability for a laser scanner is modeled by the vehicle's pitch and detected object's z coordinate, which reduces the detection probability when the laser scanner's measurement layers are above or below the detectable object. The detection probability of a camera sensor is also modeled in [196] by using the camera's field of view, detected object's position, length and width and the object's orientation. The detection probability could also be derived directly from the result of a classifier, such as Adaboost clusters for a camera [169]. The simplest solution for modeling the detection probability would be simply to choose an appropriate constant.

A more generic approach is presented here, where the detection probability is a combi-

nation of three values: the modeled detection probability $p_{\mathrm{d}}^{\mathrm{mod}}(\hat{\mathbf{x}}_{k|k-1})$, measurement true positive probability $p_{\mathrm{d}}^{\mathrm{meas}}(\mathbf{z}_k)$ and the track probability $p_{\mathrm{d}}^{\mathrm{track}}(\hat{\mathbf{x}}_{k|k})$:

$$p_{\mathrm{d}}(k) = p_{\mathrm{d}}^{\mathrm{mod}}(\hat{\mathbf{x}}_{k|k-1}) \cdot p_{\mathrm{d}}^{\mathrm{meas}}(\mathbf{z}_k) \cdot p_{\mathrm{d}}^{\mathrm{track}}(\hat{\mathbf{x}}_{k|k}). \tag{5.34}$$

If a measurement was not associated with an object, then $p_{\mathrm{d}}^{\mathrm{meas}}(\mathbf{z}_k)$ and $p_{\mathrm{d}}^{\mathrm{track}}(\hat{\mathbf{x}}_{k|k})$ are simply omitted.

The modeled detection probability $p_{\mathrm{d}}^{\mathrm{mod}}(\hat{\mathbf{x}}_{k|k-1})$ depends solely on the predict state of the track, therefore no measurement information is considered. The idea is to model the fact that given any location in the sensor's field of view, what is the probability that the sensor it is able to produce a valid measurement at that location if an object were to exist. The simplest method is to simply assume that the sensor can detect an object everywhere if it exists, resulting in a constant value for $p_{\mathrm{d}}^{\mathrm{mod}}(\hat{\mathbf{x}}_{k|k-1})$. Such an assumption may suffice in many applications. A similar model which considers the sensor's field-of-view, as done with the persistence probability, can improve the model. Other sensor properties, such as an a-priori spatial signal-to-noise ratio could also be considered, particularly for radar sensors. The host vehicle's state, as done in [196], can also be considered for improving the model.

The measurement true positive probability, $p_{\mathrm{d}}^{\mathrm{meas}}(\mathbf{z}_k)$, is a measure of the quality of the new measurement, $\mathbf{z}_k$, associated with the object. It represents how well a specific measurement could represent a relevant object in the environment. For automotive applications, relevant objects usually include cars, trucks, motorcycles, road infrastructure, etc. Such relevant objects are usually determined using a classifier on the measurement, as will be described in Chapter 6. One possible generic model for $p_{\mathrm{d}}^{\mathrm{meas}}(\mathbf{z}_k)$ is to simply use the classification result as a representation of the measurement's quality, where any classification, other than the class "Other" which may not represent a relevant object, signals a strong measurement:

$$p_{\mathrm{d}}^{\mathrm{meas}}(\mathbf{z}_k) = 1 - C_{\mathrm{Other}}. \tag{5.35}$$

Modeling the measurement true positive probability in this manner makes the existence probability update sensor independent, since the classification algorithms take into account the sensor-specific information in order to determine the classification likelihoods. However, a more sensor-specific model could also be used, for example using classifier result from an image processing algorithm for camera sensors or using a function dependant on the radar cross-section of an object for radar sensors.

The track probability, $p_{\mathrm{d}}^{\mathrm{track}}(\hat{\mathbf{x}}_{k|k})$, represents the quality of the state estimate of the track. This ensures that objects which are being tracked in an unstable fashion to not contribute strongly to the detection probability, as an unstable track is usually a sign of a false track. The measure for track state estimate quality used is the Average Normalized Innovation Squared (ANIS) [25]. The Normalized Innovation Squared (NIS) is a measure of how well the innovation, $\tilde{\mathbf{z}}(k)$, of the new measurement to the predicted track is consistent with the innovation covariance, $\mathbf{S}(k)$, at time k and is given by

$$\epsilon_{\tilde{\mathbf{z}}}(k) = \tilde{\mathbf{z}}(k)' \mathbf{S}(k)^{-1} \tilde{\mathbf{z}}(k). \tag{5.36}$$

The NIS is χ^2 distributed with n_{z} degrees of freedom, where n_{z} is the dimension of the measurement space. For a more stable measure of the state estimate's quality over time,

the NIS is averaged over the K previous measurement updates in order to obtain the ANIS:

$$\bar{\epsilon}_{\tilde{\mathbf{z}}}(k) = \frac{1}{K} \sum_{i=0}^{K-1} \tilde{\mathbf{z}}(k-i)' \mathbf{S}(k-i)^{-1} \tilde{\mathbf{z}}(k-i). \tag{5.37}$$

The ANIS, $\bar{\epsilon}_{\tilde{\mathbf{z}}}(k)$, is χ^2 distributed with $K \cdot n_\mathrm{z}$ degrees of freedom. Using the inverse χ^2 distribution, a probability is derived which represents the object's state estimate quality:

$$p_\mathrm{d}^\mathrm{track}(\hat{\mathbf{x}}_{k|k}) = 1 - P\{\chi^2(x; Kn_\mathrm{z}) \leq K\bar{\epsilon}_{\tilde{\mathbf{z}}}(k)\} \tag{5.38}$$

The larger the inconsistency of the track from the ANIS, the smaller the track detection probability.

In combination, the modeled, measurement and track detection probability have the effect of rising the existence probability either more quickly or slowly, depending on the location and quality of the measurement associated with an object.

Clutter Probability

The clutter probability is often modeled as a spatial Poisson process in tracking applications [147, 159, 169]. It represents the probability that a false measurement occurs in a given area or during a given time frame. The basic Poisson process for the probability that exactly m measurements are false measurements at time k, denoted as the set of false measurements Z_k^F, is

$$p_c(|Z_k^\mathrm{F}| = m; \lambda) = \frac{\lambda^m e^{-\lambda}}{m!} \tag{5.39}$$

where λ is the rate parameter of the Poisson process. The rate parameter, depending on the application of the Poisson process, can be defined as the unit occurrence of an event, in this case a false measurement, with respect to either time or space. For the probability that up to and including m false measurements have occurred is given by the sum

$$p_c(|Z_k^\mathrm{F}| \leq m; \lambda) = \sum_{i}^{|Z_k^\mathrm{F}|} \frac{\lambda^{m_i} e^{-\lambda}}{m_i!}. \tag{5.40}$$

In order to apply (5.40), it is necessary to estimate the number of potential false measurements $|Z_k^\mathrm{F}|$ and the rate parameter λ.

The estimated number of false measurements, $\hat{N}_Z = |\hat{Z}_k^\mathrm{F}|$, can be calculated using

$$\hat{N}_Z(k) = |\hat{Z}_k^\mathrm{F}| = |Z_k| - \hat{N}_\mathrm{A}(k) - \hat{N}_\mathrm{B}(k) \tag{5.41}$$

where $|Z_k|$ is the total number of measurements at time k, $\hat{N}_\mathrm{A}(k)$ is the estimated number of measurements successfully assigned to an object at time k and $\hat{N}_\mathrm{B}(k)$ is the estimated number of potentially newly valid (birth) objects from the measurements at time k. The estimated number of successful measurement-to-object assignments, $\hat{N}_\mathrm{A}(k)$, can be determined by combining the predicted existence probability of the object, $p(\exists \mathbf{x}_k \mid Z^{k-1})$, and the measurement true positive probability, $p_\mathrm{d}^\mathrm{meas}(\mathbf{z}_k)$, for all successful measurement-to-object assignments, such that

$$\hat{N}_\mathrm{A}(k) = \sum_{i,j}^{\forall \mathbf{A}_{i,j}} p(\exists \mathbf{x}_k \mid Z^{k-1})_i \cdot p_\mathrm{d}^\mathrm{meas}(\mathbf{z}_k)_j \tag{5.42}$$

where $\forall \mathbf{A}_{i,j}$ are the elements of the association matrix where a successful measurement-to-object association was made. The number of potentially new and valid objects, or birth objects, are given by

$$\hat{N}_{\mathrm{B}}(k) = \sum_{i}^{\forall \mathbf{z}_k \notin \mathbf{A}_{i,j}} p_{\mathrm{b}}(\mathbf{z}_k)_i \tag{5.43}$$

where $\forall \mathbf{z}_k \notin \mathbf{A}_{i,j}$ are the measurements at time k which were not assigned to an object and $p_{\mathrm{b}}(\mathbf{z}_k)$ is the birth probability as a new potential object of the measurement.

The rate parameter λ is usually assumed constant and is known a-priori for a sensor and is usually determined experminentally. This assumption may be valid for radar applications in the aerospace industry, where such Poisson processes were first applied in tracking applications. In automotive applications, however, sensors are quite sensitive to their current environment. The rate at which false measurements are made are dependent on the weather, speed of the host vehicle and many other factors. Due to these reasons, it is more practical to make an online calculation of an estimated rate parameter $\hat{\lambda}(k)$ at every new time step k. In order for the estimated rate parameter to adapt to a changing environment, it is estimated recursively with

$$\hat{\lambda}(k) = (1 - w)\hat{\lambda} + w\hat{\lambda}(k - 1) \tag{5.44}$$

where $\hat{\lambda}$ is the current estimated rate parameter, $\hat{\lambda}(k-1)$ is the estimated rate parameter from the previous time step $k - 1$ and w is a recursive weight factor. A simple method for choosing the current rate parameter is to model it as the number of potential false measurement per time step such that

$$\hat{\lambda} = \hat{N}_Z(k). \tag{5.45}$$

In this case, the rate parameter in the Poisson process models the number of occurrences (the false measurements) per unit time (a single iteration). With the recursive estimation, the rate parameter should converge to reflect the current state of the vehicle's environment.

5.1.5 Object Management

Object management is achieved by thresholding the existence probability of an object. In general, three threshold levels can be defined where an action on the object occurs: confirmation threshold (τ_{c}), unconfirmation threshold (τ_{uc}) and the deletion threshold (τ_{d}). When an object's existence probability reaches the confirmation threshold τ_{c}, it is promoted to a validated object and is including in the list of objects which the sensor delivers as its output. As an object's existence probability falls below the unconfirmation threshold τ_{uc}, then the object is not considered to be a validated object anymore, but it is still kept in the sensor's internal object list. Note that the unconfirmation threshold should be chosen such that $\tau_{\mathrm{uc}} < \tau_{\mathrm{c}}$ which has the effect of keeping object validated for a longer period of time once they have reached τ_{c} sometime in their history. As an object's existence probability falls further, it will eventually reach the deletion threshold τ_{d}, after which t is completely deleted from the sensor's object list. Figure 5.4 visualizes the relations between the various existence threshold.

The thresholds should be chosen such that the detection rate (true positive and false positive rates) at the thresholds guarantee certain desired level. It is also possible to

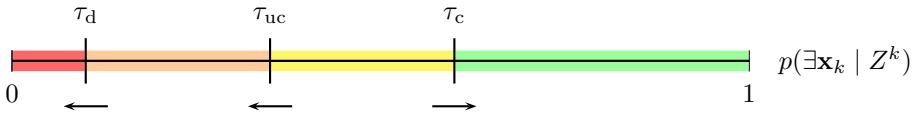


Figure 5.4: Existence thresholds for confirming and deleting objects.

define several confirmation threshold, each corresponding to a different desired detection rate such that existence probability between confirmation threshold correspond to a logical reliability of an object. Such a logical reliability can interpreted with respect to a driver assistance application, for example, a safety system such as an emergency braking system with high decelerations should only react to objects that meets the highest confirmation threshold, whereas a comfort system such as active cruise control would react earlier to objects which meet a lower confirmation threshold.

5.2 Fusion

At the fusion-level, it is necessary to fuse the existence probability from several sensors into the global object list using a sensor-to-global fusion strategy, as described in Section 3.2.2. This section describes a novel approach to existence probability fusion at the object-level using Dempster-Shafer evidence theory.

5.2.1 Architecture

The object model interface treats the existence probability as a single probability, typically derived from a Bayesian estimation algorithm, as described for the sensor-level in Section 5.1, or from an IPDA/JIPDA algorithm [197, 199]. However, in order to take advantage of different sensors' performance and to deal with complex situations, such as occlusions, a single value for modeling the existence probability at the fusion-level does not suffice. Therefore, the existence probability is modeled using Dempster-Shafer evidence theory within the fusion module, as will be described in the following sections, and then converted back to a single existence probability value at the output of the fusion module. The processing flow for existence probability fusion is shown in Figure 5.5.

5.2.2 Modeling with Dempster-Shafer Evidence Theory

Modeling and fusing existence using Dempster-Shafer Evidence Theory (DST) was described in [310] for a sensor-to-sensor fusion strategy. In this section, the same method is used, but in a sensor-to-global fusion strategy, where existence information from sensor-level objects is fused into a global object at the fusion-level. For a detailed introduction and overview of the Dempster-Shafer evidence theory and its many applications, refer to [149, 223, 242].

The DST defines a frame of discernment, Θ, which consists of mutually exclusive hypotheses, or states, of a system. For modeling existence, the most simple set of mutually exclusive hypotheses are that an object exists, $\exists$, or does not exist, $\nexists$, such that

$$\Theta = \{\exists, \nexists\}. \tag{5.46}$$

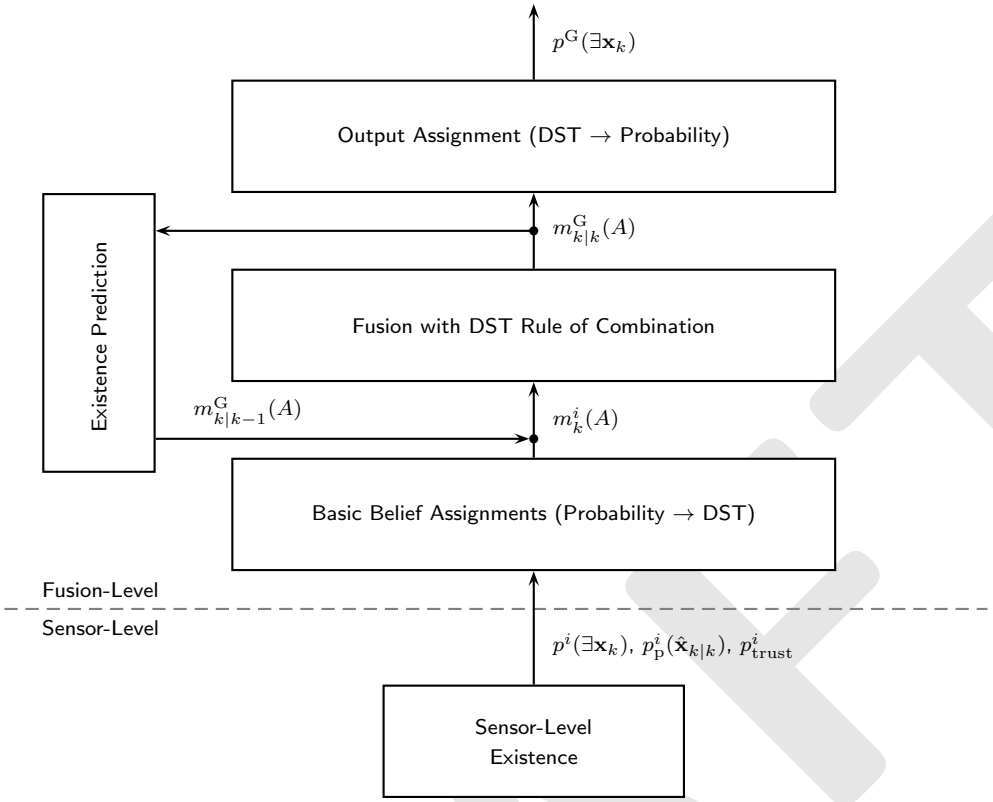


Figure 5.5: Fusion-level processing flow of fusing existence probabilities from several sensors using Dempster-Shafer evidence theory.

DST then defines the power set, 2^{Θ}, which is the set of all of the subsets of Θ, including the empty set $\varnothing$. For modeling existence, as defined by (5.46), this results in

$$2^{\Theta} = \{\varnothing, \{\exists\}, \{\nexists\}, \{\exists, \nexists\}\}. \tag{5.47}$$

The power set contains all of the combinations of Θ, which allows for an assignment of belief, or evidence, to not just the mutually exclusive hypotheses themselves, but also their combinations. This can be exploited to model ignorance, or uncertainty, in evidence and is quite useful when fusing data from several sensors which may have different performance. Such a modeling of uncertainty in evidence is not easily possible in traditional Bayesian inference methods. The power set for object existence includes the subset $\{\exists, \nexists\}$, which allows for an assignment of belief that information about an object's existence is unknown. For each set in the power set, a Basic Belief Assignment (BBA), often also called a mass function, can be made, where

$$m : 2^{\Theta} \to [0, 1]\,. \tag{5.48}$$

This BBA, or mass function, represents the amount of evidence that a given element in 2^{Θ} is believed to be true. If BBAs are made only to the mutually exclusive hypotheses of Θ, then DST is equivalent to traditional Bayesian methods. The assignments of BBAs over the power set must be normalized:

$$\sum_{A:A\subseteq\Theta} m(A) = 1. \tag{5.49}$$

Furthermore, DST defines a belief function

$$\mathrm{Bel}(A) = \sum_{B:B\subseteq A} m(B) \tag{5.50}$$

which contains the BBAs of all of the subsets of A, which can be interpreted as a lower-bound probability of the evidence for A. The upper-bound probability for the evidence in A is defined as the plausibility

$$\mathrm{Pl}(A) = \sum_{B:B\cap A\neq\varnothing} m(B) \tag{5.51}$$

where $\mathrm{Pl}(A)$ contains all of the sets in 2^Θ that support the belief in A. The difference between the belief and plausibility represents the amount of uncertainty in the evidence for A.

Given the BBAs for the power set from two different sources, DST defines a rule of combination that combines the BBAs. In the framework of object existence fusion using a sensor-to-global fusion strategy, one source is the global object, G, and the other source the object existence probability from sensor i, then the DST rule of combination for obtaining the new global object existence with the information from sensor i is

$$m^{\mathrm{G}}_{k|k}(C) = \frac{\sum_{A\cap B=C} m^{\mathrm{G}}_{k|k-1}(A)\, m^i_k(B)}{1-\sum_{A\cap B=\varnothing} m^{\mathrm{G}}_{k|k-1}(A)\, m^i_k(B)}. \tag{5.52}$$

For calculating the DST combination of a BBA, it is useful to evaluate the table of intersections in order to determine the expressions $A\cap B = C$ and $A\cap B = \varnothing$ in (5.52). The table of intersections for modeling existence with the power set from (5.47) is shown in Table 5.1. Each time object data from a sensor arrives at the fusion-level, each global object's BBAs are updated using (5.52) either with the fact that an object was associated to the global track or that no object was associated. The modeling of the BBAs from a sensor i for updating a global object G will be described in the following.

Table 5.1: Intersections of the power set for existence, useful for evaluating DST combinations.

$\cap$	$\{\exists\}$	$\{\nexists\}$	$\{\exists,\nexists\}$
$\{\exists\}$	$\{\exists\}$	$\varnothing$	$\{\exists\}$
$\{\nexists\}$	$\varnothing$	$\{\nexists\}$	$\{\nexists\}$
$\{\exists,\nexists\}$	$\{\exists\}$	$\{\nexists\}$	$\{\exists,\nexists\}$

Existence Prediction

Before fusion, existence prediction is carried out in a similar fashion as done at the sensor-level (Section 5.1.1). At the fusion-level, however, existence prediction is carried out within the DST framework. This is achieved by transferring evidence from the mutually exclusive elements in 2^Θ to the existence uncertainty element $\{\exists,\nexists\}$. Given a prediction weight of

γ, a percentage of the evidence for $\{\exists\}$ and $\{\nexists\}$ is transferred to the existence uncertainty element $\{\exists, \nexists\}$ using

$$m^{\mathrm{G}}_{k|k-1}(\{\exists\}) = (1-\gamma)\, m^{\mathrm{G}}_{k-1|k-1}(\{\exists\}) \tag{5.53}$$

$$m^{\mathrm{G}}_{k|k-1}(\{\nexists\}) = (1-\gamma)\, m^{\mathrm{G}}_{k-1|k-1}(\{\nexists\}) \tag{5.54}$$

$$m^{\mathrm{G}}_{k|k-1}(\{\exists, \nexists\}) = m^{\mathrm{G}}_{k|k-1}(\{\exists, \nexists\}) + \gamma\, (m^{\mathrm{G}}_{k-1|k-1}(\{\exists\}) + m^{\mathrm{G}}_{k-1|k-1}(\{\nexists\})). \tag{5.55}$$

The prediction weight γ can be chosen to be constant, but is more appropriately modeled as a function of time, where longer prediction steps result in a larger prediction weight, thereby modeling the greater uncertainty in the existence of an object during longer prediction. This approach is similar to the discounting method for aging evidence, as described in [242].

Object Associated

If an object from sensor i is associated to a global object G, then more evidence should be assigned to the fact that the object exists. The assignments of the BBAs for sensor i should reflect this fact. The BBA for object existence is the weighted probability of existence from sensor i

$$m^i_k(\{\exists\}) = p^i_{\mathrm{p}}(\hat{\mathbf{x}}_{k|k}) \cdot p^i_{\mathrm{trust}} \cdot p^i(\exists \mathbf{x}_k) \tag{5.56}$$

where $p^i_{\mathrm{p}}(\hat{\mathbf{x}}_{k|k})$ is the persistence probability and p^i_{trust} is the sensor's trust probability. The trust probability represents the sensor's performance in providing a reliable existence probability at the sensor-level. The BBA for existence is then also weighted by the persistence probability, which is described in Section 5.1.4, in order to weigh the evidence for existence less at the edges of the sensor's field-of-view. Similarly, the BBA for non-existence is then the weighted non-existence of the object from sensor i

$$m^i_k(\{\nexists\}) = p^i_{\mathrm{p}}(\hat{\mathbf{x}}_{k|k}) \cdot p^i_{\mathrm{trust}} \cdot \left[1 - p^i(\exists \mathbf{x}_k)\right] \tag{5.57}$$

The remainder due to the weights assigned to the existence and non-existence is assigned to the BBA for $\{\exists, \nexists\}$, which represents the uncertainty that knowledge about the object's existence is unknown

$$m^i_k(\{\exists, \nexists\}) = 1 - \left[m^i_k(\{\exists\}) + m^i_k(\{\nexists\})\right]. \tag{5.58}$$

Using the persistence probability and trust probability as a weight of the existence and non-existence, a modeling of uncertainty in the sensor's ability to provide a reliable existence probability is achieved using the fact that DST is able to model ignorance by assigning evidence to the BBA for $\{\exists, \nexists\}$.

The global object's BBAs are then updated with the sensor-level BBAs from the associated object from sensor i, as described above, using the DST rule of combination from (5.52). Using the DST rule of combination and referencing Table 5.1, the BBA for the fact that the object exists is update with

$$\begin{aligned} m^{\mathrm{G}}_{k|k}(\{\exists\}) &= \frac{\sum_{A\cap B=\exists} m^{\mathrm{G}}_{k|k-1}(A)\, m^i_k(B)}{1 - \sum_{A\cap B=\varnothing} m^{\mathrm{G}}_{k|k-1}(A)\, m^i_k(B)} \\ &= \frac{m^{\mathrm{G}}_{k|k-1}(\{\exists\})\, m^i_k(\{\exists\}) + m^{\mathrm{G}}_{k|k-1}(\{\exists\})\, m^i_k(\{\exists, \nexists\}) + m^{\mathrm{G}}_{k|k-1}(\{\exists, \nexists\})\, m^i_k(\{\exists\})}{1 - \left[m^{\mathrm{G}}_{k|k-1}(\{\exists\})\, m^i_k(\{\nexists\}) + m^{\mathrm{G}}_{k|k-1}(\{\nexists\})\, m^i_k(\{\exists\})\right]}. \end{aligned} \tag{5.59}$$

Similarly, the hypothesis that the object does not exist is updated with

$$m^{\mathrm{G}}_{k|k}(\{\nexists\}) = \frac{\sum_{A\cap B=\nexists} m^{\mathrm{G}}_{k|k-1}(A)\, m^i_k(B)}{1-\sum_{A\cap B=\varnothing} m^{\mathrm{G}}_{k|k-1}(A)\, m^i_k(B)} \tag{5.60}$$
$$= \frac{m^{\mathrm{G}}_{k|k-1}(\{\nexists\})\, m^i_k(\{\nexists\}) + m^{\mathrm{G}}_{k|k-1}(\{\nexists\})\, m^i_k(\{\exists,\nexists\}) + m^{\mathrm{G}}_{k|k-1}(\{\exists,\nexists\})\, m^i_k(\{\nexists\})}{1-\left[m^{\mathrm{G}}_{k|k-1}(\{\exists\})\, m^i_k(\{\nexists\}) + m^{\mathrm{G}}_{k|k-1}(\{\nexists\})\, m^i_k(\{\exists\})\right]}.$$

Finally, the ignorance, or uncertainty, about the object's existence is updated with

$$m^{\mathrm{G}}_{k|k}(\{\exists,\nexists\}) = \frac{\sum_{A\cap B=\exists,\nexists} m^{\mathrm{G}}_{k|k-1}(A)\, m^i_k(B)}{1-\sum_{A\cap B=\varnothing} m^{\mathrm{G}}_{k|k-1}(A)\, m^i_k(B)} \tag{5.61}$$
$$= \frac{m^{\mathrm{G}}_{k|k-1}(\{\exists,\nexists\})\, m^i_k(\{\exists,\nexists\})}{1-\left[m^{\mathrm{G}}_{k|k-1}(\{\exists\})\, m^i_k(\{\nexists\}) + m^{\mathrm{G}}_{k|k-1}(\{\nexists\})\, m^i_k(\{\exists\})\right]}.$$

No Object Associated

If a sensor-level object is not associated to a global object, then existence must also be updated to reflect the fact that the global object was not observed. This has the effect of reducing the global object's existence over time, if an association fails several times in a row. Similar to when an object is associated, the BBAs for sensor i are updated as a weight using the persistence probability and the trust probability. However, since an object was not associated, the BBA for existence is given a value of 0:

$$m^i_k(\{\exists\}) = 0. \tag{5.62}$$

The BBA for the hypothesis that the object does not exist is then

$$m^i_k(\{\nexists\}) = p^i_{\mathrm{p}}(\hat{\mathbf{x}}_{k|k}) \cdot p^i_{\mathrm{trust}}. \tag{5.63}$$

The remainder is assigned to the BBA for the hypothesis that information about the object's existence is unknown:

$$m^i_k(\{\exists,\nexists\}) = 1 - p^i_{\mathrm{p}}(\hat{\mathbf{x}}_{k|k}) \cdot p^i_{\mathrm{trust}}. \tag{5.64}$$

The DST combination for updating the global object's BBAs are carried out in a similar fashion as when an object is associated. The DST combination for the object's existence, due to the fact that the sensor-level BBA for existence is zero, is now

$$m^{\mathrm{G}}_{k|k}(\{\exists\}) = \frac{m^{\mathrm{G}}_{k|k-1}(\{\exists\})\, m^i_k(\{\exists,\nexists\})}{1-m^{\mathrm{G}}_{k|k-1}(\{\exists\})\, m^i_k(\{\nexists\})} \tag{5.65}$$

where the combined BBA for the hypothesis that the object does not exist or information about the object's existence is unknown remains the same as in (5.60-5.61).

The global object should only be updated with non-existence evidence as described in this section if it falls within the field-of-view of sensor i. This can be determined by a threshold on the persistence probability.

Conversion to Bayesian Probability

The object model interface described in Section 2.2.1 requires that the existence probability is a single value, which requires a conversion of existence from the Dempster-Shafer framework form back to a Bayesian probabilistic form. A conversion from the Dempster-Shafer framework to a probability function can be achieved using the Pignistic transformation [247], given by

$$\mathrm{BetP}(A) = \sum_{B \subseteq \Theta} \frac{|A \cap B|}{|B|} m(B) \tag{5.66}$$

where $|A \cap B|$ is the number of elementary elements of the power set in $A \cap B$ and $|B|$ is the number of elementary elements of B. Therefore, the probability of existence for the fused global object, $p^{\mathrm{G}}(\exists \mathbf{x}_k)$, from the DST framework can be calculating using the Pignistic transformation with

$$\begin{aligned} p^{\mathrm{G}}(\exists \mathbf{x}_k) = \mathrm{BetP}(\{\exists\}) &= \sum_{B \subseteq \Theta} \frac{|\{\exists\} \cap B|}{|B|} m^{\mathrm{G}}_{k|k}(B) \\ &= m^{\mathrm{G}}_{k|k}(\{\exists\}) + \frac{1}{2} m^{\mathrm{G}}_{k|k}(\{\exists, \nexists\}). \end{aligned} \tag{5.67}$$

The non-existence is then simply the complement

$$p^{\mathrm{G}}(\nexists \mathbf{x}_k) = 1 - p^{\mathrm{G}}(\exists \mathbf{x}_k). \tag{5.68}$$

5.2.3 Extension for Occlusion Modeling

The problem with the modeling of existence from the previous section is that some sensors are able to reliably detect objects, even during occlusions. When such a sensor exists and is observing the same area as a sensor that is unable to detect occluded objects, then the situation occurs that one sensor tends to raise the existence and the other tends to decrease it, creating a conflict. Occlusion modeling in tracking applications is a common problem for which many different solutions have been presented [155, 236]. In this section, an approach to model existence with occlusions using DST is described.

The frame of discernment for existence is changed slightly compared to (5.46). The hypothesis that an object exists is now split into two separate hypotheses, one representing that the object exists but is not occluded and the other representing the fact the object exists, but is occluded:

$$\Theta = \{\exists \bar{\mathrm{O}}, \exists \mathrm{O}, \nexists\} \tag{5.69}$$

where $\bar{\mathrm{O}}$ stands for not occluded and O for occluded. Expanding Θ to the power set results in

$$2^{\Theta} = \{\varnothing, \{\exists \bar{\mathrm{O}}\}, \{\exists \mathrm{O}\}, \{\nexists\}, \{\exists \bar{\mathrm{O}}, \nexists\}, \{\exists \mathrm{O}, \nexists\}, \{\exists \bar{\mathrm{O}}, \exists \mathrm{O}\}, \Theta\} \tag{5.70}$$

where new possibilities for modeling ignorance, or uncertainty, exist. With $\{\exists \mathrm{O}, \nexists\}$ it is possible to assign evidence to the fact that the object may exists and is occluded or that it does not exist at all, an uncertainty that is very realistic for sensors that are not able to detect occluded objects. Using $\{\exists \bar{\mathrm{O}}, \exists \mathrm{O}\}$, evidence can be assigned to the fact that an object exists, regardless of whether it is occluded or not, which would model the existence evidence for a sensor that is able to detect occluded objects. Only $\{\exists \bar{\mathrm{O}}, \nexists\}$ is an element

Table 5.2: Intersections of the power set for existence with occlusion modeling, useful for evaluating DST combinations.

$\cap$	$\{\exists\bar{\mathrm{O}}\}$	$\{\exists\mathrm{O}\}$	$\{\nexists\}$	$\{\exists\bar{\mathrm{O}},\nexists\}$	$\{\exists\mathrm{O},\nexists\}$	$\{\exists\bar{\mathrm{O}},\exists\mathrm{O}\}$	Θ
$\{\exists\bar{\mathrm{O}}\}$	$\{\exists\bar{\mathrm{O}}\}$	$\varnothing$	$\varnothing$	$\{\exists\bar{\mathrm{O}}\}$	$\varnothing$	$\{\exists\bar{\mathrm{O}}\}$	$\{\exists\bar{\mathrm{O}}\}$
$\{\exists\mathrm{O}\}$	$\varnothing$	$\{\exists\mathrm{O}\}$	$\varnothing$	$\varnothing$	$\{\exists\mathrm{O}\}$	$\{\exists\mathrm{O}\}$	$\{\exists\mathrm{O}\}$
$\{\nexists\}$	$\varnothing$	$\varnothing$	$\{\nexists\}$	$\{\nexists\}$	$\{\nexists\}$	$\varnothing$	$\{\nexists\}$
$\{\exists\bar{\mathrm{O}},\nexists\}$	$\{\exists\bar{\mathrm{O}}\}$	$\varnothing$	$\{\nexists\}$	$\{\exists\bar{\mathrm{O}},\nexists\}$	$\{\nexists\}$	$\{\exists\bar{\mathrm{O}}\}$	$\{\exists\bar{\mathrm{O}},\nexists\}$
$\{\exists\mathrm{O},\nexists\}$	$\varnothing$	$\{\exists\mathrm{O}\}$	$\{\nexists\}$	$\{\nexists\}$	$\{\exists\mathrm{O},\nexists\}$	$\{\exists\mathrm{O}\}$	$\{\exists\mathrm{O},\nexists\}$
$\{\exists\bar{\mathrm{O}},\exists\mathrm{O}\}$	$\{\exists\bar{\mathrm{O}}\}$	$\{\exists\mathrm{O}\}$	$\varnothing$	$\{\exists\bar{\mathrm{O}}\}$	$\{\exists\mathrm{O}\}$	$\{\exists\bar{\mathrm{O}},\exists\mathrm{O}\}$	$\{\exists\bar{\mathrm{O}},\exists\mathrm{O}\}$
Θ	$\{\exists\bar{\mathrm{O}}\}$	$\{\exists\mathrm{O}\}$	$\{\nexists\}$	$\{\exists\bar{\mathrm{O}},\nexists\}$	$\{\exists\mathrm{O},\nexists\}$	$\{\exists\bar{\mathrm{O}},\exists\mathrm{O}\}$	Θ

that is not realistic in practice and will therefore not be used. As in the previous section, complete ignorance can be modeled by assigning evidence to Θ.

With the help of Table 5.2, showing the intersections of the power set, the DST rule of combination can be used to update a global object's existence at k using the predicted object's existence and the existence probability of the sensor i. For each update, all of the BBAs of the power set must be evaluated:

$$m^{\mathrm{G}}_{k|k}(\{\exists\bar{\mathrm{O}}\}) = \frac{\sum_{A\cap B=\exists\bar{\mathrm{O}}} m^{\mathrm{G}}_{k|k-1}(A)\, m^{i}_{k}(B)}{1-\sum_{A\cap B=\varnothing} m^{\mathrm{G}}_{k|k-1}(A)\, m^{i}_{k}(B)} \tag{5.71}$$

$$m^{\mathrm{G}}_{k|k}(\{\exists\mathrm{O}\}) = \frac{\sum_{A\cap B=\exists\mathrm{O}} m^{\mathrm{G}}_{k|k-1}(A)\, m^{i}_{k}(B)}{1-\sum_{A\cap B=\varnothing} m^{\mathrm{G}}_{k|k-1}(A)\, m^{i}_{k}(B)} \tag{5.72}$$

$$m^{\mathrm{G}}_{k|k}(\{\nexists\}) = \frac{\sum_{A\cap B=\nexists} m^{\mathrm{G}}_{k|k-1}(A)\, m^{i}_{k}(B)}{1-\sum_{A\cap B=\varnothing} m^{\mathrm{G}}_{k|k-1}(A)\, m^{i}_{k}(B)} \tag{5.73}$$

$$m^{\mathrm{G}}_{k|k}(\{\exists\bar{\mathrm{O}},\nexists\}) = \frac{\sum_{A\cap B=\exists\bar{\mathrm{O}},\nexists} m^{\mathrm{G}}_{k|k-1}(A)\, m^{i}_{k}(B)}{1-\sum_{A\cap B=\varnothing} m^{\mathrm{G}}_{k|k-1}(A)\, m^{i}_{k}(B)} \tag{5.74}$$

$$m^{\mathrm{G}}_{k|k}(\{\exists\mathrm{O},\nexists\}) = \frac{\sum_{A\cap B=\exists\mathrm{O},\nexists} m^{\mathrm{G}}_{k|k-1}(A)\, m^{i}_{k}(B)}{1-\sum_{A\cap B=\varnothing} m^{\mathrm{G}}_{k|k-1}(A)\, m^{i}_{k}(B)} \tag{5.75}$$

$$m^{\mathrm{G}}_{k|k}(\{\exists\mathrm{O},\exists\bar{\mathrm{O}}\}) = \frac{\sum_{A\cap B=\exists\mathrm{O},\exists\bar{\mathrm{O}}} m^{\mathrm{G}}_{k|k-1}(A)\, m^{i}_{k}(B)}{1-\sum_{A\cap B=\varnothing} m^{\mathrm{G}}_{k|k-1}(A)\, m^{i}_{k}(B)} \tag{5.76}$$

$$m^{\mathrm{G}}_{k|k}(\Theta) = \frac{\sum_{A\cap B=\Theta} m^{\mathrm{G}}_{k|k-1}(A)\, m^{i}_{k}(B)}{1-\sum_{A\cap B=\varnothing} m^{\mathrm{G}}_{k|k-1}(A)\, m^{i}_{k}(B)} \tag{5.77}$$

Assigning the BBAs for sensor i is a bit more involved as in the previous section, due to the fact that the information about the sensor's occlusion detecting abilities must be known and properly modeled. The following will describe how to predict and update a global object's existence with this new modeling of occlusion information.

Existence Prediction

Prediction of the existence probability at the fusion-level with occlusion modeling is achieved in the same as described in the previous section. Using a weight factor γ, a small amount of evidence is removed from all of the elements of the power sets except for Θ. The removed evidence is then added to the element Θ representing complete ignorance or uncertainty in the object's existence.

Object Associated with Occlusion Detecting Sensor

The first possibility is that an object is associated to the global object with a sensor that is able to detect potentially occluded objects. Therefore, evidence for the object's existence, weighted by $p_{\mathrm{p}}^{i}(\hat{\mathbf{x}}_{k|k})$ and p_{trust}^{i}, is assigned to the BBA $m_i(\{\exists\bar{\mathrm{O}}, \exists\mathrm{O}\})$, which represents the fact that the object exists, whether it is actually occluded or not:

$$m_i(\{\exists\bar{\mathrm{O}}, \exists\mathrm{O}\}) = p_{\mathrm{p}}^{i}(\hat{\mathbf{x}}_{k|k}) \cdot p_{\mathrm{trust}}^{i} \cdot p^{i}(\exists\mathbf{x}_k). \tag{5.78}$$

The complement of the existence probability is also weighted and assigned to the BBA for non-existence, just as in (5.57):

$$m_i(\{\nexists\}) = p_{\mathrm{p}}^{i}(\hat{\mathbf{x}}_{k|k}) \cdot p_{\mathrm{trust}}^{i}(1 - p^{i}(\exists\mathbf{x}_k)). \tag{5.79}$$

Similar to (5.58), the remainder is modeled as complete ignorance

$$m_i(\Theta) = 1 - \left[m_i(\{\exists\bar{\mathrm{O}}, \exists\mathrm{O}\}) + m_i(\{\nexists\})\right]. \tag{5.80}$$

By assigning evidence to $\{\exists\bar{\mathrm{O}}, \exists\mathrm{O}\}$, object existence, whether the object is occluded or not, is modeled for a sensor that has the ability to detect occluded objects, for example a radar sensor or a vehicle-to-vehicle communications unit.

Object Associated with Non-Occlusion Detecting Sensor

More specific knowledge about an object's occlusion is known if a sensor-level object is associated to a global object with a sensor which does not have the ability to detect occluded objects. With such a sensor, it is specifically known that the detected object is not occluded, and therefore evidence is assigned to the BBA for $\{\exists\bar{\mathrm{O}}\}$

$$m_i(\{\exists\bar{\mathrm{O}}\}) = p_{\mathrm{p}}^{i}(\hat{\mathbf{x}}_{k|k}) \cdot p_{\mathrm{trust}}^{i} \cdot p^{i}(\exists\mathbf{x}_k). \tag{5.81}$$

As usual, the weighted complement of the existence probability is assigned to the BBA that the object does not exist

$$m_i(\{\nexists\}) = p_{\mathrm{p}}^{i}(\hat{\mathbf{x}}_{k|k}) \cdot p_{\mathrm{trust}}^{i}(1 - p^{i}(\exists\mathbf{x}_k)) \tag{5.82}$$

and the remainder is assigned to the BBA for complete ignorance about the object's existence

$$m_i(\Theta) = 1 - \left[m_i(\{\exists\bar{\mathrm{O}}\}) + m_i(\{\nexists\})\right]. \tag{5.83}$$

Using this new model for existence with occlusions, such specific information that an object exists and is definitely not occluded can be defined if a sensor detects an object and it is known that this sensor, for example a camera or a laser scanner, does not have the ability to detect occlusions.

No Object Associated with Occlusion Detecting Sensor

When no sensor-level object was associated with a global object, then the persistence probability of the sensor can be used to determine if the global object falls within the field-of-view of the sensor. If it does, then the BBAs for existence need to be updated with the fact that the sensor should have detected something, but did not. For a sensor which can detect occlusions, there is no question on whether the sensor should have or should have not detected something. Therefore, weighted evidence of the trust probability is assigned directly to the BBA that the object does not exist:

$$m_i(\{\nexists\}) = p_{\mathrm{p}}^i(\hat{\mathbf{x}}_{k|k}) \cdot p_{\mathrm{trust}}^i. \tag{5.84}$$

The remaining evidence goes to the BBA for complete ignorance about the object's existence

$$m_i(\Theta) = 1 - m_i(\{\nexists\}). \tag{5.85}$$

With an occlusion detecting sensor, if no sensor-level object was associated, then it can be inferred that it is very likely that no object actually exists. Therefore, the evidence for complete non-existence, regardless of occlusion, is updated.

No Object Associated with Non-Occlusion Detecting Sensor

For a sensor which cannot detect occlusions, updating the global track when no sensor-level object was associated is a bit different than with an occlusion detecting sensor. The fact that an object was not associated, since it is a sensor that cannot detect occlusions, does not give the global object any real new information on whether the object exists or not. With a non-occlusion detecting sensor, one of two things is true: either the object exists, but is occluded, or the object simply does not exist. The set $\{\exists \mathrm{O}, \nexists\}$ from 2^{Θ} models exactly this situation, and therefore, the weighted evidence of the trust probability is assigned to this BBA:

$$m_i(\{\exists \mathrm{O}, \nexists\}) = p_{\mathrm{p}}^i(\hat{\mathbf{x}}_{k|k}) \cdot p_{\mathrm{trust}}^i \tag{5.86}$$

The remaining evidence, as always, goes to the BBA for complete ignorance about the object's existence

$$m_i(\Theta) = 1 - m_i(\{\exists \mathrm{O}, \nexists\}) \tag{5.87}$$

Updating the BBA for $\{\exists \mathrm{O}, \nexists\}$ has the effect of equally assigning evidence to the fact that the object could exist, but is occluded, or that it does not exist. This trick allows for a stable existence estimation of an occluded object at the fusion-level, even with sensors that do not have the ability to detect occluded objects.

Conversion to Bayesian Probability

Similar to existence modeling without occlusion, the Pignistic transformation from (5.66) is used to convert to a probability from the DST framework. This time, however, the Pignistic transformation is applied to the evidence associated with $\{\exists \bar{\mathrm{O}}, \exists \mathrm{O}\}$, which represents that an object exists, regardless of occlusion. Applying (5.66) whilst excluding the evidence for

$\{\exists\bar{\mathrm{O}}, \nexists\}$ leads to

$$p^{\mathrm{G}}(\exists\mathbf{x}_k) = \mathrm{BetP}(\{\exists\bar{\mathrm{O}}, \exists\mathrm{O}\}) = \sum_{B\subseteq\Theta} \frac{|\{\exists\bar{\mathrm{O}}, \exists\mathrm{O}\} \cap B|}{|B|} m^{\mathrm{G}}_{k|k}(B)$$
$$= m^{\mathrm{G}}_{k|k}(\{\exists\bar{\mathrm{O}}\}) + m^{\mathrm{G}}_{k|k}(\{\exists\mathrm{O}\}) + \frac{1}{2}m^{\mathrm{G}}_{k|k}(\{\exists\mathrm{O}, \nexists\}) + m^{\mathrm{G}}_{k|k}(\{\exists\bar{\mathrm{O}}, \exists\mathrm{O}\}) + \frac{2}{3}m^{\mathrm{G}}_{k|k}(\Theta).$$

For calculating the non-existence, (5.68) is also used.

5.2.4 Modeling the Trust Probability

The reason for using a trust probability is that the existence probability can be modeled and calculated differently for various sensors. The problem is that an existence probability for a sensor S_1 may not be statistically equivalent to the same existence probability from sensor S_2, such that for a probability P, $p^{\mathrm{S}_1}(\exists\mathbf{x}_k) = P \not\equiv p^{\mathrm{S}_2}(\exists\mathbf{x}_k) = P$. Therefore, when fusing the existence probabilities for a sensor, a weight is used to normalized the statistically meaning of the existence probability for each sensor. In this thesis, this weight is called the trust probability p_{trust}.

The trust probability p_{trust} can be derived from experimental evaluation of a sensor's performance regarding its ability to accurately calculate an existence probability for relevant objects. This achieved by evaluating the existence probability against a large data set, where true and relevant objects are labeled. A Receiver Operating Characteristic (ROC) curve can be calculated from the data set by varying the existence probability and using it as a threshold in order to determine the sensor's performance regarding its true positive rate (TPR) and false positive rate (FPR) [95, 166]. The ROC curve shows the relationship between the true positive rate and false positive rate as the existence probability threshold is varied and can therefore be exploited to obtain a statistically meaningful value for p_{trust}. It can be chosen to normalize the fusion of existence probabilities using a desired false positive rate, such that a defined false positive rate, $\mathrm{FPR} = f$, is used to bring a statistical equivalency to all of the sensors. From a given existence probability ROC curve for sensor i, $\mathrm{ROC}^i_\exists$, a true positive rate can be obtained given a false positive rate f such that

$$p^i_{\mathrm{trust}} = \mathrm{TPR}^i_{\mathrm{FPR}=f} = \mathrm{ROC}^i_\exists(\mathrm{FPR} = f) \tag{5.88}$$

where the trust probability for sensor i is defined by the true positive rate for existence given a fixed false positive rate f amongst all of the sensors. This trust probability, in conjunction with the fusion methods presented in this chapter, is then used to weight the existence probabilities from the various senors in order to obtain a more consistent fusion result.

Discussion

Next to the estimation of an object's state and covariance, it is just as important to calculate a proper quality measure for an object's existence, thereby improving the detection rate and reducing false detections. In this chapter, the well known concept of the existence probability for such a quality measure. In a high-level sensor data fusion architecture, it

is important to properly calculate the existence probability at the sensor-level and fusion-level. At the sensor-level, it is typical to use an IPDA or Joint Integrated Probabilistic Data Association (JIPDA) algorithm in order to simultaneously estimate state and existence. In this thesis, however, a Bayes formulation for object existence was presented, thereby allowing for a simple and loosely coupled estimation of existence, which only slightly depends on the state and covariance estimation process. This allows for more flexibility in the existence probability calculation, especially when applying the algorithm generically for different sensors. The result of the existence probability calculation at the sensor-level can then be used in order to make decisions about object confirmation and deletion by thresholding the existence probability. For the fusion-level, an approach for fusion the existence probability using the Dempster-Shafer evidence theory was presented. The motivation for using DST lies in the fact that existence probabilities from different sensors do not necessarily have a statistically similar meaning. DST allows for a simple representation of uncertainty between hypothesis, in this case existence and non-existence. By weighting the existence probability from various sensor based on their statistical performance, difference amount of evidence for unknown existence are represented and exploited during fusion using the Dempster-Shafer rule of combination. A further extension of the DST formulation for existence fusion for object occlusion was also presented.

At the application-level, the existence probability plays an important role in driver assistance systems and automated driving. As opposed to using heuristics and experimentation for setting the object selection parameters in a driver assistance system's reaction to a detected object, a threshold on the existence probability based on statistical evaluation can reveal information about the relationship between the true positive rate and false positive rate of detection. This information, contained in the existence probability, can be used to trigger a driver assistance system or variably condition the reaction of an assistance system, given the reliability of a detected object. However, such a usage of existence probability and the direct integration of such a quality measure in driver assistance and automated driving systems is still quite new and much still needs to be learned. The effect of setting the thresholds of an existence probability of a driver assistance function, and the increase in safety, comfort and reliability it could provide, still need to be proven.

6 Classification

Besides the state and covariance estimation and the existence probability, the object model introduced in Section 2.2.1 defines a classification vector $\mathbf{c}$. In vehicle environment perception, classification serves the purpose of determining if an object belongs to a certain class type, or label, for example car, truck, motorcycle, pedestrian, etc. Such a classification attribute allows different driver assistance system to react specifically to objects of a certain type. A pedestrian safety system, for example, should only warn the driver if a pedestrian is detected and if that pedestrian poses a potential risk for a collision.

In the field of machine learning, classification is typically considered to be a type of pattern recognition algorithm [39, 88]. For the purposes in this thesis, the problem of classification can be formulated as follows. Given a set of classes

$$\mathcal{C} = \{C_1, C_2, \ldots, C_K\} \tag{6.1}$$

determine the probability $p(C_i \mid \mathcal{I})$ that some given information $\mathcal{I}$ describes an object of type C_i from the set of classes $\mathcal{C}$. For automotive applications the class set can be defined as

$$\mathcal{C} = \{C_{\text{Car}}, C_{\text{Truck}}, C_{\text{Motorcycle}}, C_{\text{Bicycle}}, C_{\text{Pedestrian}}, C_{\text{Stationary}}, C_{\text{Other}}\} \tag{6.2}$$

where the classification probabilities are summarized in a normalized classification vector

$$\mathbf{c} = \begin{bmatrix} C_{\text{Car}} & C_{\text{Truck}} & C_{\text{Motorcycle}} & C_{\text{Bicycle}} & C_{\text{Pedestrian}} & C_{\text{Stationary}} & C_{\text{Other}} \end{bmatrix}' \tag{6.3}$$

where, for example, $C_{\text{Car}} = p(C_{\text{Car}} \mid \mathcal{I})$.

In the previous chapter, the existence probability represented a quality measure for the detection of a relevant object for a driver assistance system. It is often argued what is meant by "relevant object". In this thesis, relevant objects will be considered the traffic object classes from (6.2) and the stationary class, where the stationary class represents free-standing objects (such as cones, tress, etc.) which may obstruct the vehicle's direct driving path. Large stationary objects such as buildings, or continuous road barriers are not considered, since these types of stationary objects cannot be properly represented using a rectangular object model. For more complex stationary objects, a grid-based representation is necessary. The class "Other" includes objects which may be potential relevant objects but have not yet been reliably detected as a member of another class. The class "Other" can also be thought of as a classification hypothesis for an unknown class.

Following the proposed sensor data fusion architecture from Section 2.2, the classification vector is first processed at the sensor-level using only sensor-specific information (Section 6.1). At the fusion-level, classification vectors from several sensors are combined in order to generate a refined classification vector for a specific object (Section 6.2). The presented algorithms and methods in this section, as well as the results from Chapter 7, have been published in [309].

6.1 Sensor-Level Processing

At the sensor-level, it is desired to determine an object's class based solely on that sensor's data over time. The process can vary depending on the type of sensor being used, as the most effective classification algorithm is dependent on the type of data that a sensor can generate for an object.

In the follow sections, some algorithms for object classification at the sensor-level will be described. Throughout these sections, example data extracted from an automotive laser scanner (see Section 7.1) are used to demonstrate the proposed algorithms for classification. Other sensors can follow a similar classification process, but may use different types of input data, depending on which features are the most relevant for distinguishing between the class types described by (6.2).

Note that approaches presented here are a very small subset of the different possibilities for classification, where many other approaches have been applied to automotive applications. For example, a classification framework [292] applied to a laser scanner has been developed, where a classifier based on neural networks is used [291, 293]. Gaussian mixture models with a Bayesian classifier on laser scanner data have also been developed [222]. Classification in urban environment with dense 3D laser scanner data can also be performed [261]. For camera-based systems, typically some form of histogram of oriented gradients (HOG) [72] or Haar-like [281] features are used to detect objects in an image [245], such as vehicles, pedestrians [4], etc., which are then typically trained using a classifier such as neural networks [4] or Support Vector Machines (SVM)s [116]. Classification on intermediate representations, such as stixels, can also be performed, as in [90] with neural networks. Fusion of sensor classification data at a low-level within a Joint Integrated Probabilistic Data Association (JIPDA) framework using lidar and camera have also been implemented [171, 191], where classification focused on whether an object is relevant to an Advanced Driver Assistance Systems (ADAS) function or not.

6.1.1 Measurement Classification

The first opportunity to classify object data is with a sensor measurement, $\mathbf{z}_k$, before data association in the tracking process. For some sensors, the measurements are the raw sensor detections, whereas for other sensors the measurements may already be the result of a preprocessing step, such as point cloud segmentation for a laser scanner sensor. It is desired to determine $p(C_i \mid \mathbf{z}_k)$, the probability that the measurement $\mathbf{z}_k$ belongs to the class C_i, for all classes $C = \{C_1, C_2, \ldots, C_K\}$. In this section, a few supervised learning techniques will be presented and compared to one another for measurement classification.

The first step is to determine which attributes of the measurements in a training set are relevant, or the most useful, for being able to separate the measurement as distinctly as possible into each of the classes. This is usually, and most easily, determined empirically by analyzing a training data set's attributes with respect to each of the classes and identifying the minimal number of attributes that best separate the data into the desired classes. Given a training set of measurement attribute data

$$\mathcal{T} = \{(\mathbf{y}_1, C_1), \ldots, (\mathbf{y}_i, C_i), \ldots, (\mathbf{y}_L, C_L)\} \quad \text{where} \quad i = 1 \ldots L, C_i \in \mathcal{C}, \mathbf{y} \in \mathbb{R}^D \tag{6.4}$$

where

$$\mathbf{y}_i = \mathbf{H}_c \mathbf{z}_i \tag{6.5}$$

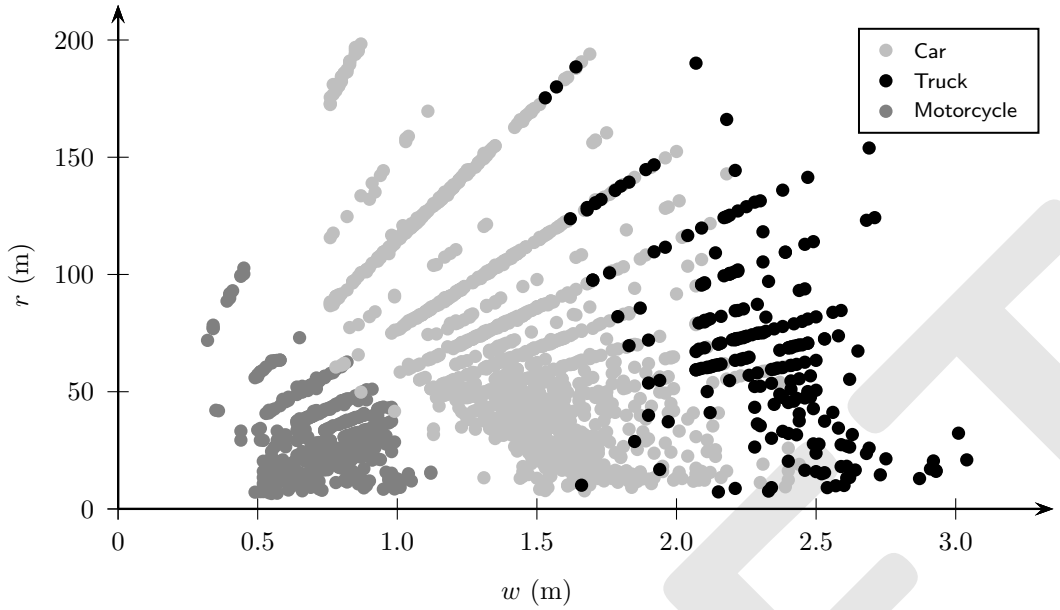


Figure 6.1: Training data from a forward-facing laser scanner sensor in a highway scenario for the classes Car, Truck and Motorcycle using an attribute space containing the range r and width w of a measurement.

is the attribute vector from the full measurement $\mathbf{z}_i$ with attribute transformation matrix $\mathbf{H}_\mathrm{c}$, L is the size of the training set $\mathcal{T}$ and D is the dimension of the attribute space used for classification. Based on this labeled training set, it is desired to generate a model which can be used to classify new sensor measurement data. This section will discuss two such algorithms and their application to laser scanner measurement classification: SVM and Gaussian Discriminant Analysis (GDA). Supervised machine learning, however, is a very large field and many other algorithms exist, such as neural networks [38] or Relevance Vector Machines [272]. Classification of measurement data may heavily depend on the type of sensor used and the sensor data's ability to distinguish between the desired classes. Please refer to [39, 88, 230] for an overview of many different types of machine learning and classification algorithms.

The remainder of this section will present the aforementioned methods for measurement classification and will apply the methods on a training set from a forward-facing laser scanner sensor in a highway scenario. The classification problem on the highway can be restricted to determining if a measurement belongs to the classes Car, Truck or Motorcycle, which is a subset of the full set of classes in the object model. The training data is extracted from a combination of a highway scenario and motorcycle scenarios on a test track (see Appendix C.1). After pruning the training data set from occluded and implausible measurements, a total of $L = 21,592$ data points are used for classification (14,860 Car, 2,334 Truck and 4,398 Motorcycle). After empirical analysis of the training data set, it was determined that the width, w, and range r, where $r = \sqrt{x^2 + y^2}$, were the attributes that best separate the data into the three desired classes, such that $\mathbf{y}_i = \begin{bmatrix} w_i & r_i \end{bmatrix}'$. The attribute space with a reduced visualization of the training data set for the three classes is shown in Figure 6.1.

Support Vector Machines

A common and popular method for classification are SVM, which are a type of maximum margin classifier [39]. SVMs were first introduced by Boser in 1992 [55] and have since been widely studied and used in various pattern recognition applications. For a comprehensive explanation on SVMs, refer to [39, 59, 88, 280]. Traditionally, SVM is a binary classification algorithm, but has been extended to the multi-class problem using various methods [39, 295, 296].

The idea behind SVMs is to find a decision boundary that maximizes the margin between data points from two separate classes using training data $\mathcal{T}$. SVM is based on a linear model decision function of the form

$$f(\mathbf{y}) = \mathbf{w}'\phi(\mathbf{y}) + b \tag{6.6}$$

where $\mathbf{y}$ is a data point, $\phi(\mathbf{y})$ is an attribute-space transformation, $\mathbf{w}$ is a vector normal to the decision boundary and b is a bias parameter. The hyperplane defined by $f(\mathbf{y}) = 0$ is a decision boundary that separates two target classes, t, into two regions where the classes are defined by $t \in \{+1, -1\}$. The goal is to find the parameters $\mathbf{w}$ and b such that for all of the training data $\mathcal{T}$, $f(\mathbf{y}_i) > 0$ for $t_i = +1$ and $f(\mathbf{y}_i) < 0$ for $t_i = -1$ is satisfied, which can also be written as

$$t_i f(\mathbf{y}) \geq 0 \ \ \forall_i \tag{6.7}$$

and such that the margin, or distance, between the hyperplanes defined by $f(\mathbf{y}) = +1$ and $f(\mathbf{y}) = -1$ is maximized. It is assumed that the two classes are linearly separable with the attribute-space transformation, meaning that there exists a hyperplane that can perfectly separate all of the data points from the two classes. The perpendicular distance from a data point to the decision boundary is then defined as

$$\frac{t_i f(\mathbf{y})}{||\mathbf{w}||}. \tag{6.8}$$

The goal of SVM is to maximize the margin, which from (6.8) is $||\mathbf{w}||^{-1}$. This is equivalent to maximizing the expression

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{such that} \quad t_i(\phi(\mathbf{y}_i)\mathbf{w} + b) - 1 \geq 0 \ \ \forall_i \tag{6.9}$$

where the factor $\frac{1}{2}$ is introduced in order to formulate a quadratic programming problem. This optimizing problem is usually solved by introducing Lagrange multipliers which are to be solved in order to determine the optimal values for $\mathbf{w}$ and b [39]. Once the model has been trained, new data points are classified by evaluating the sign of the original expression from (6.6).

The introduction of a slack variables ξ_i allows the SVM problem to be reformulated such that some of the training data may lie on the wrong side of the decision boundary, where the optimization from (6.9) becomes an additional term which introduces a penalization for data points whose slack variable puts them on the wrong side of the decision boundary. Furthermore, the linearity of SVM can be avoided by introducing a non-linear kernel function which operates on the input feature space in order to transform the data into a higher dimensional feature space such that a linear separator can be found. In [220], an

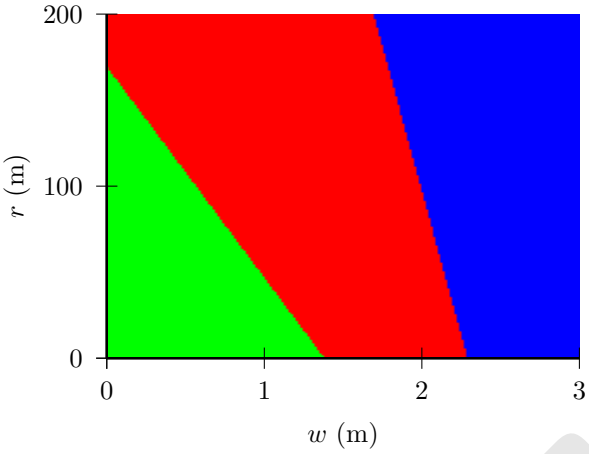


Figure 6.2: Example output of applying SVMs on the laser scanner data from Figure 6.1. The red region corresponds to C_{Car}, blue to C_{Truck} and green to $C_{\text{Motorcycle}}$, where the line between the regions is the maximum margin decision boundary from the SVM models.

approach for producing probabilistic outputs from an SVM model is demonstrated based on the idea of fitting a sigmoid function dependent on a trained model of the classifier.

SVM inherently solves a two-class classification problem. However, most practical applications, as is the case with the classification of objects in this thesis, it is necessary to distinguish between more than two classes. This is solved using SVM by formulating several classifiers and evaluating the results of each against one another. Two schemes for multi-class SVM are typically used: one-versus-all and one-versus-one. In one-versus-all, K classifiers (one for each class to be classified) is trained, where for each classifier the negative samples are defined as the samples which do not belong to the positive class (hence the name one-versus-all). In one-versus-one, several SVM classifiers are trained such that every class is trained against every other class, hence also the name one-versus-one. The most probable class for a new data point is then determined using a heuristic voting approach. This approach can be more effective, but at the cost of higher computation in training and in new data point evaluation. Using a directed acyclic graph approach can reduce these computational costs [221].

The above described SVM approach was applied to the laser scanner data from Figure 6.1 using a one-versus-one multi-class SVM. The classifiers were trained using the LIBSVM [62] software. The resulting decision boundaries are shown in Figure 6.2, where the red region corresponds to C_{Car}, blue to C_{Truck} and green to $C_{\text{Motorcycle}}$.

Gaussian Discriminant Analysis

A Bayesian probabilistic view is an alternative approach to classification. The idea is to use probabilistic models in order to separate data into various classes. A familiar model is to use a Gaussian distribution to represent data for a specific class. This Gaussian model can then be used to discriminate between classes, thus called *Gaussian discriminant analysis* (GDA). In [39], such approaches are called *probabilistic generative models*, whose approach is the basis for the work presented in this section.

Consider the case with just two classes. It is desired to calculate the posterior probability of a class C_i given input data $\mathbf{y}$. Using Bayes' theorem with two classes, this can be

accomplished for C_1 with

$$p(C_1 \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid C_1)p(C_1)}{p(\mathbf{y} \mid C_1)p(C_1) + p(\mathbf{y} \mid C_2)p(C_2)} \tag{6.10}$$

where $p(\mathbf{y} \mid C_1)$ is the class-conditional density of the data $\mathbf{y}$ and $p(C_1)$ is the prior probability. By substituting the log-ratio of the probabilities, (6.10) can be rewritten in the form of a logistic sigmoid function:

$$p(C_1 \mid \mathbf{y}) = \sigma(a) = \frac{1}{1 + e^{-a}} \tag{6.11}$$

$$a = \ln \frac{p(\mathbf{y} \mid C_1)p(C_1)}{p(\mathbf{y} \mid C_2)p(C_2)}. \tag{6.12}$$

For more than two classes, where $N > 2$, using Bayes' theorem gives

$$p(C_i \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid C_i)p(C_i)}{\sum_{j=1}^{N} p(\mathbf{y} \mid C_j)p(C_j)} \tag{6.13}$$

which can be written in the normalized exponential form with

$$p(C_i \mid \mathbf{y}) = \frac{e^{a_i}}{\sum_{j=1}^{N} e^{a_j}} \tag{6.14}$$

$$a_i = \ln p(\mathbf{y} \mid C_i)p(C_i). \tag{6.15}$$

In order to evaluate $p(C_i \mid \mathbf{y})$, the class-conditional density $p(\mathbf{y} \mid C_i)$ and the prior class probability $p(C_i)$ must be determined. For GDA, a Gaussian distribution is used to model the class-conditional density

$$p(\mathbf{y} \mid C_i) = \frac{1}{(2\pi)^{\frac{D}{2}} |\mathbf{\Sigma}_i|^{\frac{1}{2}}} \exp\Big\{ -\frac{1}{2}(\mathbf{y} - \boldsymbol{\mu}_i)' \mathbf{\Sigma}_i^{-1} (\mathbf{y} - \boldsymbol{\mu}_i) \Big\} \tag{6.16}$$

where D is the dimension of the attribute space used for classification, $\boldsymbol{\mu}_i$ is the model mean for C_i and $\mathbf{\Sigma}_i$ is its covariance. The parameters of the Gaussian distribution are calculated using a maximum likelihood estimator on the training data set $\mathcal{T}$ for the labeled data corresponding to class C_i. For each class in the data set, a separate Gaussian distribution, and therefore model, is calculated. The prior class probability is the fraction of data points in the training set $\mathcal{T}$ assigned to class C_i to the total number of data points:

$$p(C_i) = \frac{L_i}{L} \tag{6.17}$$

where L_i is the number labeled data point corresponding to class C_i and L is the total number of data points in the training set $\mathcal{T}$. Based on the training set, this prior class probability represents how often a given class occurs with respect to the other classes.

If the covariance parameter in $p(\mathbf{y} \mid C_i)$ is forced to be the same over all of the classes C_i in the training set $\mathcal{T}$, then a simple linear decision boundary between the classes results [39]. From (6.14-6.15), for the shared covariance case, a_i can be written in the form

$$a_i = \mathbf{w}_i' \mathbf{y} + w_{i_0} \tag{6.18}$$

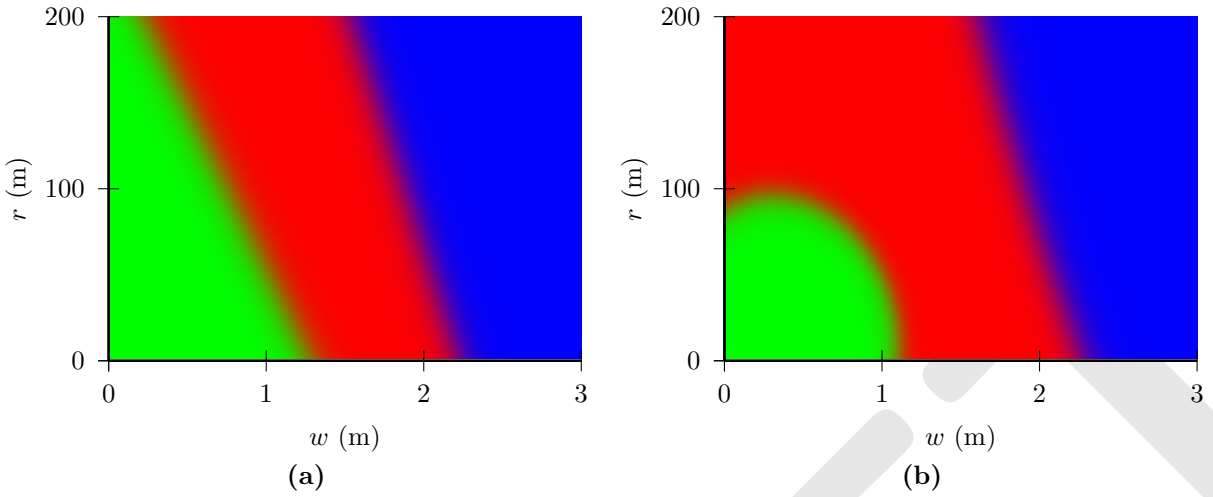


Figure 6.3: Classification using Gaussian discriminant analysis (GDA) using a shared covariance, resulting in a linear decision boundary (a) and class-independent covariances, resulting in quadratic decision boundaries (b). The class posterior probability is color-coded as an RGB image, where red is $p(C_{\text{Car}} \mid \mathbf{y})$, blue is $p(C_{\text{Truck}} \mid \mathbf{y})$ and green is $p(C_{\text{Motorcycle}} \mid \mathbf{y})$.

where the parameters $\mathbf{w}_i$ and w_{i_0} are

$$\mathbf{w}_i = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i \tag{6.19}$$

$$w_{i_0} = -\frac{1}{2}\boldsymbol{\mu}_i{}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i + \ln p(C_i) \tag{6.20}$$

where here $\boldsymbol{\Sigma}$, without a class subscript, is the shared covariance result of the maximum likelihood estimator over all of the classes [39]. The results of applying the shared covariance method for GDA classification to the data from Figure 6.1 is shown in Figure 6.3a. Notice the linear decision boundary between the three classes.

Alternatively, the Gaussian model for each class can have its own, independent covariance. Assuming independent covariance between class models, (6.14-6.15) evaluates to the quadratic form

$$a_i = \mathbf{y}'\mathbf{W}_i\mathbf{y} + \mathbf{w}_i'\mathbf{y} + w_{i_0} \tag{6.21}$$

where

$$\mathbf{W}_i = -\frac{1}{2}\boldsymbol{\Sigma}_i^{-1} \tag{6.22}$$

$$\mathbf{w}_i = \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i \tag{6.23}$$

$$w_{i_0} = -\frac{1}{2}\boldsymbol{\mu}_i{}'\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i + \ln\frac{1}{|\boldsymbol{\Sigma}_i|^{\frac{1}{2}}} + \ln p(C_i) \tag{6.24}$$

This time $\boldsymbol{\Sigma}_i$ represents the covariance of the Gaussian model for class C_i, where for each class there exists a separate covariance. The results of applying the quadratic form of GDA classification to the laser scanner training data from Figure 6.1 is shown in Figure 6.3b. Note that the decision boundaries are no longer linear, but quadratic.

The shared covariance GDA method results in a linear separator quite similar to maximum margin separator from the SVM algorithm. However, the GDA method more easily calculates a classification probability. For simple attribute spaces, as the one in the example training data from Figure 6.1, the GDA algorithm is a great alternative to SVM.

Mahalanobis Distance Class Likelihood

In the previous section, a Gaussian model was used to generate decision boundaries between classes in order to calculate the class posterior probability. The problem with the aforementioned approach is that the method fails to properly model data points which lie far from the class Gaussian median. For example, using the linear GDA model on the training data set used in this section (see Figure 6.3a) and calculating the class posterior probability for the data point $\mathbf{y} = \begin{bmatrix} w & r \end{bmatrix}' = \begin{bmatrix} 10 & 100 \end{bmatrix}'$ for the class *truck* results in $p(\mathbf{y} \mid C_{\text{Truck}}) = 1$, which is an unrealistic result (it is very improbable that a truck has a width of 10 m). It is desired to model the posterior probability that a data point does not belong to any of the modeled classes, which can be represented in the class C_{Other}. In this section, a class likelihood method using a Gaussian model based on the Mahalanobis distance is introduced in order to solve this problem.

In the state vector object association methods presented in Section 3.3, the Mahalanobis distance is used to determine if two objects from different sources represent the same object in reality. The same idea can be used to determine how well a data point in the classification attribute space represents one of the modeled classes. Given the data point $\mathbf{y}$ and the Gaussian class model, then the Mahalanobis distance is

$$d^2_{C_i} = (\mathbf{y} - \boldsymbol{\mu}_i)' \boldsymbol{\Sigma}_i^{-1} (\mathbf{y} - \boldsymbol{\mu}_i) \tag{6.25}$$

The resulting statistical distance, $d^2_{C_i}$, is χ^2 distributed with D degrees of freedom. Using the χ^2 cumulative distribution function $F_{\chi^2_D}$, the posterior probability, or likelihood, $p(C_i \mid \mathbf{y})$ that the data point represents the class C_i can be calculated:

$$p(C_i \mid \mathbf{y}) = 1 - F_{\chi^2_D}(d^2_{C_i}) \tag{6.26}$$

The probability that a data point does not belong to one of the modeled classes can be computed with

$$p(C_{\text{Other}} \mid \mathbf{y}) = 1 - \left[p(C_{\text{Car}} \mid \mathbf{y}) + p(C_{\text{Truck}} \mid \mathbf{y}) + p(C_{\text{Motorcycle}} \mid \mathbf{y}) \right]. \tag{6.27}$$

Note that in some cases, the classification vector, $\mathbf{c}$, must be normalized if the sum of all of the classes is greater than 1. The result of applying the Mahalanobis distance method of class membership likelihood on the training set used in this section is shown in Figure 6.4a. In contrast to the methods presented in the previous section, the likelihood that the data point cannot be classified to one of the modeled classes ($p(C_{\text{Other}} \mid \mathbf{y})$) is visualized as black.

Unfortunately, using the maximum likelihood estimator for modeling the class Gaussian, depending on the distribution of the training set, leads to conservative posterior class probabilities with a large assignment to C_{Other}. In practice, this problem can be easily solved by introducing a gain γ on the class covariance $\boldsymbol{\Sigma}_i$, which has the effect of inflating the spread of the Gaussian distribution:

$$d^2_{C_i} = (\mathbf{y} - \boldsymbol{\mu}_i)' (\gamma \boldsymbol{\Sigma}_i)^{-1} (\mathbf{y} - \boldsymbol{\mu}_i). \tag{6.28}$$

This modified calculation of the Mahalanobis distance allows data points which lie further from the class Gaussian median to be assigned higher posterior probabilities. The result of applying the modified Mahalanobis distance to the training set used in this section is shown in Figure 6.4b, where a gain of $\gamma = 5$ was used for each class. Ideally, γ should be chosen such that for any value of the attribute space, the sum of the posterior probabilities is maximized and at most equal to 1.

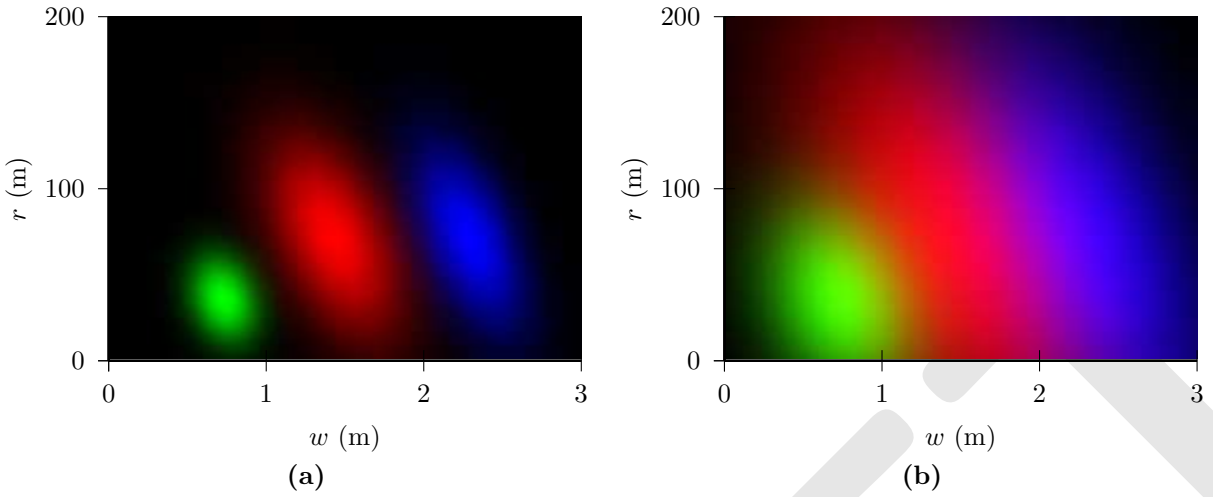


Figure 6.4: Classification using a Gaussian class model with the Mahalanobis distance (a). This method is able to represent the fact that a data point in the attribute space could not be classified to one of the modeled classes (shown in black). In (b), a gain of $\gamma = 5$ is used to inflate the Gaussian model for improved class association.

6.1.2 Temporal Filtering

In the previous section, some examples were shown for generating a classification model based on some measurement data. This data, however, is newly classified for every processing cycle, which may be prone to temporal outliers if for a single given processing cycle the classification results were not that reliable. Therefore, it is desired to temporally filter the classification results over time in order to avoid such outliers.

A simple Bayesian approach can be used for filtering the classification vector $\mathbf{c}$ over time

$$p(\mathbf{c}_k \mid Z^k) = \eta\, p(\mathbf{z}_k \mid \mathbf{c}_k)\, p(\mathbf{c}_k \mid Z^{k-1}) \tag{6.29}$$

where $p(\mathbf{c}_k \mid Z^{k-1})$ is the predicted classification vector at time k given measurement data up to time $k-1$, $p(\mathbf{z}_k \mid \mathbf{c}_k)$ is the new classification information at the current time k, $p(\mathbf{c}_k \mid Z^k)$ is the updated classification vector at time k with new measurement data from time k and η is a normalization factor. The predicted classification vector $p(\mathbf{c}_k \mid Z^{k-1})$ can be realized by defining a classification transition probabilities $p(\mathbf{c}_k \mid C_i^{k-1})$ for transitioning the classification vector from time $k-1$ to time k:

$$p(\mathbf{c}_k \mid Z^{k-1}) = \sum_{C_i \in \mathcal{C}} p(\mathbf{c}_k \mid C_i^{k-1})\, p(C_i^{k-1} \mid Z^{k-1}). \tag{6.30}$$

The new classification information $p(\mathbf{z}_k \mid \mathbf{c}_k)$ is derived directly from the classification results using the trained sensor models:

$$p(\mathbf{z}_k \mid \mathbf{c}_k) = \sum_{C_i \in \mathcal{C}} p(C_i^k \mid \mathbf{y}_k). \tag{6.31}$$

6.2 Fusion

As with the existence probability, an object's classification probabilities from various sensors also need to be fused together using a sensor-to-global fusion strategy. Unlike with an

existence probability, the classification vector is multi-dimensional and probabilities pertaining to several classes need to be fused together in a consistent manner. In this chapter, a novel algorithm for fusing classification probabilities together will be presented.

6.2.1 Modeling with the Dempster-Shafer Evidence Theory

A similar approach is used to fuse classification probabilities as was done with existence probabilities in Chapter 5. The process of classification fusion with the Dempster-Shafer evidence theory (DST) begins with defining the frame of discernment of mutually exclusive hypotheses. The individual classes for possible objects in a vehicle's environment is used

$$\Theta = \{C_{\text{Car}}, C_{\text{Truck}}, C_{\text{Motorcycle}}, C_{\text{Bicycle}}, C_{\text{Pedestrian}}, C_{\text{Stationary}}\} \tag{6.32}$$

where (6.32) differs from (6.2) in that the class C_{Other} is not included in the frame of discernment, as it does not represent an identifiable class of an object and is rather a placeholder class for objects that do not fit the characteristics of a relevant object class. However, as will be shown, the class C_{Other} will still be determined within the Dempster-Shafer Evidence Theory (DST) fusion framework.

With the six object class hypotheses, a power set 2^{Θ} with $2^6 = 64$ combinations of the frame of discernment would result. However, a DST formulation with such a large power set is quite impractical. In order to reduce the complexity, the power set is reduced to a subset of the complete power set

$$\begin{aligned} 2_{\text{r}}^{\Theta} \subset 2^{\Theta} = \{&\varnothing, \{C_{\text{Car}}\}, \{C_{\text{Truck}}\}, \{C_{\text{Motorcycle}}\}, \{C_{\text{Bicycle}}\}, \{C_{\text{Pedestrian}}\}, \{C_{\text{Stationary}}\}, \\ &\{C^*_{\text{Vehicle}}\}, \{C^*_{\text{VRU}}\}, \{C^*_{\text{Traffic}}\}, \{C^*_{\text{Vehicle}}, C_{\text{Stationary}}\}, \{C^*_{\text{VRU}}, C_{\text{Stationary}}\}, \Theta\} \end{aligned} \tag{6.33}$$

where the following "super classes" are defined as a superset of the class definitions in Θ:

$$C^*_{\text{Vehicle}} = \{C_{\text{Car}}, C_{\text{Truck}}, C_{\text{Motorcycle}}\} \tag{6.34}$$

$$C^*_{\text{VRU}} = \{C_{\text{Bicycle}}, C_{\text{Pedestrian}}\} \tag{6.35}$$

$$C^*_{\text{Traffic}} = \{C^*_{\text{Vehicle}}, C^*_{\text{VRU}}\}. \tag{6.36}$$

These so-called "super classes" define a logical grouping of the individual object classes, such that a distinction can be made to a "category" of a class using these supersets. The three supersets chosen are that of a vehicle, which contains the classes Car, Truck and Motorcycle, which are all three typical traffic vehicle that have the capability of reaching all traffic speeds (city or highway) and tend to follow the same rules on the road. The second superset is that of a Vulnerable Road User (VRU), which contains the classes pedestrian and bicycle, both of which are objects not protected by their own vehicle and follow different rules on the road, but are still involved in many traffic scenarios, especially in cities. Furthermore, the superset Traffic refers simply to any object which can actively participate and influence traffic in some way. Using these distinctions, it is not necessary to assign evidence directly to a specific class, but to a category of classes. This will be useful for sensors which may not be able to differentiate well between individual classes, but may have the ability to differentiate between the supersets.

The basic belief assignments (BBAs) are then made to the reduced power set

$$m : 2_{\text{r}}^{\Theta} \rightarrow [0, 1] \tag{6.37}$$

where the sum of all of the Basic Belief Assignment (BBA)s add to one. The fusion module tracks the BBAs for each object over time, where new sensor evidence, also in form of BBAs, is combined into the global running evidence. The Dempster-Shafer rule of combination is used to combine new sensor evidence into the fused, global classification evidence of the object:

$$m_k^{\mathrm{G}}(C) = \frac{\sum_{A \cap B = C} m_{k-1}^{\mathrm{G}}(A)\, m_k^i(B)}{1 - \sum_{A \cap B = \varnothing} m_{k-1}^{\mathrm{G}}(A)\, m_k^i(B)} \tag{6.38}$$

where $m_k^{\mathrm{G}}(C)$ is the updated BBA with the new evidence provided by sensor i for the set C of the reduced power set 2_{r}^{Θ}.

Evaluating the Basic Belief Assignments

The BBAs for the new sensor evidence, $m_k^i(C)$, is assigned to the sets in the reduced power set 2_{r}^{Θ} using the input classification vector, $\mathbf{c}_k^i$ from sensor i at the current time k. The problem lies in how to assign the classification vector in the sets of the reduced power set, where $\dim(\mathbf{c}_k^i) < \dim(2_{\mathrm{r}}^{\Theta})$.

Not all sensors will be able to equally quantify the classification of an object, therefore weighting factors are introduced in order to compensate for the quality of classification estimation for each sensor. For each class, a separate weighting class, or trust probability, is defined, similar to the weighting process for existence probability fusion in Section 5.2. Using the trust probabilities for the classes in c, the BBA for the class sets in Θ are assigned as follows:

$$m_k^i(\{C_{\mathrm{Car}}\}) = p_{\mathrm{trust}}^i(C_{\mathrm{Car}}) \cdot p_k^i(C_{\mathrm{Car}}) \tag{6.39}$$
$$m_k^i(\{C_{\mathrm{Truck}}\}) = p_{\mathrm{trust}}^i(C_{\mathrm{Truck}}) \cdot p_k^i(C_{\mathrm{Truck}}) \tag{6.40}$$
$$m_k^i(\{C_{\mathrm{Motorcycle}}\}) = p_{\mathrm{trust}}^i(C_{\mathrm{Motorcycle}}) \cdot p_k^i(C_{\mathrm{Motorcycle}}) \tag{6.41}$$
$$m_k^i(\{C_{\mathrm{Bicycle}}\}) = p_{\mathrm{trust}}^i(C_{\mathrm{Bicycle}}) \cdot p_k^i(C_{\mathrm{Bicycle}}) \tag{6.42}$$
$$m_k^i(\{C_{\mathrm{Pedestrian}}\}) = p_{\mathrm{trust}}^i(C_{\mathrm{Pedestrian}}) \cdot p_k^i(C_{\mathrm{Pedestrian}}) \tag{6.43}$$
$$m_k^i(\{C_{\mathrm{Stationary}}\}) = p_{\mathrm{trust}}^i(C_{\mathrm{Stationary}}) \cdot p_k^i(C_{\mathrm{Stationary}}) \tag{6.44}$$

Each BBA of the single class set is a weighted factor of the probability in the classification vector for sensor i. The trust probabilities are either given by the sensor or determined experimentally; see Section 6.2.2 for an example.

With the trust probabilities, the BBAs of the single class sets allow for some uncertainty in the classification estimation for each sensor. The left over evidence after assigning the single class sets is distributed over the remaining sets in 2_{r}^{Θ}. First, the super sets C_{Vehicle}^* and C_{VRU}^* are assigned proportionally to the amount of evidence for or against a vehicle or VRU und on the probability that the object has moved. For C_{Vehicle}^*, this leads to

$$\begin{aligned} m_k^i(\{C_{\mathrm{Vehicle}}^*\}) = p_{\mathrm{moved}}^i \cdot p_k^i(C_{\mathrm{Vehicle}}^*) \cdot \Big[&(1 - p_{\mathrm{trust}}^i(C_{\mathrm{Car}})) \cdot p_k^i(C_{\mathrm{Car}}) + \\ &+ (1 - p_{\mathrm{trust}}^i(C_{\mathrm{Truck}})) \cdot p_k^i(C_{\mathrm{Truck}}) + (1 - p_{\mathrm{trust}}^i(C_{\mathrm{Motorcycle}})) \cdot p_k^i(C_{\mathrm{Motorcycle}}) \Big] \end{aligned} \tag{6.45}$$

where

$$p_k^i(C^*_{\text{Vehicle}}) = \frac{m_k^i(\{C_{\text{Car}}\}) + m_k^i(\{C_{\text{Truck}}\}) + m_k^i(\{C_{\text{Motorcycle}}\})}{m_k^i(\{C_{\text{Car}}\}) + m_k^i(\{C_{\text{Truck}}\}) + m_k^i(\{C_{\text{Motorcycle}}\}) + m_k^i(\{C_{\text{Bicycle}}\}) + m_k^i(\{C_{\text{Pedestrian}}\})} \tag{6.46}$$

and for C^*_{VRU} this leads to

$$m_k^i(\{C^*_{\text{VRU}}\}) = p^i_{\text{moved}} \cdot p_k^i(C^*_{\text{VRU}}) \cdot \Big[(1 - p^i_{\text{trust}}(C_{\text{Bicycle}})) \cdot p_k^i(C_{\text{Bicycle}}) + \\ + (1 - p^i_{\text{trust}}(C_{\text{Pedestrian}})) \cdot p_k^i(C_{\text{Pedestrian}})\Big] \tag{6.47}$$

where

$$p_k^i(C^*_{\text{VRU}}) = \frac{m_k^i(\{C_{\text{Bicycle}}\}) + m_k^i(\{C_{\text{Pedestrian}}\})}{m_k^i(\{C_{\text{Car}}\}) + m_k^i(\{C_{\text{Truck}}\}) + m_k^i(\{C_{\text{Motorcycle}}\}) + m_k^i(\{C_{\text{Bicycle}}\}) + m_k^i(\{C_{\text{Pedestrian}}\})} \tag{6.48}$$

and where p^i_{moved} can be determined using a simple state machine or other method which keeps track if an object has moved or not. The remaining evidence that the object is a traffic object and has moved is assigned to the super set C^*_{Traffic}

$$m_k^i(\{C^*_{\text{Traffic}}\}) = p^i_{\text{moved}} \cdot p_k^i(C^*_{\text{VRU}}) \cdot \Big[(1 - p^i_{\text{trust}}(C_{\text{Car}})) \cdot p_k^i(C_{\text{Car}}) + \\ + (1 - p^i_{\text{trust}}(C_{\text{Truck}})) \cdot p_k^i(C_{\text{Truck}}) + (1 - p^i_{\text{trust}}(C_{\text{Motorcycle}})) \cdot p_k^i(C_{\text{Motorcycle}})\Big] + \\ + p^i_{\text{moved}} \cdot p_k^i(C^*_{\text{Vehicle}}) \cdot \Big[(1 - p^i_{\text{trust}}(C_{\text{Bicycle}})) \cdot p_k^i(C_{\text{Bicycle}}) + \\ + (1 - p^i_{\text{trust}}(C_{\text{Pedestrian}})) \cdot p_k^i(C_{\text{Pedestrian}})\Big]. \tag{6.49}$$

The remaining evidence that the object is a vehicle or VRU, disregarding that the object has moved, is assigned to the BBA of the respective super set in union with the set $C_{\text{Stationary}}$. For C^*_{Vehicle} this results in

$$m_k^i(\{C^*_{\text{Vehicle}}, C_{\text{Stationary}}\}) = (1 - p^i_{\text{moved}}) \cdot p_k^i(C^*_{\text{Vehicle}}) \cdot \Big[(1 - p^i_{\text{trust}}(C_{\text{Car}})) \cdot p_k^i(C_{\text{Car}}) + \\ + (1 - p^i_{\text{trust}}(C_{\text{Truck}})) \cdot p_k^i(C_{\text{Truck}}) + (1 - p^i_{\text{trust}}(C_{\text{Motorcycle}})) \cdot p_k^i(C_{\text{Motorcycle}})\Big] \tag{6.50}$$

and for C^*_{VRU}

$$m_k^i(\{C^*_{\text{VRU}}, C_{\text{Stationary}}\}) = (1 - p^i_{\text{moved}}) \cdot p_k^i(C^*_{\text{VRU}}) \cdot \Big[(1 - p^i_{\text{trust}}(C_{\text{Bicycle}})) \cdot p_k^i(C_{\text{Bicycle}}) + \\ + (1 - p^i_{\text{trust}}(C_{\text{Pedestrian}})) \cdot p_k^i(C_{\text{Pedestrian}})\Big]. \tag{6.51}$$

All remaining evidence, including $p_k^i(C_{\text{Other}})$ from the original classification vector c_k^i, is assigned to the BBA for all sets of the frame of discernment Θ, representing complete ignorance about object classification:

$$\begin{aligned} m_k^i(\Theta) = 1 - \Big[& m_k^i(\{C_{\text{Car}}\}) + m_k^i(\{C_{\text{Truck}}\}) + m_k^i(\{C_{\text{Motorcycle}}\}) + m_k^i(\{C_{\text{Bicycle}}\}) + \\ & + m_k^i(\{C_{\text{Pedestrian}}\}) + m_k^i(\{C_{\text{Stationary}}\}) + m_k^i(\{C^*_{\text{Vehicle}}\}) + m_k^i(\{C^*_{\text{VRU}}\}) + \\ & + m_k^i(\{C^*_{\text{Traffic}}\}) + m_k^i(\{C^*_{\text{Vehicle}}, C_{\text{Stationary}}\}) + m_k^i(\{C^*_{\text{VRU}}, C_{\text{Stationary}}\}) \Big]. \end{aligned} \tag{6.52}$$

Note that evidence for $p_k^i(C_{\text{Other}})$ is implied as the evidence for the other classes in $\mathbf{c}$ have already been fully assigned.

Fusion with the Dempster-Shafer Rule of Combination

Once the BBAs for the input sensor i are assigned, then the Dempster-Shafer rule of combination is used to fuse the sensor's evidence with the previous classification evidence for the global object from time $k-1$. Note that only the evidence for the defined sets in 2^{Θ}_{r} are assigned, as a combination with the undefined evidences from 2^{Θ} (which are all 0) is not necessary. The following DST rule of combination formulas must be evaluated:

$$m_k^{\text{G}}(\{C_{\text{Car}}\}) = \frac{\sum_{A\cap B = C_{\text{Car}}} m_{k-1}^{\text{G}}(A)\, m_k^i(B)}{1 - \sum_{A\cap B=\varnothing} m_{k-1}^{\text{G}}(A)\, m_k^i(B)}$$

$$m_k^{\text{G}}(\{C_{\text{Truck}}\}) = \frac{\sum_{A\cap B = C_{\text{Truck}}} m_{k-1}^{\text{G}}(A)\, m_k^i(B)}{1 - \sum_{A\cap B=\varnothing} m_{k-1}^{\text{G}}(A)\, m_k^i(B)}$$

$$m_k^{\text{G}}(\{C_{\text{Motorcycle}}\}) = \frac{\sum_{A\cap B = C_{\text{Motorcycle}}} m_{k-1}^{\text{G}}(A)\, m_k^i(B)}{1 - \sum_{A\cap B=\varnothing} m_{k-1}^{\text{G}}(A)\, m_k^i(B)}$$

$$m_k^{\text{G}}(\{C_{\text{Bicycle}}\}) = \frac{\sum_{A\cap B = C_{\text{Bicycle}}} m_{k-1}^{\text{G}}(A)\, m_k^i(B)}{1 - \sum_{A\cap B=\varnothing} m_{k-1}^{\text{G}}(A)\, m_k^i(B)}$$

$$m_k^{\text{G}}(\{C_{\text{Pedestrian}}\}) = \frac{\sum_{A\cap B = C_{\text{Pedestrian}}} m_{k-1}^{\text{G}}(A)\, m_k^i(B)}{1 - \sum_{A\cap B=\varnothing} m_{k-1}^{\text{G}}(A)\, m_k^i(B)}$$

$$m_k^{\text{G}}(\{C_{\text{Stationary}}\}) = \frac{\sum_{A\cap B = C_{\text{Stationary}}} m_{k-1}^{\text{G}}(A)\, m_k^i(B)}{1 - \sum_{A\cap B=\varnothing} m_{k-1}^{\text{G}}(A)\, m_k^i(B)}$$

$$m_k^{\text{G}}(\{C^*_{\text{Vehicle}}\}) = \frac{\sum_{A\cap B = C_{\text{Car}}, C_{\text{Truck}}, C_{\text{Motorcycle}}} m_{k-1}^{\text{G}}(A)\, m_k^i(B)}{1 - \sum_{A\cap B=\varnothing} m_{k-1}^{\text{G}}(A)\, m_k^i(B)}$$

$$m_k^{\text{G}}(\{C^*_{\text{VRU}}\}) = \frac{\sum_{A\cap B = C_{\text{Bicycle}}, C_{\text{Pedestrian}}} m_{k-1}^{\text{G}}(A)\, m_k^i(B)}{1 - \sum_{A\cap B=\varnothing} m_{k-1}^{\text{G}}(A)\, m_k^i(B)}$$

$$m_k^{\text{G}}(\{C^*_{\text{Traffic}}\}) = \frac{\sum_{A\cap B = C_{\text{Car}}, C_{\text{Truck}}, C_{\text{Motorcycle}}, C_{\text{Bicycle}}, C_{\text{Pedestrian}}} m_{k-1}^{\text{G}}(A)\, m_k^i(B)}{1 - \sum_{A\cap B=\varnothing} m_{k-1}^{\text{G}}(A)\, m_k^i(B)}$$

$$m_k^{\mathrm{G}}(\{C^*_{\mathrm{Vehicle}}, C_{\mathrm{Stationary}}\}) = \frac{\sum_{A\cap B=C_{\mathrm{Car}},C_{\mathrm{Truck}},C_{\mathrm{Motorcycle}},C_{\mathrm{Stationary}}} m_{k-1}^{\mathrm{G}}(A)\, m_k^i(B)}{1-\sum_{A\cap B=\varnothing} m_{k-1}^{\mathrm{G}}(A)\, m_k^i(B)}$$

$$m_k^{\mathrm{G}}(\{C^*_{\mathrm{VRU}}, C_{\mathrm{Stationary}}\}) = \frac{\sum_{A\cap B=C_{\mathrm{Bicycle}},C_{\mathrm{Pedestrian}},C_{\mathrm{Stationary}}} m_{k-1}^{\mathrm{G}}(A)\, m_k^i(B)}{1-\sum_{A\cap B=\varnothing} m_{k-1}^{\mathrm{G}}(A)\, m_k^i(B)}$$

$$m_k^{\mathrm{G}}(\Theta) = \frac{\sum_{A\cap B=\Theta} m_{k-1}^{\mathrm{G}}(A)\, m_k^i(B)}{1-\sum_{A\cap B=\varnothing} m_{k-1}^{\mathrm{G}}(A)\, m_k^i(B)}.$$

After fusion, $m_k^{\mathrm{G}}(C)$ contains the newly fused evidence for all of the BBAs in 2_{r}^{Θ}. Refer to Table 6.1 for an overview of the reduced power set intersections, useful for evaluating the Dempster-Shafer rule of combination expressions above.

Classification Vector Output from DST Fusion

From the Dempster-Shafer formulation of object classification a conversion must be made to normalized vector of classification probabilities. Similar to Section 5.2 as done with the existence probability, the Pignistic transformation [247] is used in order to properly divide the evidence of the reduced power set 2_{r}^{Θ} to the fused classification probabilities $p_k^{\mathrm{G}}(C_i)$. The Pignistic transformation of (5.66) is slightly modified such that it does not include the element Θ, resulting in

$$\mathrm{BetP}(A) = \sum_{B\subseteq\Theta, B\neq\Theta} \frac{|A\cap B|}{|B|} m(B) \tag{6.53}$$

where $m^{\mathrm{G}}(\Theta)$ is interpreted to be equivalent to $p_k^{\mathrm{G}}(C_{\mathrm{Other}})$. Generalizing (6.53) results in evaluating the follow expression for obtaining the fused classification probabilities:

$$p_k^{\mathrm{G}}(C_i) = \mathrm{BetP}(\{C_i\}) = \sum_{B\subseteq\Theta, B\neq\Theta} \frac{|C_i\cap B|}{|B|} m_k^{\mathrm{G}}(B). \tag{6.54}$$

Evaluating this expression for all of the defined classes results in

$$\begin{aligned} p_k^{\mathrm{G}}(C_{\mathrm{Car}}) = {} & m_k^{\mathrm{G}}(\{C_{\mathrm{Car}}) + \frac{1}{3} m_k^{\mathrm{G}}(\{C^*_{\mathrm{Vehicle}}) + \frac{1}{5} m_k^{\mathrm{G}}(\{C^*_{\mathrm{Traffic}}) + \\ & + \frac{1}{4} m_k^{\mathrm{G}}(\{C^*_{\mathrm{Vehicle}}, C_{\mathrm{Stationary}}\}) \end{aligned} \tag{6.55}$$

$$\begin{aligned} p_k^{\mathrm{G}}(C_{\mathrm{Truck}}) = {} & m_k^{\mathrm{G}}(\{C_{\mathrm{Truck}}) + \frac{1}{3} m_k^{\mathrm{G}}(\{C^*_{\mathrm{Vehicle}}) + \frac{1}{5} m_k^{\mathrm{G}}(\{C^*_{\mathrm{Traffic}}) + \\ & + \frac{1}{4} m_k^{\mathrm{G}}(\{C^*_{\mathrm{Vehicle}}, C_{\mathrm{Stationary}}\}) \end{aligned} \tag{6.56}$$

$$\begin{aligned} p_k^{\mathrm{G}}(C_{\mathrm{Motorcycle}}) = {} & m_k^{\mathrm{G}}(\{C_{\mathrm{Motorcycle}}) + \frac{1}{3} m_k^{\mathrm{G}}(\{C^*_{\mathrm{Vehicle}}) + \frac{1}{5} m_k^{\mathrm{G}}(\{C^*_{\mathrm{Traffic}}) + \\ & + \frac{1}{4} m_k^{\mathrm{G}}(\{C^*_{\mathrm{Vehicle}}, C_{\mathrm{Stationary}}\}) \end{aligned} \tag{6.57}$$

$$p_k^{\mathrm{G}}(C_{\mathrm{Bicycle}}) = m_k^{\mathrm{G}}(\{C_{\mathrm{Bicycle}}) + \frac{1}{2}m_k^{\mathrm{G}}(\{C^*_{\mathrm{VRU}}) + \frac{1}{5}m_k^{\mathrm{G}}(\{C^*_{\mathrm{Traffic}}) + \\ + \frac{1}{3}m_k^{\mathrm{G}}(\{C^*_{\mathrm{VRU}}, C_{\mathrm{Stationary}}\}) \tag{6.58}$$

$$p_k^{\mathrm{G}}(C_{\mathrm{Pedestrian}}) = m_k^{\mathrm{G}}(\{C_{\mathrm{Pedestrian}}) + \frac{1}{2}m_k^{\mathrm{G}}(\{C^*_{\mathrm{VRU}}) + \frac{1}{5}m_k^{\mathrm{G}}(\{C^*_{\mathrm{Traffic}}) + \\ + \frac{1}{3}m_k^{\mathrm{G}}(\{C^*_{\mathrm{VRU}}, C_{\mathrm{Stationary}}\}) \tag{6.59}$$

$$p_k^{\mathrm{G}}(C_{\mathrm{Stationary}}) = m_k^{\mathrm{G}}(\{C_{\mathrm{Stationary}}) + \frac{1}{3}m_k^{\mathrm{G}}(\{C^*_{\mathrm{Vehicle}}, C_{\mathrm{Stationary}}\}) + \\ + \frac{1}{3}m_k^{\mathrm{G}}(\{C^*_{\mathrm{VRU}}, C_{\mathrm{Stationary}}\}) \tag{6.60}$$

where $p_k^{\mathrm{G}}(C_{\mathrm{Other}})$ is the remaining probability

$$p_k^{\mathrm{G}}(C_{\mathrm{Other}}) = 1 - \Big[p_k^{\mathrm{G}}(C_{\mathrm{Car}}) + p_k^{\mathrm{G}}(C_{\mathrm{Truck}}) + p_k^{\mathrm{G}}(C_{\mathrm{Motorcycle}}) + p_k^{\mathrm{G}}(C_{\mathrm{Bicycle}}) + \\ + p_k^{\mathrm{G}}(C_{\mathrm{Pedestrian}}) + p_k^{\mathrm{G}}(C_{\mathrm{Stationary}})\Big] \tag{6.61}$$

such that the complete classification vector sums up to 1.

6.2.2 Modeling the Trust Probability

As with the existence probability fusion, described in Section 5.2.4, a weighting factor, a trust probability $p^i_{\mathrm{trust}}(C_i)$, for each class is introduced in order to model each sensor's specific ability to correctly identify that particular class. Again, it is assumed that different sensors will produce different a statistical meaning for their probability values, such that for a given probability P, $p_k^{\mathrm{S}_1}(C_i) = P \not\equiv p_k^{\mathrm{S}_2}(C_i) = P$. This requires that the input classification vectors from various sensors should be normalized to some common statistical meaning.

As with the trust probability the existence fusion, the trust probability for classification fusion is derived from experimental data in order to identify the performance of each sensor's ability to classify objects. This performance is again evaluated in form of an Receiver Operating Characteristic (ROC) curve, used to evaluated classification performance [95]. For each class C_i for a given sensor, a ROC curve is generated in order to gain an understanding of the relationship between the true positive rate and false positive rate of that sensor's ability to classify the class C_i. For each of these ROC curves, the trust probability $p^i_{\mathrm{trust}}(C_i)$ is determined by evaluating the true positive rate given a specific false positive rate f, such that

$$p^i_{\mathrm{trust}}(C_i) = \mathrm{TPR}^i_{\mathrm{FPR}=f} = \mathrm{ROC}^i_{C_i}(\mathrm{FPR} = f) \tag{6.62}$$

where $\mathrm{ROC}^i_{C_i}(\mathrm{FPR} = f)$ is the resulting ROC curve evaluated at $\mathrm{FPR} = f$ in order to produce $\mathrm{TPR}^i_{\mathrm{FPR}=f}$, which in turn is interpreted as the trust probability $p^i_{\mathrm{trust}}(C_i)$. For the complete fusion algorithm, the same f should be used on each ROC curve across all classes and all sensors such that the fusion result remains consistent given this parameter.

Discussion

Classification estimation completes the object estimation problem as defined by the object model in this thesis. This chapter demonstrated some basic ways classification can be achieved at the sensor-level and introduced a novel approach for classification fusion using the Dempster-Shafer evidence theory. The field of machine learning is quite large and hundreds, if not thousands, of different algorithms exist for classification given a data set. It is impossible to touch on even just a small portion of the possibilities. However, when choosing a classification approach at the sensor-level, it is important to understand the data for the particular sensor and choose an algorithm which has the potential to provide the best results given the data type. Point-cloud type sensors, such as laser scanner, will perform better using certain machine learning techniques than an visual sensor, such as a camera, where image processing techniques offer completely different possibilities. As with any classification approach, the diverse the training data, the better result can be achieved. With the recent improvements in memory size and cost and the added benefit of multi-core processing, it is possible to collect an immense amount of data and process it with some quite complex training methods, where some approaches, such as deep neural networks, can even provide good classification results using unsupervised learning methods. Despite these recent developments, classification results will always contain some form of uncertainty, which makes it important to output the results in a probabilistic fashion, as done here using the classification vector $\mathbf{c}$, such that further processing of the data can consider the uncertainty.

The presented fusion approach using Dempster-Shafer theory does not rely on any machine learning methods for classification fusion. Instead, the problem is formulated such that it simply boils down to a combination of probability vectors, given their uncertainty, in order to provide the best fused classification result for each detected object. The input data from the sensor-level is handled probabilistically in that the complete classification vector and its class probabilities are considered, but also each sensor's classification performance for each class is also modeled using the trust probability $p^i_{\text{trust}}(C_i)$. This should give the most confidence possible when weighting the results of the sensor-level classification vectors into the fused global object.

Table 6.1: Intersection table of the reduced power set 2_{r}^{Θ} for object classification fusion.

$\cap$	$\{C_{\mathrm{Car}}\}$	$\{C_{\mathrm{Truck}}\}$	$\{C_{\mathrm{Moto}}\}$	$\{C_{\mathrm{Bicycle}}\}$	$\{C_{\mathrm{Ped}}\}$	$\{C_{\mathrm{Stat}}\}$	$\{C^*_{\mathrm{Veh}}\}$	$\{C^*_{\mathrm{VRU}}\}$	$\{C^*_{\mathrm{Traffic}}\}$	$\{C^*_{\mathrm{Veh}}, C_{\mathrm{Stat}}\}$	$\{C^*_{\mathrm{VRU}}, C_{\mathrm{Stat}}\}$	Θ
$\{C_{\mathrm{Car}}\}$	$\{C_{\mathrm{Car}}\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\{C_{\mathrm{Car}}\}$	$\varnothing$	$\{C_{\mathrm{Car}}\}$	$\{C_{\mathrm{Car}}\}$	$\varnothing$	$\{C_{\mathrm{Car}}\}$
$\{C_{\mathrm{Truck}}\}$	$\varnothing$	$\{C_{\mathrm{Truck}}\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\{C_{\mathrm{Truck}}\}$	$\varnothing$	$\{C_{\mathrm{Truck}}\}$	$\{C_{\mathrm{Truck}}\}$	$\varnothing$	$\{C_{\mathrm{Truck}}\}$
$\{C_{\mathrm{Moto}}\}$	$\varnothing$	$\varnothing$	$\{C_{\mathrm{Moto}}\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\{C_{\mathrm{Moto}}\}$	$\varnothing$	$\{C_{\mathrm{Moto}}\}$	$\{C_{\mathrm{Moto}}\}$	$\varnothing$	$\{C_{\mathrm{Moto}}\}$
$\{C_{\mathrm{Bicycle}}\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\{C_{\mathrm{Bicycle}}\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\{C_{\mathrm{Bicycle}}\}$	$\{C_{\mathrm{Bicycle}}\}$	$\varnothing$	$\{C_{\mathrm{Bicycle}}\}$	$\{C_{\mathrm{Bicycle}}\}$
$\{C_{\mathrm{Ped}}\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\{C_{\mathrm{Ped}}\}$	$\varnothing$	$\varnothing$	$\{C_{\mathrm{Ped}}\}$	$\{C_{\mathrm{Ped}}\}$	$\varnothing$	$\{C_{\mathrm{Ped}}\}$	$\{C_{\mathrm{Ped}}\}$
$\{C_{\mathrm{Stat}}\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\{C_{\mathrm{Stat}}\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\{C_{\mathrm{Stat}}\}$	$\{C_{\mathrm{Stat}}\}$	$\{C_{\mathrm{Stat}}\}$
$\{C^*_{\mathrm{Veh}}\}$	$\{C_{\mathrm{Car}}\}$	$\{C_{\mathrm{Truck}}\}$	$\{C_{\mathrm{Moto}}\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\{C^*_{\mathrm{Veh}}\}$	$\varnothing$	$\{C^*_{\mathrm{Veh}}\}$	$\{C^*_{\mathrm{Veh}}\}$	$\varnothing$	$\{C^*_{\mathrm{Veh}}\}$
$\{C^*_{\mathrm{VRU}}\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\{C_{\mathrm{Bicycle}}\}$	$\{C_{\mathrm{Ped}}\}$	$\varnothing$	$\varnothing$	$\{C^*_{\mathrm{VRU}}\}$	$\{C^*_{\mathrm{VRU}}\}$	$\varnothing$	$\{C^*_{\mathrm{VRU}}\}$	$\{C^*_{\mathrm{VRU}}\}$
$\{C^*_{\mathrm{Traffic}}\}$	$\{C_{\mathrm{Car}}\}$	$\{C_{\mathrm{Truck}}\}$	$\{C_{\mathrm{Moto}}\}$	$\{C_{\mathrm{Bicycle}}\}$	$\{C_{\mathrm{Ped}}\}$	$\varnothing$	$\{C^*_{\mathrm{Veh}}\}$	$\{C^*_{\mathrm{VRU}}\}$	$\{C^*_{\mathrm{Traffic}}\}$	$\{C^*_{\mathrm{Veh}}\}$	$\{C^*_{\mathrm{VRU}}\}$	$\{C^*_{\mathrm{Traffic}}\}$
$\{C^*_{\mathrm{Veh}}, C_{\mathrm{Stat}}\}$	$\{C_{\mathrm{Car}}\}$	$\{C_{\mathrm{Truck}}\}$	$\{C_{\mathrm{Moto}}\}$	$\varnothing$	$\varnothing$	$\{C_{\mathrm{Stat}}\}$	$\{C^*_{\mathrm{Veh}}\}$	$\varnothing$	$\{C^*_{\mathrm{Veh}}\}$	$\{C^*_{\mathrm{Veh}}, C_{\mathrm{Stat}}\}$	$\{C_{\mathrm{Stat}}\}$	$\{C^*_{\mathrm{Veh}}, C_{\mathrm{Stat}}\}$
$\{C^*_{\mathrm{VRU}}, C_{\mathrm{Stat}}\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\{C_{\mathrm{Bicycle}}\}$	$\{C_{\mathrm{Ped}}\}$	$\{C_{\mathrm{Stat}}\}$	$\varnothing$	$\{C^*_{\mathrm{VRU}}\}$	$\{C^*_{\mathrm{VRU}}\}$	$\{C_{\mathrm{Stat}}\}$	$\{C^*_{\mathrm{VRU}}, C_{\mathrm{Stat}}\}$	$\{C^*_{\mathrm{VRU}}, C_{\mathrm{Stat}}\}$
Θ	$\{C_{\mathrm{Car}}\}$	$\{C_{\mathrm{Truck}}\}$	$\{C_{\mathrm{Moto}}\}$	$\{C_{\mathrm{Bicycle}}\}$	$\{C_{\mathrm{Ped}}\}$	$\{C_{\mathrm{Stat}}\}$	$\{C^*_{\mathrm{Veh}}\}$	$\{C^*_{\mathrm{VRU}}\}$	$\{C^*_{\mathrm{VRU}}\}$	$\{C^*_{\mathrm{Veh}}, C_{\mathrm{Stat}}\}$	$\{C^*_{\mathrm{VRU}}, C_{\mathrm{Stat}}\}$	Θ

7 Evaluation

The proposed high-level sensor data fusion architecture presented in the previous chapters are evaluated using real data from a prototype test vehicle designed for highly automated driving on the highways. Two main evaluation scenarios are considered: object estimation accuracy with ground truth data and detection rate performance using labeled scenes. For each of the two scenarios, different metrics are used to evaluated the performance of the fused global object list compared to the results of each individual sensor.

The following sections describe the test vehicle and test setup used during evaluation and present the results for the two evaluation scenarios.

7.1 Test Vehicle and Sensor Configuration

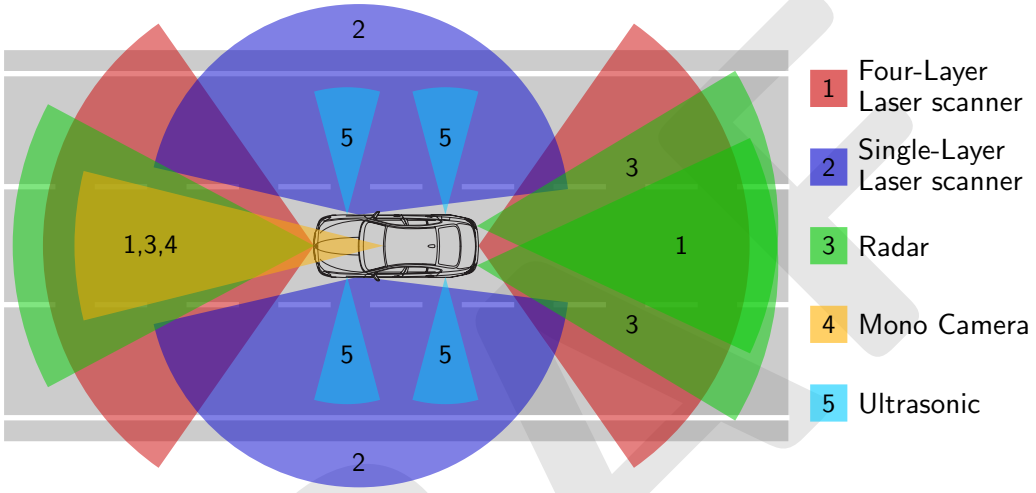
The test vehicle is a 2011 BMW 528i outfitted with several prototype sensors by the BMW Group Research and Technology in order to achieve full surround environment perception. In addition to these sensors, the vehicle is also equipped with a differential Global Positioning System (GPS) in order to obtain highly accurate global positioning. Prototype computers process all of the data from the sensors and control the actuators for vehicle control. This vehicle is used for research in Advanced Driver Assistance Systems (ADAS) and is capable of highly automated driving on highways, as described in [312]. Figure 7.1 shows a picture of the test vehicle along with its sensor configuration and system architecture. Note that all twelve of the sensors are integrated into the body of the vehicle, thereby realizing a close-to-production integration of the automated driving technology.

The research hardware for development is integrated into the trunk of the test vehicle. The system's hardware architecture is shown in Figure 7.1d. The processing of the system is distrusted across two computing platforms: an industry-grade PC and a real-time embedded prototyping computer (RTEPC). The environment perception sensors, GPS and vehicle data and actuators are connected to these two computing platforms either via Controller Area Network (CAN) or Ethernet. An Ethernet connection allows for communication between the two computers. The PC handles most of the sensor data processing, including some of the senser-level processing as well as the described fusion-level algorithms presented in this thesis, and also contains the database for the high-precision maps. The RTEPC platform processes the application-level algorithms which have real-time constraints, which are vehicle localization, maneuver planning and trajectory planning and vehicle control.

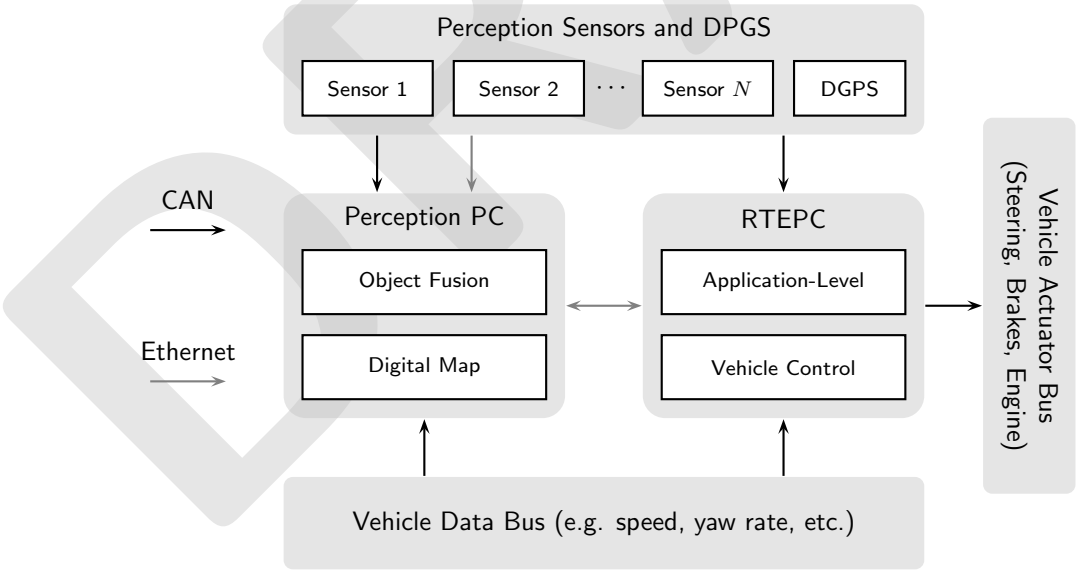
Each area of the vehicle is observed by at least two sensors of different measurement principles, achieving the redundancy and reliability required for highly automated driving. The front of the vehicle features three sensors, each of a different measurement principle. A four-layer laser scanner with a field of view of $\pm 55°$ and maximum range between 150 m-200 m offers object detection with a large field-of-view. In addition to object detection, the four-layer laser scanner can also be used for lane marking and road boundary detection. A

(a)

(b)



(c)



(d)

Figure 7.1: BMW 5 Series test vehicle (a) with research development platform in the trunk (b) and sensor configuration for surround environment perception (c). The vehicle's system architecture is shown in (d).

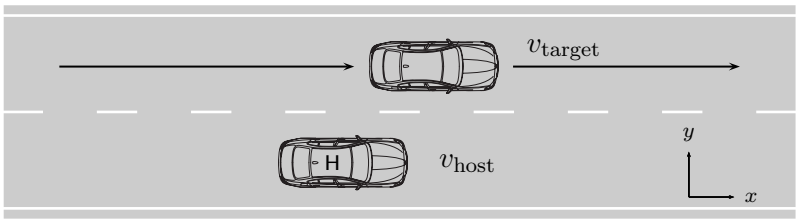


Figure 7.2: Illustration of the evaluated overtaking scenario on the test track, where the target vehicle with a speed of v_{target} overtakes the host vehicle with a speed of v_{host}.

radar sensor with a field-of-view of around $\pm 25°$ and a maximum range of 250 m is used for reliable object detection, even under partial occlusions and in various weather conditions. The radar sensor used is very similar to the type of radar found in production vehicles for Active Cruise Control (ACC). In order to achieve high reliability in object classification, a camera sensor with a field-of-view of $\pm 20°$ and a maximum range of ca. 120 m is used. The camera sensor is also capable of reliable lane marking detection. The camera sensor used is similar to the type used in production vehicles for Lane Departure Warning (LDW) or lane keeping systems.

Each side of the vehicle is covered by three sensors with two different measurement principles. A wide-angle single-layer laser scanner with a field-of-view of $\pm 80°$ and an effective range of around 80 m-100 m is used to detect objects in neighboring lanes. This laser scanner partially overlaps with the laser scanners in the front and rear, ensuring complete 360° perception coverage. The laser scanner is supported by two ultrasonic sensors on each side, aimed at improving the reliability of object detection for objects that find themselves directly next to the vehicle. The ultrasonic sensors are similar to the ones used in production vehicles for park assist applications.

The rear area of the vehicle is perceived using three sensors with two different measurement principles. The primary sensors for rear perception are two automotive radar sensors, similar to the one mounted in the front of the vehicle. These two sensors provide reliable perception of moving objects up to 150 m-200 m. This allows for early detection of objects with high relative velocity, which is critical for robust lane change maneuvers. The radar sensors are supported by a 4-layer laser scanner, also similar to the front-mounted sensor, and provides additional robustness and a wider field-of-view for detecting objects.

The above described sensor configuration allows for a reliable 360° surround environment perception. Some of the sensors used, such as the radar, camera and ultrasonic sensors, are already proven sensors that are in today's production vehicles. The addition of prototype laser scanner sensors further increase reliability in order to realize a thorough research platform for automated driving applications.

7.2 Overtaking Maneuver with Ground Truth

The algorithms for object detection and sensor data fusion are evaluated on a closed test track using two prototype vehicles for testing driver assistance systems, as described in the previous section. The following sections will evaluate the state estimation, existence probability and classification against a single known vehicle in the environment on the test track. An overtaking maneuver is performed for evaluation such that the target vehicle

being perceived is observed by all sensors of the host vehicle. The overtaking maneuver is shown in Figure 7.2. The progress of the state estimation, existence probability and classification will be shown for a single overtaking maneuver with $v_{\text{host}} = 80\,\text{km/h}$ and $v_{\text{target}} = 100\,\text{km/h}$. Additional overtaking maneuvers will be evaluated statistically in order to evaluate the errors of the state estimation over several overtaking scenarios with varying parameters.

7.2.1 Ground Truth Calculation

Both vehicles are equipped with a highly accurate global positioning system, consisting of an integrated carrier-phase differential GPS and inertial measurement unit, providing a global positioning accuracy of 2 cm and velocity accuracy of 0.014 m/s [249]. In order to evaluate the algorithms, a ground truth trajectory of the target vehicle relative to the host vehicle is calculated using this highly accurate position system in both vehicles. Global positioning data is recorded from both vehicles and a temporal and spatial synchronization of the data is carried out in order to obtain a ground truth position and velocity of the target vehicle in the host vehicle's coordinate system. The target vehicle's ground truth position and velocity is then used to evaluate the object detection algorithms from the sensors and the proposed high-level fusion architecture and its algorithms.

7.2.2 State Estimation

The estimation of the state vector is evaluated against the ground truth trajectory and velocity of the target vehicle. The information matrix fusion algorithm is implemented as the global fusion algorithm using a sensor-to-global fusion strategy. For the ground truth overtaking scenarios, the orientation and yaw rate of the target vehicle are assumed to have a constant value of 0 and will therefore not be evaluated. This section will show the results for the state estimation, presented separately for the position, velocity and dimensions. Additionally the consistency of the state estimation is tested.

The proposed feature-based association algorithm was able to successfully track the target vehicle as a unique object in the vehicle's environment throughout the entire overtaking maneuver. Figure 7.3 shows the absolute errors for the positions and velocity in the x and y-directions. The fusion results are plotted against the absolute errors of each sensor individually.

Despite fluctuations in the absolute error of the individual sensors, the proposed fusion algorithm produces a consistent state estimation as the target vehicle enters and leaves the field-of-views of all of the sensors. The error is greatest as the target vehicle enters the side area of the vehicle. This is due to the fact that the sensor-level tracking algorithms for the side-viewing laser scanner do not produce optimal results, as tracking objects with large dimensions at such a close distance is a challenge for the sensor. Other than in the side area, the absolute error in the longitudinal and lateral positions remain below 0.5 m. Throughout the entire overtaking maneuver, an average absolute error of 0.23 m longitudinally and 0.08 m laterally is achieved. Similarly for the velocities, the absolute error in the longitudinal and lateral positions remains for the most part below 0.5 m/s, other than in the side area, with an average absolute error of 0.22 m/s and 0.24 m/s, respectively. Note that some of the sensors were not able to estimate all of the states of the object model

interface, for example, the camera does not estimate a lateral velocity and the ultrasonic sensors do not measure states in the longitudinal direction.

The state estimation is also consistent with respect to the estimated covariance error throughout the overtaking maneuver. The normalized estimation error squared (NEES), as described in Section 4.3.4, is evaluated against the ground truth. Figure 7.4 shows the NEES and its 95% confidence interval for the duration of the overtaking maneuver. The low NEES shows that in a real application, the sensor tracking algorithms are conservatively parametrized in order to guarantee a consistent estimation which does not overestimate the true error. Only in the side area, where the laser scanner tracking is not quite as robust, does the NEES increase and for a brief moment cross the 95% confidence interval.

The evolution of the estimation of the target vehicle's dimensions (length and width) are shown in Figure 7.5, where the real vehicle has a width of 1.86 m and a length of 4.899 m (shown in the figure with a horizontal dashed line). The width of the target vehicle is very quickly correctly estimated, as the front of the target vehicle is well observable at the beginning of the overtaking maneuver. The length of the target vehicle is eventually also accurately estimated once the length is fully observable as the target vehicle moves closes to the host vehicle and into its side area. Even after the vehicle has moved out in front of the host vehicle, the estimated length remains at the correct length, despite the fact that the length is no longer directly observable from a sensor.

Three scenes from the overtaking maneuver are shown in Figure 7.6 using the vehicle's 3D visualization software. The top row of the figure shows the sensor-level object detection results for the target vehicle whereas the bottom row shows the fused result in blue. In all of the images, the ground truth is visualized as a white wire-frame box. The first column shows a scene where the target vehicle is perceived by the rear-facing sensors, where in Figure 7.6a the orange boxes are the radar sensors' object detections and the green box that from the laser scanner sensor. It can be seen that at this range, the laser scanner is already able to get quite an accurate measurement of the length of the target vehicle. The fusion result in Figure 7.6d confirms the consistent and well estimated object. The second column shows the perception and fusion results as the target vehicle is directly in the side area of the host vehicle. The results from the side laser scanner are visualized in violet and the ultrasonic sensors in gray. Again, the fusion result shows a consistent object with proper dimensions with respect to the ground truth, despite the fact that the side laser scanner has a limited observability of the object, such as its width. Finally, the third column shows the results as the target vehicle has moved in front of the host vehicle, where again the orange box belong to the radar sensor, the green to the front laser scanner and yellow to the camera sensor. These three scenes show how the fused object is consistently detected throughout the overtaking maneuver, including the proper detection of the target vehicle's dimensions.

In addition to evaluating the results of a single overtaking maneuver, the position and velocity errors are evaluated statistically over a total of 12 overtaking scenarios with various parameters (see Appendix C.2.1 for a detailed description of the scenarios). For the statistical evaluation of the overtaking maneuvers, the Root Mean Squared Error (RMSE) of the position and velocity, as per (4.40) and (4.41), is calculated over all for the scenarios. The results are shown in Table 7.1.

For the 12 scenarios evaluated, the fusion result has an RMSE of 0.51 m for the position and 0.60 $^{\mathrm{m}}/_{\mathrm{s}}$ for the velocity, both of which are the best results when compared to every sensor individually. Only the front mounted laser scanner has a similar position accuracy

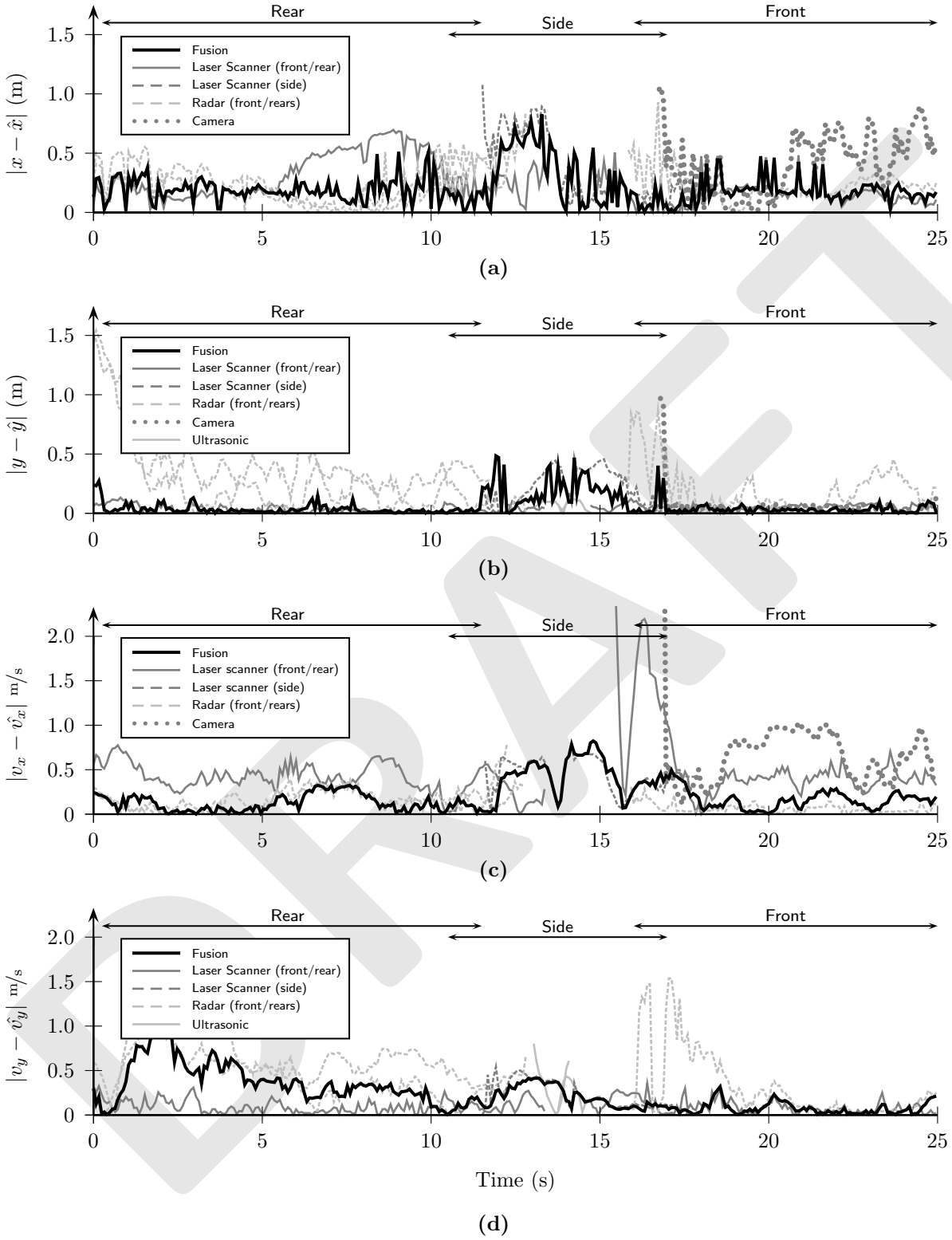


Figure 7.3: Absolute error of the longitudinal and lateral, (a) and (b), position and the longitudinal and lateral, (c) and (d), velocity during the overtaking maneuver with a host vehicle speed of $v_{\text{host}} = 80\,\text{km/h}$ and a target vehicle speed of $v_{\text{target}} = 100\,\text{km/h}$. The error of the fusion algorithm is plotted against the error of each of the individual sensors.

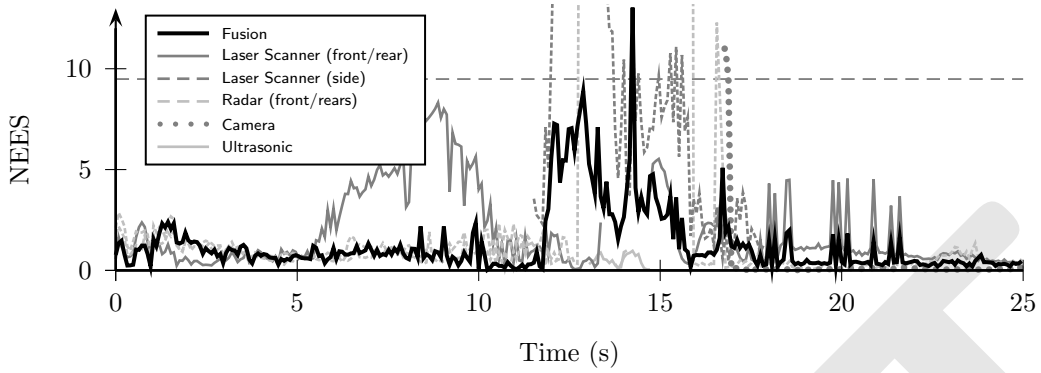


Figure 7.4: The normalized estimation error squared (NEES), demonstrating the consistency of the state estimation of the fused global object and the sensors. The 95% confidence interval is shown by the horizontal dashed line.

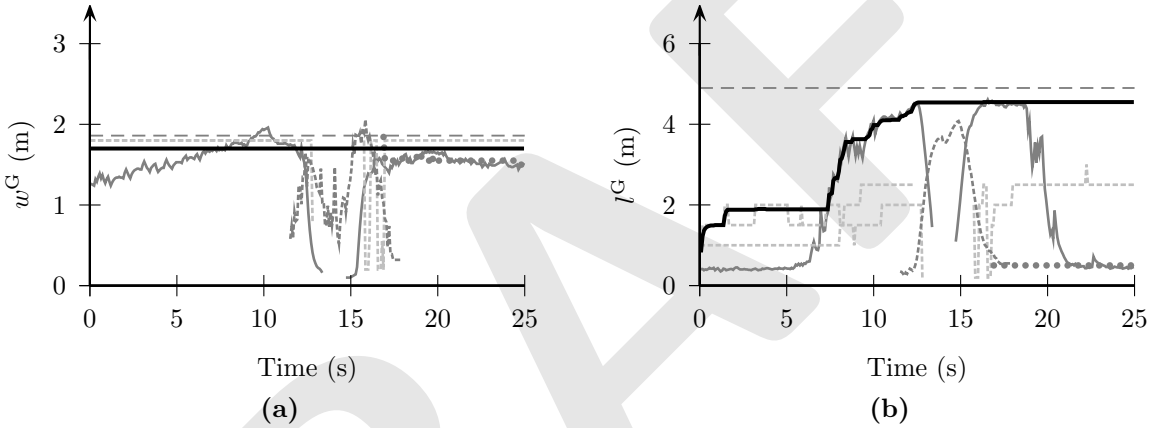


Figure 7.5: Estimation of the width (a) and length (b) of the fused global object. The true width and length is depicted by the horizontal dashed gray line. The fusion result is shown in black. Refer to Figure 7.3 for a legend identifying the various sensors.

as the fusion result. Note that for the ultrasonic sensors only the lateral position and velocity was evaluated, which, for the small area in which the sensors can actively detect objects, showed a better performance than the fusion results.

7.2.3 Existence

As presented in Chapter 5, a detected object's probability of existence is also estimated. The existence estimation is a measure of quality representing the fact that the object actually exists, as opposed to a short-term false detection from a sensor. The global fused existence probability, $p^{\mathrm{G}}(\exists \mathbf{x})$, is plotted in Figure 7.7 against the existence probability of the individual sensors. The fusion trust probability parameter, p^i_{trust}, for each of the sensors is chosen using the process described in Section 5.2.4, where a system false positive rate per iteration of 0.5 is used. See Section B.1 for details on how the specific values of p^i_{trust} were derived.

The existence probability during the overtaking maneuver very quickly converges to 1, since all three of the rear sensors immediately observe the oncoming target vehicle.

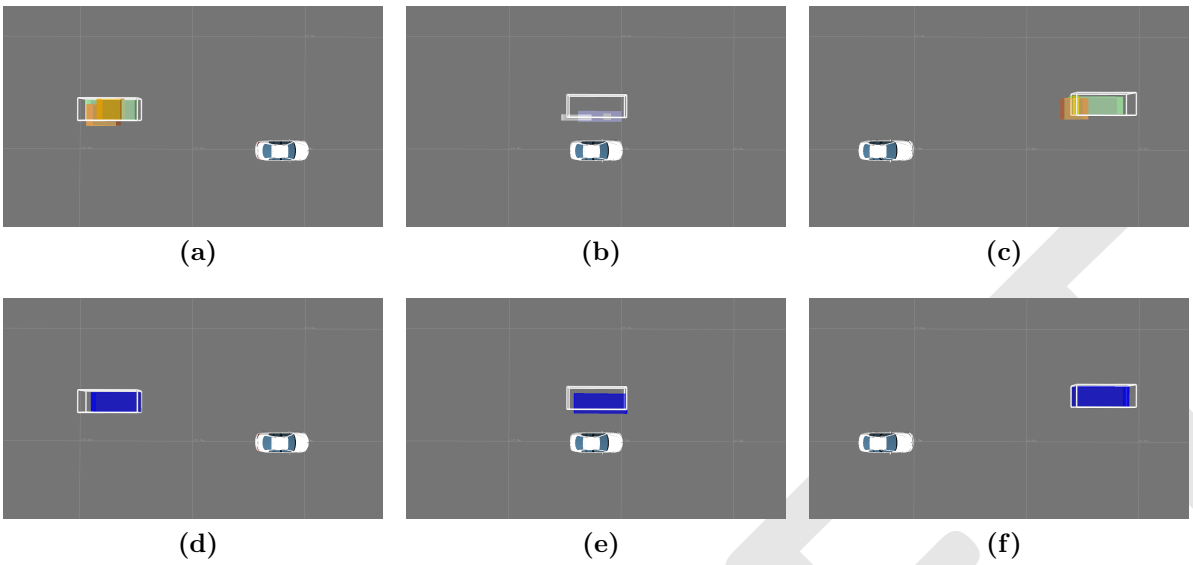


Figure 7.6: Visualization of the overtaking maneuver when the target vehicle is in different positions around the host vehicle (rear, side and front). The top row (a)-(c) shows object detection results at the sensor-level and and the bottom row (d)-(f) shows the result of the fused object. The ground truth of the target vehicle is shown as a wire-frame white box. Radar is shown orange, front/rear laser scanner in green, side laser scanner in violet, camera in yellow and ultrasonic in gray. The fused result is in blue.

As the target vehicle moves into the host vehicle's side area, the existence probability of the rear sensors begins to decrease as the existence probability of the side sensors begin to increase. However, throughout the transition from rear to side to front, the global existence probability estimation remains very close to 1, correctly suggesting that the target vehicle being tracked is detected by the system with a very high reliability.

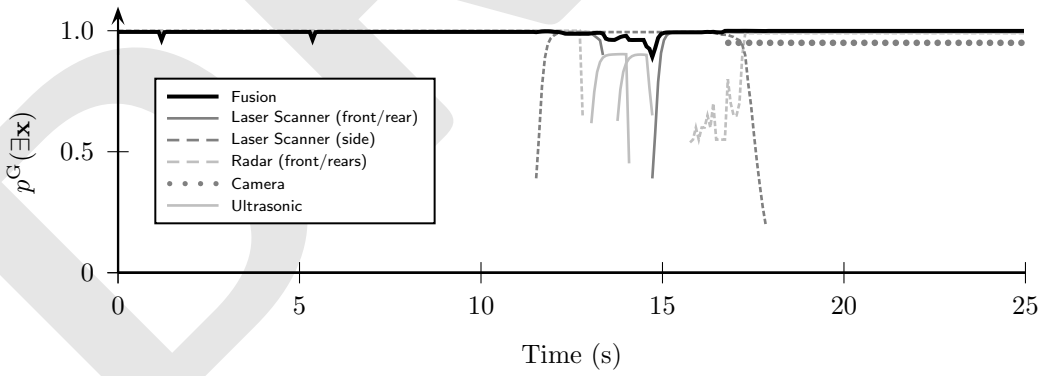


Figure 7.7: Evolution of the existence probability of the fused global object and the sensors as the target vehicle passes by the host vehicle.

7.2.4 Classification

The classification vector $\mathbf{c}^{\mathrm{G}}$ produces an estimation of the vehicle's class, where for this overtaking scenario the target vehicle is of class C_{Car}. The results of classification esti-

Table 7.1: Position and velocity RMSE evaluated for 12 overtaking scenarios.

Result	RMSE($\tilde{\mathbf{p}}(k)$)	RMSE($\tilde{\mathbf{v}}(k)$)	Evaluated Cycles
Fusion	**0.51**	**0.60**	**4,891**
Laser scanner front	0.51	0.75	2,488
Laser scanner rear	0.53	0.75	2,109
Laser scanner side	0.58	0.78	1,344
Radar front	1.04	0.92	2,354
Radar rear	0.69	0.71	3,734
Camera	0.89	0.79	2,077
Ultrasonic	0.32	0.49	464

mation is shown in Figure 7.8. Only classification results for C_{Car}, C_{Truck}, $C_{\text{Motorcycle}}$ and C_{Other} are shown, since the other classes $C_{\text{Pedestrian}}$, C_{Bicycle} and $C_{\text{Stationary}}$ are throughout the overtaking maneuver practically 0 and as a result irrelevant. As with the previous results for state estimation and existence, the classification estimations are also plotted against the results of each of the sensors individually. The trust parameters for the fusion of sensor-level classification results were determined using the process described in Section 6.2.2, a false positive rate of 0.2 is used. See Section B.2 for more details on the specific values of the trust probability as well as the Receiver Operating Characteristic (ROC) curves from which they were derived. The parameters for the classes $C_{\text{Pedestrian}}$, C_{Bicycle} and $C_{\text{Stationary}}$ are again omitted. Furthermore, it is important to note that some sensors have a continually changing estimation of an object's classification, such as the laser scanner, whereas other sensors use a more heuristic approach, which results in a more discrete output from the sensor's classification vector. Regardless of the classification approach at the sensor-level, the fusion algorithm described in Section 6.2 fuses the results together using the trust probability quality parameters.

The results in Figure 7.8 show that the classification fusion algorithm successfully classifies the target vehicle belonging to the class C_{Car} for the complete duration of the overtaking maneuver. The classification probability $p^{\text{G}}(C_{\text{Car}})$ converges to the maximum value of 1 in the front and rear area of the host vehicle and drops slightly as the target vehicle passes the side area. Despite this reduction in $p^{\text{G}}(C_{\text{Car}})$ in the side area, the value for C_{Car} is still the maximum probability value when considered over the complete classification vector $\mathbf{c}^{\text{G}}$. The reduction of $p^{\text{G}}(C_{\text{Car}})$ in the side area is due to the intermittent increase in $p^{\text{G}}(C_{\text{Truck}})$, as the side-viewing laser scanner's performance in classification estimation contains a fair amount of uncertainty.

7.3 Performance in Real Traffic Scenarios

In addition to the evaluation of controlled scenarios on a private test track, the proposed architecture and algorithms are also evaluated against real-world highway scenarios. Each scenario is semi-manually labeled in order to provide a ground truth for the number and location of cars, trucks and motorcycles at each processing cycle. An overview of the scenarios used for the evaluation can be found in Appendix C.

For these highway scenarios, only the detection rate and classification performance is

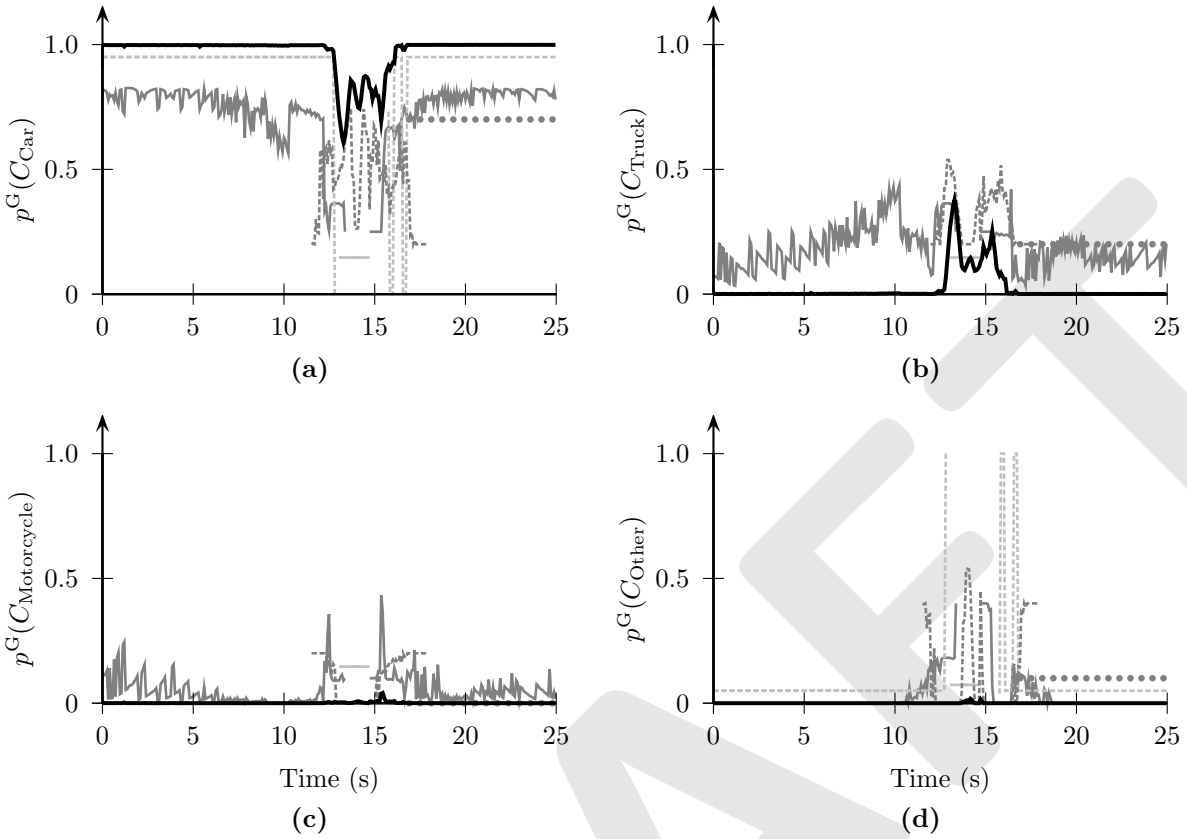


Figure 7.8: Evolution of the classification vector for the classes C_{Car} (a), C_{Truck} (b), $C_{\mathrm{Motorcycle}}$ (c) and C_{Other} (d) for the evaluated overtaking scenario. The fusion result is shown in black. Refer to Figure 7.7 for a legend identifying the various sensors.

evaluated, as it is impractical to obtain ground truth data about the surrounding traffic's true state. Therefore, only existence and classification is evaluated, as ground truth data for these attributes can be correctly generated. For each scenario, the ground truth data and the results of the proposed fusion architecture are compared to one another. In each processing cycle, a unique 1-to-1 association is made between labeled ground truth data and a resulting object from the fusion architecture. This association is made semi-manually, first by processing the data using the auction algorithm (see Section 3.3.6) and then manually correcting incorrect associations by visual inspection in a 3D visualizer (see Figure 7.9). These associations between labeled ground truth data and results from the fusion architecture are the basis on which the evaluations in the following sections are made.

The evaluation of the highway traffic scenarios is restricted to a rectangular area around the vehicle, representing the relevant field-of-view for typical driver assistance systems and automated driving. Furthermore, this rectangular area is divided into three regions: front, rear and side. These regions are illustrated in Figure 7.10. These regions are again chosen due to their relevance in typical driver assistance systems, for example the front area of the vehicle is important for ACC or lane keeping, whereas the side of the vehicle is important for lane changes. Each of these driver assistance systems and applications have different requirements on the detection performance and by evaluating the scenarios separately for

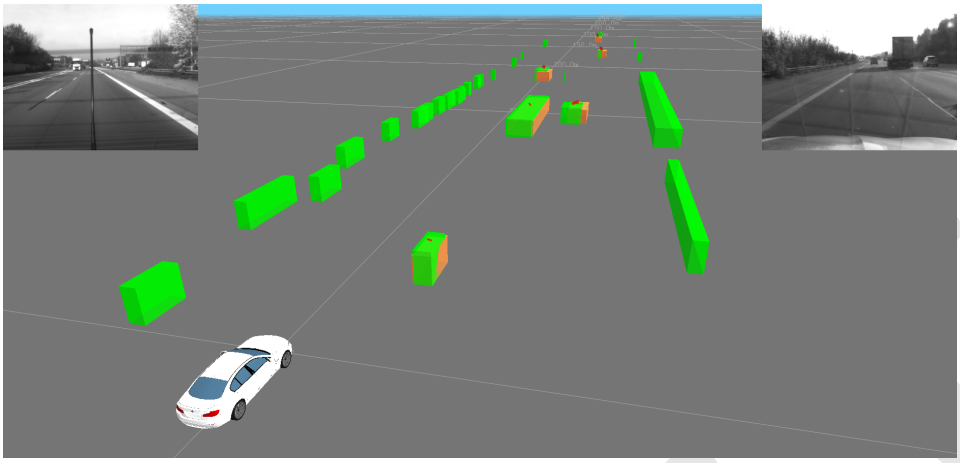


Figure 7.9: 3D visualization used for correcting the association between ground truth data (orange) and results from the fusion architecture (green). The red lines between the objects visualize the associations, which can be manually corrected using a graphical user interface.

each of these regions, much can be learned about how to apply the proper parameters in a system in order to take into account the performance of the object detection in a specific region. The parameters of the regions were chosen as follows: $d_{\text{front}} = 150\,\text{m}$, $d_{\text{rear}} = 150\,\text{m}$, $d_{\text{side}} = 50\,\text{m}$ and $w_{\text{lane}} = 4\,\text{m}$.

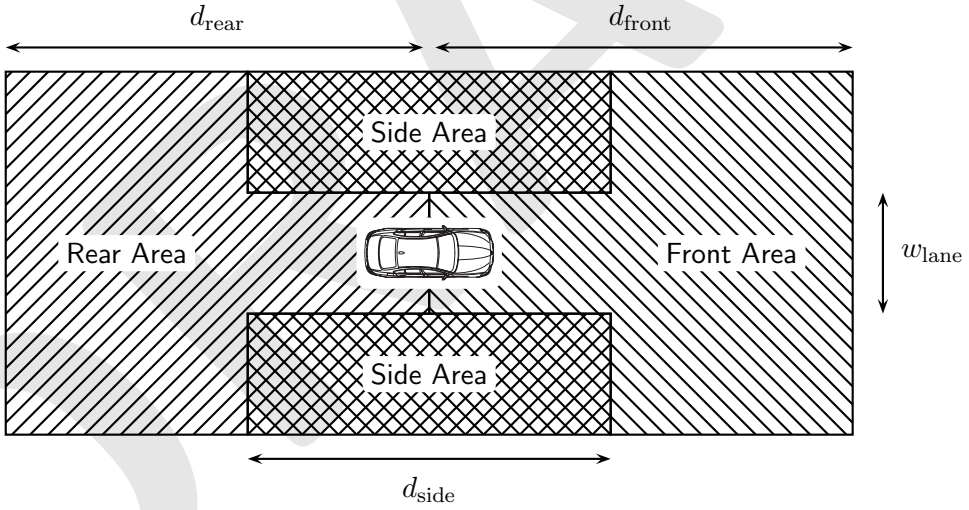


Figure 7.10: Areas around the host vehicle for which the detection rate and classification performance was evaluated. This area was chosen in order to simulate a three-lane highway, because most ADAS and automated driving applications depend on the detection of objects in these areas for various purposes.

7.3.1 Detection Rate

The detection rate of the proposed fusion architecture and its algorithms is evaluated based on the existence probability $p^{\text{G}}(\exists \mathbf{x})$ of a global object. Results are presented in form of a ROC curve, which shows the relationship between the true positive rate and false positive rate of a classifier [95]. For evaluating detection performance with the existence

probability, the classes $\exists$ and $\nexists$ are used to symbolize if an object exists or not. Existence of an object is determined by comparing the ground truth data and the results of the fusion architecture from the evaluated scenario. Since no negative instances are labeled in the ground truth data (for the problem of object existence, everything not labeled as on object is per definition a non-existent object; it is not possible to put a finite number on this value), a slightly modified version of a ROC curve is used, where the false positive rate is instead replaced by the *false positive rate per iteration*. This metric is similar to the one used in [166] for existence probability evaluation.

For a given existence probability threshold $p^{\mathrm{G}}(\exists \mathbf{x}) = P$, the true positive rate (TPR) is given by

$$\mathrm{TPR}_{p^{\mathrm{G}}(\exists \mathbf{x})=P} = \frac{\sum_{k=1}^{K} N^{\mathrm{tp}}_{p^{\mathrm{G}}(\exists \mathbf{x})=P}(k)}{\sum_{k=1}^{K} N^{\mathrm{p}}(k)} \tag{7.1}$$

where $N^{\mathrm{tp}}_{p^{\mathrm{G}}(\exists \mathbf{x})=P}(k)$ is the number of true positives (global fusion objects with $p^{\mathrm{G}}(\exists \mathbf{x}) \geq P$ which were associated with a labeled ground truth object) for a given processing cycle k, $N^{\mathrm{p}}(k)$ is the number of positives (total number of ground truth labels) for a given processing cycle k and K is the total number of processing cycles used during the evaluation. Each of these entities is evaluated only for the relevant regions, as illustrated by Figure 7.10.

Similarly, the false positive rate per iteration (FPR) is given by

$$\mathrm{FPR}_{p^{\mathrm{G}}(\exists \mathbf{x})=P} = \frac{1}{K} \sum_{k=1}^{K} N^{\mathrm{fp}}_{p^{\mathrm{G}}(\exists \mathbf{x})=P}(k) \tag{7.2}$$

where $N^{\mathrm{fp}}_{p^{\mathrm{G}}(\exists \mathbf{x})=P}(k)$ is the number of false positives (global fusion objects with $p^{\mathrm{G}}(\exists \mathbf{x}) \geq P$ which were *not* associated with a labeled ground truth object) for a given processing cycle k and again K being the total number of processing cycles used during the evaluation.

Combining the TPR and FPR for all evaluated existence probabilities $p^{\mathrm{G}}(\exists \mathbf{x}) = [0 \ldots 1]$ results in an ROC curve, defined as

$$\mathrm{ROC}_{p^{\mathrm{G}}(\exists \mathbf{x})} = \begin{bmatrix} \mathrm{TPR}_{p^{\mathrm{G}}(\exists \mathbf{x})=[0\ldots1]} \\ \mathrm{FPR}_{p^{\mathrm{G}}(\exists \mathbf{x})=[0\ldots1]} \end{bmatrix}. \tag{7.3}$$

Furthermore, the evaluation is carried out twice such that two ROC curves are evaluated for the highway scenarios, one which considers the fact that certain objects in the scenarios are occluded by objects and one which does not. For the evaluation considering occlusions, labeled ground truth objects which are occluded by other other labeled ground truth objects are not counted to $N^{\mathrm{p}}(k)$.

In order to quantify a ROC curve in a single scalar value, the Area Under the Curve (AUC), metric is used [95], which is defined as

$$\mathrm{AUC}_{p^{\mathrm{G}}(\exists \mathbf{x})} = \int_{\mathrm{FPR}=0}^{1} \mathrm{ROC}_{p^{\mathrm{G}}(\exists \mathbf{x})}. \tag{7.4}$$

The AUC can be used as a ROC performance metric in order to quantitatively compare ROC curves.

The ROC curves for the evaluated highways scenarios are shown in Figure 7.11, where each figure shows the results for the complete evaluation area, but also separately for

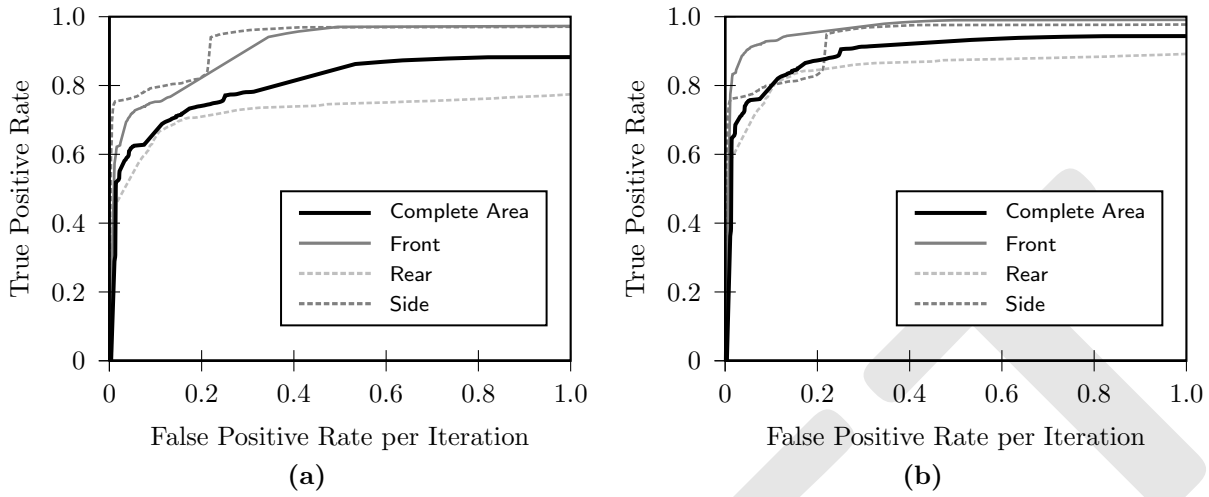


Figure 7.11: ROC curves showing the results of the detection performance using the existence probability for the evaluated highway scenarios. The ROC curve in (a) is not lenient to occluded objects whereas the ROC curve in (b) ignores occluded objects in the evaluation.

the sub-regions front, rear and side. Furthermore, the evaluation is shown once without occlusion detection (Figure 7.11a) and once with the consideration of object occlusions (Figure 7.11b). Occluded objects are defined as objects where at least three features on the rectangular object model (including the geometric middle) are in the polar shadow of another object.

For the defined evaluation area, a total of $N^{\mathrm{p}} = 70,764$ positive samples were evaluated over $K = 13,080$ processing cycles, from which 15,774 samples were occluded. Of these total samples, 36,287 were in the front area (of which 11,140 were occluded), 38,192 in the rear area (9,320 occluded) and 12,866 in the side (192 occluded). The detection rate performance is best in the front area of the vehicle, which is to be expected as it is the only area where three sensors observe the environment. The side area performs at first glance surprisingly well, but this is also to be expected, as objects which appear in the side area have already had a long tracking history, coming either from the front or the rear of the vehicle. The rear area of the vehicle has the worst performance, but is at the same time the least relevant area for most driver assistance and automated driving applications. Evaluating the complete evaluation area while allowing for occluded objects results in a convergence of the true positive rate at about 0.88 at around a false positive rate per iteration of 0.5. Without occluded objects, the performance increases such that a true positive rate of 0.94 at a false positive rate per iteration of 0.5 is reached. The AUC for the evaluation with and without occluded objects for each of the evaluation areas is given in Table 7.2. When occluded objects are not considered in the evaluation, a roughly 11% increase in performance is achieved.

7.3.2 Classification Performance

The performance of the proposed fusion architecture with regard to its ability to correctly classify objects is evaluated using the global object's classification vector $\mathbf{c}^{\mathrm{G}}$. For the evaluated highway scenarios, only car, truck and motorcycle objects were recorded and therefore

Table 7.2: AUC of the detection rate performance for the evaluated highway scenarios.

$\mathrm{AUC}_{p^{\mathrm{G}}(\exists \mathbf{x})}$	Complete Area	Front	Rear	Side
With Occlusions	0.803	0.906	0.720	0.927
Without Occlusions	0.894	0.967	0.847	0.934

the evaluation will only consider the results for classifying these types of objects. Additionally, the super-class C^{*}_{Vehicle}, where $p^{\mathrm{G}}(C^{*}_{\mathrm{Vehicle}}) = p^{\mathrm{G}}(C_{\mathrm{Car}}) + p^{\mathrm{G}}(C_{\mathrm{Truck}}) + p^{\mathrm{G}}(C_{\mathrm{Motorcycle}})$, is also evaluated.

As with the evaluation of the detection rate using the existence probability, ROC curves are used in order to evaluate the classification performance of the proposed fusion architecture, where each class is evaluated separately. The true positive rate is evaluated in a similar manner as with the detection rate performance, such that for a given classification probability threshold $p^{\mathrm{G}}(C_i) = P$, the true positive rate (TPR) is given by

$$\mathrm{TPR}_{p^{\mathrm{G}}(C_i)=P} = \frac{\sum_{k=1}^{K} N^{\mathrm{tp}}_{p^{\mathrm{G}}(C_i)=P}(k)}{\sum_{k=1}^{K} N^{\mathrm{p}}_{C_i}(k)} \tag{7.5}$$

where $N^{\mathrm{tp}}_{p^{\mathrm{G}}(C_i)=P}(k)$ is the number of true positives for the class C_i where the classification probability $p^{\mathrm{G}}(C_i) \geq P$ for a given processing cycle k, $N^{\mathrm{p}}_{C_i}(k)$ is the number of positives (ground truth labels) for the given class C_i and K is the total number of cycles evaluated.

Unlike with the detection rate evaluation, the false positive rate for the classification evaluation is calculated using the traditional formulation, where for a given classification probability threshold $p^{\mathrm{G}}(C_i) = P$, the false positive rate (FPR) is given by

$$\mathrm{FPR}_{p^{\mathrm{G}}(C_i)=P} = \frac{\sum_{k=1}^{K} N^{\mathrm{fp}}_{p^{\mathrm{G}}(C_i)=P}(k)}{\sum_{k=1}^{K} N^{\mathrm{n}}_{C_i}(k)} \tag{7.6}$$

where $N^{\mathrm{fp}}_{p^{\mathrm{G}}(C_i)=P}(k)$ is the number of false positives for the class C_i where the classification probability $p^{\mathrm{G}}(C_i) \geq P$ for a given processing cycle k, $N^{\mathrm{n}}_{C_i}(k)$ is the number of negatives for the given class (ground truth labels which are *not* the class C_i) and K is the total number of cycles evaluated.

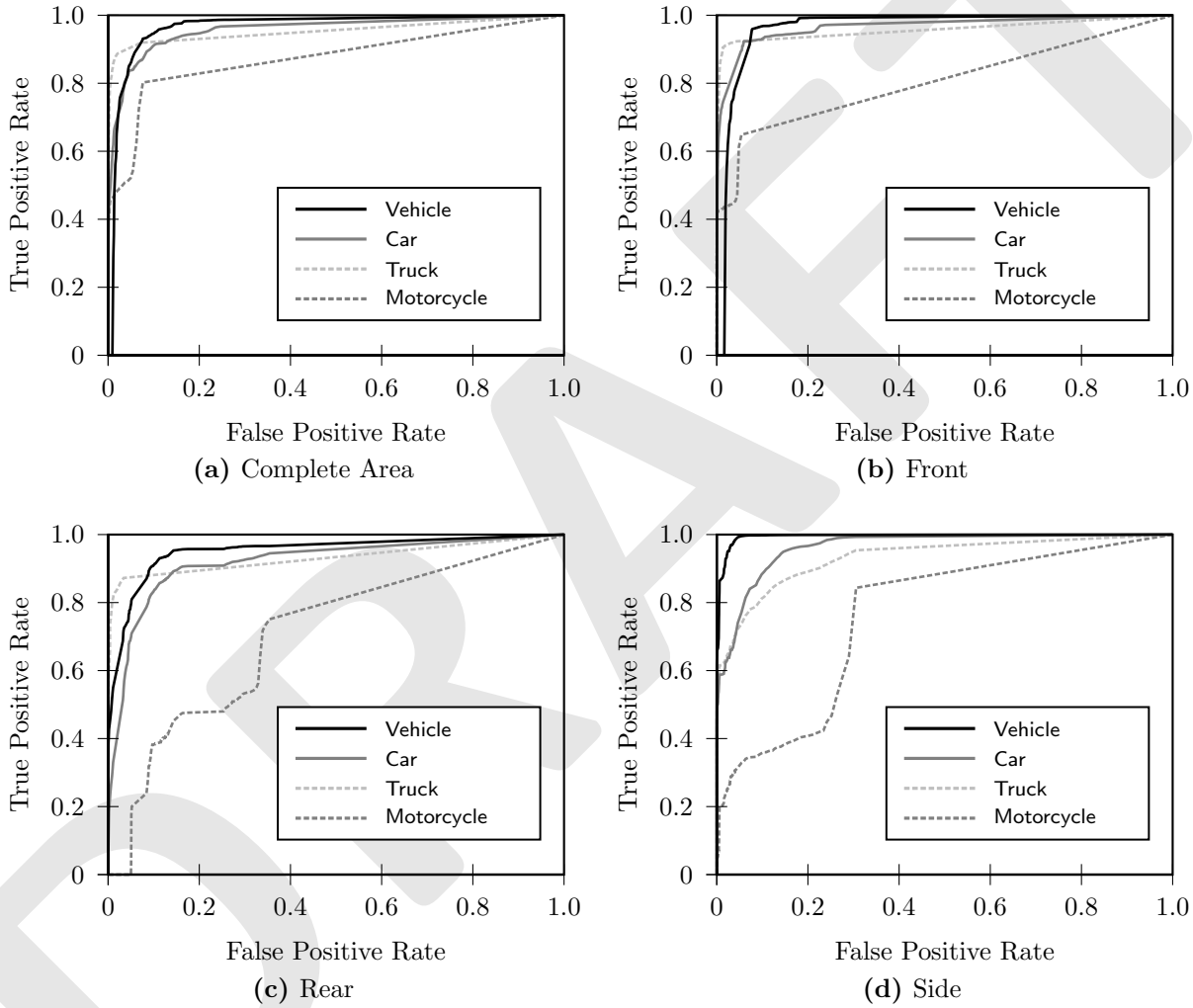
Combining the TPR and FPR for all evaluated classification probabilities $p^{\mathrm{G}}(C_i) = [0 \ldots 1]$ results in an ROC curve, defined as

$$\mathrm{ROC}_{p^{\mathrm{G}}(\exists \mathbf{x})} = \begin{bmatrix} \mathrm{TPR}_{p^{\mathrm{G}}(C_i)=[0\ldots1]} \\ \mathrm{FPR}_{p^{\mathrm{G}}(C_i)=[0\ldots1]} \end{bmatrix}. \tag{7.7}$$

The AUC is also used in the classification performance evaluation in order to quantify an ROC curve into a single value for comparison:

$$\mathrm{AUC}_{p^{\mathrm{G}}(C_i)} = \int_{\mathrm{FPR}=0}^{1} \mathrm{ROC}_{p^{\mathrm{G}}(C_i)}. \tag{7.8}$$

The ROC curves for each of the evaluated areas are shown in Figure 7.12. For the complete evaluation area, a total of $N^{\mathrm{p}}_{C^{*}_{\mathrm{Vehicle}}} = 64,987$ positive vehicle samples were evaluated (50,168 of which were C_{Car}, 12,503 C_{Truck} and 2,316 $C_{\mathrm{Motorcycle}}$). For samples divide



(a) Complete Area **(b)** Front **(c)** Rear **(d)** Side

Figure 7.12: ROC curves for the classification performance for the classes C_{Car}, C_{Truck}, $C_{\text{Motorcycle}}$ and the super-class C^{*}_{Vehicle} in each of the evaluated areas around the host vehicle.

themselves into the separate evaluation areas as follows: 36,138 in the front (27,177 C_{Car}, 7,698 C_{Truck} and 1,263 $C_{\text{Motorcycle}}$), 30,993 in the rear (23,166 C_{Car}, 7,008 C_{Truck} and 819 $C_{\text{Motorcycle}}$) and 12,728 in the side (8,189 C_{Car}, 3,875 C_{Truck} and 664 $C_{\text{Motorcycle}}$). Note that only global fusion objects which were successfully associated with a ground truth label were evaluated (hence the lower amount of a positive samples compared to the detection rate performance evaluation). As should be expected, the performance of the super-class C^*_{Vehicle} shows a better performance than each of the individual classes, demonstrating that the proposed classification algorithms are quite successful in identifying relevant traffic objects in highways scenarios. Of the three individual classes, C_{Car} has the best performance with C_{Truck} only slightly behind. This demonstrates that the classification performance is very good in identifying the two most common types of vehicles found on highways. Classification of $C_{\text{Motorcycle}}$, however, does not perform as well. This is mainly due to the fact that most of the sensors do not distinguish between motorcycle and bicycle classes, where the evidence for these two classes was split evenly between the evidences for motorcycle and bicycle with the proposed Dempster-Shafer Evidence Theory (DST) fusion approach. Furthermore, comparatively less training samples were used in order to train some of the sensors' classification algorithms for the less common motorcycle object type. As with the detection rate performance, the classification performance is also for the same reasons quite good in the side area of the vehicle, due to the fact that the class of the object has already been determined from the front or rear areas once the object arrives in the side area. The AUC results for the classification performance is shown in Table 7.3.

Table 7.3: AUC of the classification performance for the evaluated highway scenarios.

$\text{AUC}_{p^{\text{G}}(C_i)}$	Complete Area	Front	Rear	Side
Vehicle	0.966	0.963	0.936	0.967
Car	0.957	0.966	0.919	0.963
Truck	0.953	0.958	0.931	0.957
Motorcycle	0.874	0.805	0.853	0.852

Discussion

In this chapter, the algorithms which were developed in this thesis for the proposed sensor data fusion architecture were evaluated. An evaluation with a precise ground truth target vehicle using a DGPS system showed that the proposed high-level fusion method provides quite accurate results in the state estimation, but also with the estimation of the existence and classification probabilities. On average, a positional error of about 0.5 m can be achieved. The majority of this error comes from the longitudinal position, where the lateral position error is much less. This demonstrated performance meets the requirements of ADAS and automated driving applications, where there are higher performance requirements on the lateral position than the longitudinal position, for example when matching an object onto a lane, doing lane-change prediction [317, 318] or estimating a road model using object data [264]. Furthermore, performance requirements from ADAS are higher for the front area of the host vehicle than for the side and the rear, which also matches the presented results. However, given that an error is inevitable, it is important that further

processing of object data is always done so probabilistically, such that the uncertainty in the estimation can be taken into account. Using the Normalized Estimation Error Squared (NEES) consistency measure, it was shown that the estimated state and its error covariance provide a truthful estimation of the error for application-level algorithms, which is critical for reducing false reactions of an ADAS or automated driving system.

Furthermore, the developed algorithms were evaluated for their detection and classification performance using labeled data sets from real highway traffic scenarios. Such results are also important for application-level processing. The resulting ROC curves give an impression of the performance of the algorithms with regards to the relationship between the true positive rate and false positive rate given a variation of the probability thresholds. This information can be used by application-level processing in order to define thresholds for various applications such that a desired detection rate or classification performance is guaranteed. For both the detection rate and classification performance, the front of the vehicle tends to have the best performance, which is expected and desired, as the front of the vehicle is observed by three sensors and is also the most relevant and critical area for most ADAS and automated driving applications.

It is important to note that fusion algorithms can only be as good as the input received from the sensor-level. This can been seen in the previous evaluations regarding the performance of the side area, where the estimation accuracy is reduced to the fact that the sensor-level processing using the side-viewing laser scanner is quite difficult. Therefore, it is important that data from the sensor-level is not only accurate in order to get a good fusion result, but also consistent, such that if there is an error or some uncertainty about the data, it should be properly modeled such that the fusion algorithms can properly weigh the uncertain data during the fusion process.

In general, however, the evaluation showed that the proposed sensor data fusion architecture and the developed algorithms provide a solid platform for model-based object detection for ADAS and automated driving applications. These algorithms have not only been thoroughly evaluated in this thesis, but are also being actively used by the BMW Group Research and Technology's prototype vehicle for highly automated driving on highways [312], where thousands of kilometers of successful automated driving have shown the effectiveness of the proposed architecture and algorithms for surround environment perception. Additionally, this thesis presented a complete and thorough evaluation for object detection using multiple sensors in a surround environment perception algorithms, where not only the state-based (position, velocity, etc.) aspect of an object was evaluated, but also the detection and classification performance, giving a solid result for object detection and fusion performance not yet available in the current literature.

Despite the positive results, it remains to evaluate the object detection performance in conjunction with actual ADAS or automated driving applications. To date this has been done on a case-by-case basis in an empirical fashion, but a solid methodology for understanding how the performance of an object detection algorithm directly effects the performance of a driving function is still on open topic of research. This is especially true when a driving function relies on several modules, where these modules also rely on object detection, which then further complicates the performance analysis of a complete system given these types of dependencies.

8 Conclusion and Discussion

Environment perception has been and will remain one of the main challenges in Advanced Driver Assistance Systems (ADAS) and automated driving applications, in particular as higher degrees of automation will begin to be implemented. Early systems, such as Active Cruise Control (ACC), have been developed over many generations, and with each generation, such systems are improved due to the better performance of the sensors and the sensor processing algorithms. Only in recent years are series production vehicles beginning to implement fusion algorithms, which will further improve these driver assistance systems. Sensor data fusion will become a key aspect in future ADAS and automated driving systems in order to optimally and efficiently describe the complete environment of the vehicle for not just one, but several applications simultaneously. Furthermore, recent series production systems focus quite heavily on perception in the frontal area of the vehicle, with some systems, such as lane change warning, requiring some side and rear perception. As driver assistance systems become more complex and the level of automation increases, complete surround perception will be required in order realize more complex and intelligent behaviors of the vehicle.

This thesis proposed a sensor data fusion architecture with a high-level, or object-level, approach for model-based object detection. The goal was to develop an architecture and the necessary algorithms such that complete 360° surround environment perception with up to 12 automotive-grade and vehicle body integrated sensors can be realized for automated driving applications. Chapter 2 gave an overview and analysis of different fusion architecture concepts and presented a novel high-level sensor data fusion architecture which was developed within this thesis. The remaining chapters elaborated on the developed algorithms which implemented this sensor data fusion architecture and its concepts. The evaluation in Chapter 7 showed that the proposed architecture provide good results in accuracy and detection performance in the prototype vehicle. Furthermore, the presented sensor data fusion concepts have been successfully deployed in the BMW Group Research and Technology's prototypes for highly automated driving on highways.

Several new contributions were made in this thesis to the problem of object detection and fusion using multiple sensors for ADAS and automated driving applications. The analysis of the different sensor data fusion architectures showed that the high-level fusion method chosen in the proposed architecture has many advantages in performance and practicality for the ADAS application. A complete object representation is formulated, where object state, existence and classification form the object representation. Unique object association techniques are described for the high-level fusion method, in particular how to consider existence and classification attributes in the association process, as well as dealing with extended objects and their dimensions in sensor-overlap scenarios. In the literature, a sensor-to-sensor fusion strategy has been typically applied, whereas in this thesis the sensor-to-global fusion strategy is introduced, allowing for true asynchronous processing of object data within the high-level sensor data fusion architecture. The track-to-track fusion algorithms were evaluated and compared to one another using such a sensor-

to-global fusion strategy, where it was shown that the information matrix fusion algorithm performs exactly the same as a low-level central filtering approach. For object existence, a novel, yet simple, Bayes formulation was presented for sensor-level existence estimation. At the fusion-level, a new approach for existence fusion was developed using the Dempster-Shafer evidence theory framework, where sensor performance can be modeled during the fusion process. Similarly for classification fusion, a method was also derived from the Dempster-Shafer evidence theory for fusing object class information, also considering sensor performance during fusion. Finally, the proposed algorithms for object detection and fusion were for the first time thoroughly evaluated for such a surround environment perception setup with up to 12 sensors, where the state, existence and classification performance of the object detection algorithms were evaluated.

The results showed that the proposed sensor data fusion approach works quite well in highways scenarios. However, it remains to be evaluated in more complex, urban environments, where environment perception is quite a bit more challenging. Urban environments are very dense, making it difficult to properly separate objects near one another. There are also a greater number of objects to detect, including pedestrians, bicyclists and other potentially dynamic objects, such as animals, which further pose challenges in properly detecting and classifying such objects so that a driver assistance system can properly react. At the sensor-level, there already exist many algorithms which offer solutions to some of these challenges. Grid-based algorithms have proven to be quite useful in detecting static and dynamic environments in complex, urban scenarios [11, 12, 74, 136, 152, 186, 206, 258]. The challenge in such approaches lies within the proper discrimination between static obstacles and dynamic objects and the combination of the grid-based environment perception algorithms with that of the more traditional model-based approaches. Some approaches have already been presented which attempt to consistently combine different environment model representations [117, 209, 210, 239, 287]. Additionally, the grid-based approach has also been extended for model-based object tracking, where object local grid serve as a basis for a better estimation of an object's shape and position [12, 136, 240], similar to the one-dimensional grid-based approach presented in Section 4.4 used to estimate a fused object's dimensions. A lot of research has also been done with stereo camera environment perception in urban environments using the stixel representation for object detection [92, 217], where a confidence representation [115, 218] and semantic labeling have also been shown to play a crucial role [91] in developing robust object representations in complex environments. Despite these advances at the sensor-level, the fusion of object data for complex environments for complete surround environment perception is still an open topic of research. However, it would be interesting to investigate the potential of applying some of the sensor-level techniques at the fusion-level, for example object local grid maps or semantic labeling. Furthermore, complex data association still tends to be a problem, particularly in dense environments, where a one-to-one data association scheme, as presented in this thesis, may not suffice at the fusion-level. An application of probabilistic data association techniques, as common at the sensor-level or with low-level fusion algorithms using, for example, Joint Integrated Probabilistic Data Association (JIPDA), will most likely need to also be applied at the fusion-level in order to obtain the desired performance and accuracy. Combining such techniques and further developing the proposed high-level sensor data fusion architecture could in turn also provide a reliable platform for surround environment perception in such complex, urban environments while maintaining the advantages of an object-level data abstraction in a multi-sensor configuration.

As the level of automation continues to increase in driver assistance systems and as the scenarios in which such systems are to be used becomes more complex, the need for reliable, robust and accurate environment perception will remain. It will continue to be one of the main challenges facing ADAS, autonomous driving and the robotics research community. This thesis is just a small contribution and one of many possible approaches for solving the challenges faced in environment perception and future research will build upon this work, with the continued goal of improving safety, efficiency and comfort in personal mobility.

A Synchronous Track-to-Track Fusion Algorithms

The most common track-to-track fusion algorithms studied in the literature pertain to synchronous sensor configurations. In order to avoid the correlation due to common information history, these algorithms use a sensor-to-sensor fusion strategy. If the sensors are asynchronous, they are temporally aligned for the sensor-to-sensor fusion process, as described in Section 3.1.2. These algorithms have also been the most commonly used in automotive applications. In [174], synchronous track-to-track fusion algorithms are compared to one another and their practicality and performance considered for automotive applications. This Appendix will give a brief overview of the most commonly used synchronous track-to-track fusion algorithms.

A.1 Simple Weighted Fusion

The most basic fusion algorithm takes two state estimates from sensors i and j and combines them together into a fused state. The simple weighted fusion algorithm is derived from the static linear estimation equation [16], which calculates the posterior mean $\hat{\mathbf{x}}$ in terms of the prior mean, $\overline{\mathbf{x}}$, and measurement $\mathbf{z}$

$$\hat{\mathbf{x}} = \overline{\mathbf{x}} + \mathbf{P}_{\mathbf{xz}}\mathbf{P}_{\mathbf{zz}}^{-1}\left(\mathbf{z} - \overline{\mathbf{z}}\right). \tag{A.1}$$

The posterior mean for the fusion, or global state estimate, is the expected value of the true state conditioned on the measurement information from sensors i and j

$$\hat{\mathbf{x}} \rightarrow \mathrm{E}\left[\mathbf{x} \mid Z^i, Z^j\right] = \hat{\mathbf{x}}^{\mathrm{G}}. \tag{A.2}$$

The prior mean in (A.1) can be considered to be the current state estimation from sensor i with measurement information Z^i

$$\overline{\mathbf{x}} \rightarrow \mathrm{E}\left[\mathbf{x} \mid Z^i\right] = \hat{\mathbf{x}}^i. \tag{A.3}$$

The "measurement" $\mathbf{z}$ in (A.1) can be seen as the current state estimation from sensor j

$$\mathbf{z} \rightarrow \mathrm{E}\left[\mathbf{x} \mid Z^j\right] = \hat{\mathbf{x}}^j. \tag{A.4}$$

The predicted "measurement" $\overline{\mathbf{z}}$ is the expected value of the state estimation from sensor j, but conditioned on the measurement information from the prior mean, or sensor i

$$\overline{\mathbf{z}} \rightarrow \mathrm{E}\left[\hat{\mathbf{x}}^j \mid Z^i\right] = \hat{\mathbf{x}}^i. \tag{A.5}$$

The simple weighted fusion algorithm assumes that the state estimations between sensors i and j are independent, leading to the evaluation of the covariances from (A.1)

$$\begin{aligned}\mathbf{P}_{\mathbf{xz}} &= \mathrm{E}\left[\left(\mathbf{x} - \mathrm{E}\left[\mathbf{x} \mid Z^i\right]\right)\left(\hat{\mathbf{x}}^j - \mathrm{E}\left[\hat{\mathbf{x}}^j \mid Z^i\right]\right)'\right] \\ &= \mathrm{E}\left[\tilde{\mathbf{x}}^i\left(\hat{\mathbf{x}}^j - \hat{\mathbf{x}}^i\right)'\right] = \mathrm{E}\left[\tilde{\mathbf{x}}^i\left(\tilde{\mathbf{x}}^i - \tilde{\mathbf{x}}^j\right)'\right] = \mathrm{E}\left[\tilde{\mathbf{x}}^i(\tilde{\mathbf{x}}^i)'\right] - \mathrm{E}\left[\tilde{\mathbf{x}}^i(\tilde{\mathbf{x}}^j)'\right] \\ &= \mathbf{P}^i \end{aligned} \tag{A.6}$$

and

$$\begin{aligned}\mathbf{P}_{\mathbf{zz}} &= \mathrm{E}\left[\left(\hat{\mathbf{x}}^j - \mathrm{E}\left[\hat{\mathbf{x}}^j \mid Z^i\right]\right)\left(\hat{\mathbf{x}}^j - \mathrm{E}\left[\hat{\mathbf{x}}^j \mid Z^i\right]\right)'\right] \\ &= \mathrm{E}\left[\left(\hat{\mathbf{x}}^j - \hat{\mathbf{x}}^i\right)\left(\hat{\mathbf{x}}^j - \hat{\mathbf{x}}^i\right)'\right] = \mathrm{E}\left[\left(\tilde{\mathbf{x}}^i - \tilde{\mathbf{x}}^j\right)\left(\tilde{\mathbf{x}}^i - \tilde{\mathbf{x}}^j\right)'\right] \\ &= \mathbf{P}^i + \mathbf{P}^j \end{aligned} \tag{A.7}$$

where $\tilde{\mathbf{x}}^i$ and $\tilde{\mathbf{x}}^j$ are the errors of the sensor-level state estimates

$$\tilde{\mathbf{x}}^i = \mathbf{x}^i - \hat{\mathbf{x}}^i \tag{A.8}$$

$$\tilde{\mathbf{x}}^j = \mathbf{x}^j - \hat{\mathbf{x}}^j. \tag{A.9}$$

Substituting the above expressions into the static linear estimation equation from (A.1) leads to the fused, or global, state estimate after combining the state estimations from sensors i and j

$$\hat{\mathbf{x}}^{\mathrm{G}} = \hat{\mathbf{x}}^i + \mathbf{P}^i\left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1}\left(\hat{\mathbf{x}}^j - \hat{\mathbf{x}}^i\right). \tag{A.10}$$

Rearranging the above expression as follows gives the more traditional symmetrical form of the fused state

$$\begin{aligned}\hat{\mathbf{x}}^{\mathrm{G}} &= \hat{\mathbf{x}}^i + \mathbf{P}^i\left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1}\left(\hat{\mathbf{x}}^j - \hat{\mathbf{x}}^i\right) \\ &= \hat{\mathbf{x}}^i + \mathbf{P}^i\left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1}\hat{\mathbf{x}}^j - \mathbf{P}^i\left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1}\hat{\mathbf{x}}^i \\ &= \left[\mathbf{I} - \mathbf{P}^i\left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1}\right]\hat{\mathbf{x}}^i + \mathbf{P}^i\left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1}\hat{\mathbf{x}}^j \\ &= \left[\left(\mathbf{P}^i + \mathbf{P}^j\right)\left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1} - \mathbf{P}^i\left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1}\right]\hat{\mathbf{x}}^i + \mathbf{P}^i\left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1}\hat{\mathbf{x}}^j \\ &= \left(\mathbf{P}^i + \mathbf{P}^j - \mathbf{P}^i\right)\left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1}\hat{\mathbf{x}}^i + \mathbf{P}^i\left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1}\hat{\mathbf{x}}^j \\ &= \mathbf{P}^j\left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1}\hat{\mathbf{x}}^i + \mathbf{P}^i\left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1}\hat{\mathbf{x}}^j. \end{aligned} \tag{A.11}$$

The covariance of the static linear estimation equation is

$$\mathbf{P}_{\mathbf{xx|z}} = \mathbf{P}_{\mathbf{xx}} - \mathbf{P}_{\mathbf{xz}}\mathbf{P}_{\mathbf{zz}}^{-1}\mathbf{P}_{\mathbf{zx}} \tag{A.12}$$

which when substituting the expressions for the covariances yields

$$\mathbf{P}^{\mathrm{G}} = \mathbf{P}^i - \mathbf{P}^i\left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1}\mathbf{P}^{i\prime} \tag{A.13}$$

The expression for the covariance can also be rearranged into a symmetrical form

$$\begin{aligned}\mathbf{P}^{\mathrm{G}} &= \mathbf{P}^i - \mathbf{P}^i \left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1} \mathbf{P}^{i\prime} \\ &= \mathbf{P}^i \left(\mathbf{I} - \left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1} \mathbf{P}^i\right) \\ &= \mathbf{P}^i \left[\left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1} \left(\mathbf{P}^i + \mathbf{P}^j\right) - \left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1} \mathbf{P}^i\right] \\ &= \mathbf{P}^i \left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1} \left(\mathbf{P}^i + \mathbf{P}^j - \mathbf{P}^i\right) \\ &= \mathbf{P}^i \left(\mathbf{P}^i + \mathbf{P}^j\right)^{-1} \mathbf{P}^j. \end{aligned} \tag{A.14}$$

As the name suggests, this algorithm is the simplest method for fusing two state estimates in a sensor-to-sensor fashion. However, the simple weighted fusion algorithm ignores the correlation between the estimates from two sensors due to the common process noise by assuming that the estimates are independent. In [17], it was shown by evaluating the error ellipses that the simple weighted fusion algorithm overestimates the true error of the fused state estimation, meaning that the fused covariance is smaller with respect to the true error of the estimate.

Despite the fact that this algorithm ignores the dependence between sensor data, its simplicity makes it quite a favorable algorithm in practical applications. Linzmeier uses this algorithm to fuse data from an infrared and a radar sensor for pedestrian detection [160]. He argues that the error made in ignoring the dependence is negligible in practice. Other automotive applications have also used the simple weighted fusion algorithm [126, 187].

A.2 Use of Cross-Covariance

In reality, the state estimates from two sensors i and j using a sensor-to-sensor fusion strategy are correlated due to their common process noise, therefore making the estimates dependent [17]. The simple weighted fusion algorithm did not take this dependence into account. The use of cross-covariance algorithm, as the name implies, is derived in a similar manner as the simple weighted fusion algorithm, but instead considers the dependence of the sensor-level state estimates.

From (A.1) to (A.5), the derivation of the use of cross-covariance algorithm is the same. The derivation of the covariances from (A.1) differ in that the correlation between the sensor estimates is not zero:

$$\begin{aligned}\mathbf{P}_{\mathbf{xz}} &= \mathrm{E}\left[\left(\mathbf{x} - \mathrm{E}\left[\mathbf{x} \mid Z^i\right]\right)\left(\hat{\mathbf{x}}^j - \mathrm{E}\left[\hat{\mathbf{x}}^j \mid Z^i\right]\right)'\right] \\ &= \mathrm{E}\left[\tilde{\mathbf{x}}^i\left(\hat{\mathbf{x}}^j - \hat{\mathbf{x}}^i\right)'\right] = \mathrm{E}\left[\tilde{\mathbf{x}}^i\left(\tilde{\mathbf{x}}^i - \tilde{\mathbf{x}}^j\right)'\right] \\ &= \mathrm{E}\left[\tilde{\mathbf{x}}^i(\tilde{\mathbf{x}}^i)'\right] - \mathrm{E}\left[\tilde{\mathbf{x}}^i(\tilde{\mathbf{x}}^j)'\right] \\ &= \mathbf{P}^i - \mathbf{P}^{ij} \end{aligned} \tag{A.15}$$

and

$$\begin{aligned}\mathbf{P}_{\mathbf{zz}} &= \mathrm{E}\left[\left(\hat{\mathbf{x}}^j - \mathrm{E}\left[\hat{\mathbf{x}}^j \mid Z^i\right]\right)\left(\hat{\mathbf{x}}^j - \mathrm{E}\left[\hat{\mathbf{x}}^j \mid Z^i\right]\right)'\right] \\ &= \mathrm{E}\left[\left(\hat{\mathbf{x}}^j - \hat{\mathbf{x}}^i\right)\left(\hat{\mathbf{x}}^j - \hat{\mathbf{x}}^i\right)'\right] = \mathrm{E}\left[\left(\tilde{\mathbf{x}}^i - \tilde{\mathbf{x}}^j\right)\left(\tilde{\mathbf{x}}^i - \tilde{\mathbf{x}}^j\right)'\right] \\ &= \mathbf{P}^i + \mathbf{P}^j - \mathbf{P}^{ij} - \mathbf{P}^{ji} \end{aligned} \tag{A.16}$$

where $\mathbf{P}^{ij}$ is the cross-covariance matrix between the state estimates of sensors i and j, which accounts for the common process noise between the sensor estimates.

Using these new expressions for the covariances from the static linear estimation equation yields the expression for the fusion of the state from sensors i and j using the cross-covariance

$$\hat{\mathbf{x}}^{\mathrm{G}} = \hat{\mathbf{x}}^{i} + \left(\mathbf{P}^{i} - \mathbf{P}^{ij}\right)\left[\mathbf{P}^{i} + \mathbf{P}^{j} - \mathbf{P}^{ij} - \mathbf{P}^{ji}\right]^{-1}\left(\hat{\mathbf{x}}^{j} - \hat{\mathbf{x}}^{i}\right) \tag{A.17}$$

with the fused global covariance

$$\mathbf{P}^{\mathrm{G}} = \mathbf{P}^{i} - \left(\mathbf{P}^{i} - \mathbf{P}^{ij}\right)\left[\mathbf{P}^{i} + \mathbf{P}^{j} - \mathbf{P}^{ij} - \mathbf{P}^{ji}\right]^{-1}\left(\mathbf{P}^{i} - \mathbf{P}^{ij}\right)'. \tag{A.18}$$

The cross-covariance matrix can be calculated using one of the methods described in Section 4.2.1. The algorithm is optimal if it is carried out each time a sensor updates its local state estimate and the recursive method for calculating the cross-covariance matrix is used.

The use of cross-covariance algorithm has been successfully used to fuse sensor data in automotive applications. In [42], the algorithm is used to fuse infrared and radar and radar and lidar for general obstacle detection. It has also been used to fuse data in vehicle-to-vehicle communications applications, where radar sensor data is fused with communicated object data from other vehicles [263]. However, most of these applications are simple two-sensor configurations using a basic sensor-to-sensor fusion strategy where one sensor is synchronized to the other for fusion.

A.3 Covariance Intersection

An alternative to the previous fusion methods is the covariance intersection algorithm. This method of fusion was described by Julier and Uhlmann in [132] and applied to a simultaneous localization and mapping (SLAM) application in [133]. The method assumes from the very beginning that the cross-covariance between sensors is unknown and goes about fusing two sensor estimates without explicitly calculating it, as done with the use of cross-covariance algorithm described in the previous section.

The covariance intersection algorithm is derived from a geometrical interpretation of the Kalman filter equations [132]. In general form, the Kalman filter equations can be written as follows to combine two state estimates $\hat{\mathbf{x}}^{i}$, $\mathbf{P}^{i}$ and $\hat{\mathbf{x}}^{j}$, $\mathbf{P}^{j}$ using weights $\mathbf{W}^{i}$ and $\mathbf{W}^{j}$

$$\hat{\mathbf{x}}^{\mathrm{G}} = \mathbf{W}^{i}\hat{\mathbf{x}}^{i} + \mathbf{W}^{j}\hat{\mathbf{x}}^{j} \tag{A.19}$$

$$\mathbf{P}^{\mathrm{G}} = \mathbf{W}^{i}\mathbf{P}^{i}\mathbf{W}^{i\prime} + \mathbf{W}^{i}\mathbf{P}^{ij}\mathbf{W}^{j\prime} + \mathbf{W}^{j}\mathbf{P}^{ji}\mathbf{W}^{i\prime} + \mathbf{W}^{j}\mathbf{P}^{j}\mathbf{W}^{j\prime}. \tag{A.20}$$

The weights $\mathbf{W}^{i}$ and $\mathbf{W}^{j}$ are chosen to minimize the trace of the fused covariance $\mathbf{P}^{\mathrm{G}}$ [132]. Geometrically, these equations, regardless of the choice of $\mathbf{P}^{ij}$ guarantee that the fused covariance ellipse $\mathbf{P}^{\mathrm{G}}$ is enclosed within the intersection of the covariances of the two sensor estimates $\mathbf{P}^{i}$ and $\mathbf{P}^{j}$. The fused covariance that most tightly fits into the intersection best utilizes the available information.

Using a convex combination of the covariance, the fused state estimate is combined as follows:

$$(\mathbf{P}^{\mathrm{G}})^{-1} = \omega(\mathbf{P}^{i})^{-1} + (1-\omega)\,(\mathbf{P}^{j})^{-1} \tag{A.21}$$

$$\hat{\mathbf{x}}^{\mathrm{G}} = \mathbf{P}^{\mathrm{G}}\left[\omega(\mathbf{P}^{i})^{-1}\hat{\mathbf{x}}^{i} + (1-\omega)(\mathbf{P}^{j})^{-1}\hat{\mathbf{x}}^{j}\right] \tag{A.22}$$

Table A.1: Sensor configuration for the simulation of synchronous track-to-track fusion algorithms.

	Sensor 1	Sensor 2
Measurement Period (ms)	80	80
σ_x (m)	2.00	0.75
σ_y (m)	0.50	1.20

where $\omega \in [0, 1]$ is a free parameter that weighs the state estimates from sensors i and j. The parameter ω can be chosen to adhere to a specific performance criteria. Typically, ω is chosen such that the determinant of the fused covariance $\mathbf{P}^{\mathrm{G}}$ is minimized:

$$\omega = \arg\min \left(\det \left(\mathbf{P}^{\mathrm{G}}\right)\right). \tag{A.23}$$

The covariance intersection algorithm has been used by Floudas to fuse sensor data from vision and radar sensors for automotive safety applications [100] and the fusion of radar and laser scanner object detection for collision mitigation [98].

A.4 Comparison

In this section, the synchronous track-to-track fusion algorithms described in previous section are evaluated using a sensor-to-sensor fusion strategy. The simulation in the previous section is carried out with two sensors observing an object traveling in the x-direction at 20 m/s. The two sensors both have a measurement period of 80 ms and are synchronized to one another, meaning that they both take measurements at exactly the same time and have no latency to the fusion module. Table A.1 gives an overview of the two sensors used in the simulation, with the sensor measurement errors in the x- and y-direction given as a standard deviation σ_x and σ_y. As in a realistic automotive application, the sensors' performance to one another is complementary: the weakness of one sensor is compensated by the strength of the other. In this case, the uncertainties in the x- and y-directions complement each other.

The consistency of the synchronous track-to-track fusion algorithms is evaluated first. The NEES over 100 Monte Carlo runs for the 10-second simulation is shown in Figure A.1. The 95% confidence interval is depicted by the dark gray horizontal lines at $\bar{\epsilon} = 6.70$ and $\bar{\epsilon} = 5.34$. As expected, the use of cross-covariance algorithm using the recursion method for calculating the cross-covariance matrix is consistent, as its NEES lies within the condifence interval. It is also the only optimal track-to-track fusion algorithm that fully compensates for the common process noise between the two sensor state estiamtes. The next best algorithm in terms of consistency is the use of cross-covariance algorithm using the estimation method for calculating the cross-covariance matrix. However, the algorithm slightly overestimates the true error, which means that the error covariance, $\mathbf{P}^{\mathrm{G}}$, is smaller than the true error. This can be seen in Figure A.1 in that the algorithm's NEES hovers around or is slightly above the upper bound of the confidence interval. The simple weighted fusion algorithm, which ignores the correlation between sensor estimates, largely overestimates the true error. On the opposite end of the spectrum is the covariance

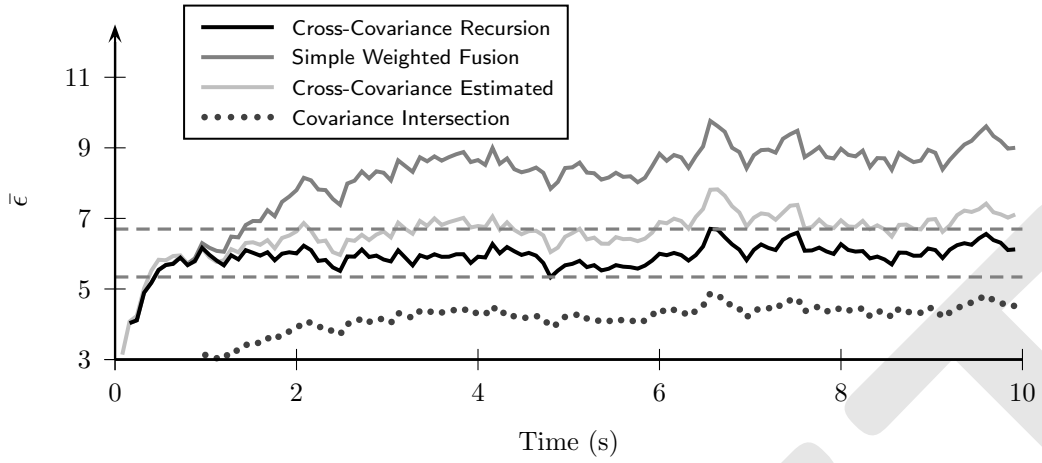


Figure A.1: The normalized estimation error squared (NEES) for synchronous track-to-track fusion algorithms during a simulation with 100 Monte Carlo runs.

intersection algorithm, which attempts to compensate for the correlation without explicitley calculating the cross-covariance matrix. This algorithm underestimates the true error of the fused state estimate, in that the error covariance, $\mathbf{P}^{\mathrm{G}}$, is larger than the true error. Underestimation of the error covariance is less critical than the overestimation due to the fact it is conservative in producing the fused accuracy of the state estimate.

The root mean square error for the position and velocity is shown in Figure A.2. The algorithm with the best performance in position is use of cross-covariance with the recursion method of calculating the cross-covariance matrix, with, after filter steady-state, yields an average RMSE over the simulation time of 0.4262 m. However, the simple weighted fusion and covariance intersection algorithms only have a very slightly larger RMSE in the position, with 0.4285 m and 0.4282 m, respectively, a difference which is for the most part negligible. The use of cross-covariance algorithm with the estimation method of calculating the cross-covariance matrix is the only algorithm that has a significantly larger RMSE in the position than the others, with a time-average RMSE over the simulation of 0.4481 m. Even this difference in error, given practical considerations, is quite good performance.

The relational performance of the velocity estimation between the different algorithms is similar to that of the position. The use of cross-covariance algorithm with the recursion method of calculating the cross-covariance matrix has a time-averaged RMSE of 0.9744 m/s, after filter steady-state, making it the algorithm with the best performance, as can be seen in Figure A.2b. The other three algorithms have a slightly larger error in the velocity estimate, with errors at around 0.988 m/s. Despite the difference in errors between the algorithms, as with the position, the difference can be considered to be negligible in practical applications.

Overall, the performance of the position and velocity state estimation for all of the synchronous track-to-track fusion algorithms is quite good. The significant difference in the algorithms is revealed in the consistency. Only the use of cross-covariance algorithm using the recursion method for calculating the cross-covariance matrix yields a consistent state estimate. The simple weighted fusion algorithm yields the most inconsistent estimate, overestimating the state by quite a bit. On the other hand, the covariance intersection algorithm underestimates the estimate.

Given all of the factors evaluated, for practical applications, the use of cross-covariance

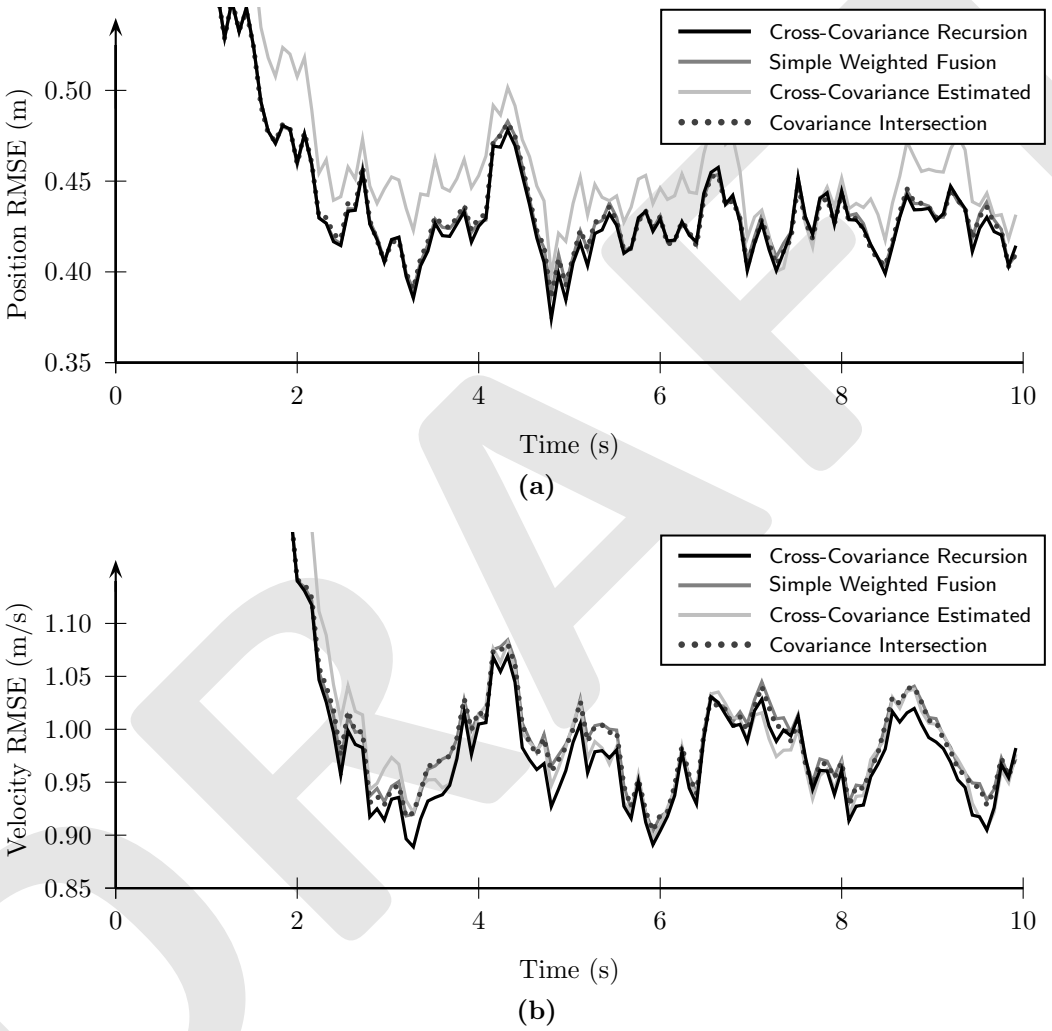


Figure A.2: The root mean square error (RMSE) of the position (a) and velocity (b) for synchronous track-to-track fusion algorithms during a simulation with 100 Monte Carlo runs.

algorithm using the estimation method for calculating the cross-covariance matrix is the best algorithm. Its independence from internal filter parameters make it feasible for use in practice. Additionally, it is the second-most consistent filter, only slightly overestimate the state.

B Determining the Trust Probability

For the fusion of the existence and classification probability, a trust probability, p_{trust} for existence and $p^i_{\text{trust}}(C_i)$ for classification, was introduced in Section 5.2 and Section 6.2. The idea of this trust probability is based on the hypothesis that sensor performance with respect to the existence and classification probabilities is not equal between different sensors. Therefore, the trust probability serves as a weighting factor during the fusion process, with the goal of properly considering sensor's performance during fusion.

In this appendix, the trust probability values are derived using the process described in Section 5.2.4 and Section 6.2.2 with the training data described in Section C.1.

B.1 Existence Trust Probability

In order to derived the trust probabilities for existence, Receiver Operating Characteristic (ROC) curves are generated for each sensor on the test vehicle from the training data of Section 6.2.2 using the evaluation process described in Section 7.3.1. The resulting ROC curves showing the sensor's detection performance is shown in Figure B.1.

A false positive rate per iteration $f = 0.5$ is used as the normalizing performance criteria for deriving the trust probabilities (depicted as the dashed vertical line in Figure B.1). The trust probabilities for each sensor, given in Table B.1, are determined by taking the true positive rate value of the ROC curves where they intersect with the false positive rate line of $f = 0.5$. These values were used as the sensor-level existence trust probability values for all of the evaluation results presented in Chapter 7. Note that the the ultrasonic sensors were not evaluated in Figure B.1 and are therefore given a value of $p^{\text{ultrasonic}}_{\text{trust}} = 0.50$.

Table B.1: Derived weighting probabilities for each sensor's individual detection performance with the existence probability.

Sensor	p^i_{trust}
Laser Scanner front	0.90
Laser Scanner rear	0.80
Laser Scanner side	0.70
Radar Front	0.65
Radar Rear	0.85
Camera	0.97
Ultrasonic	0.50

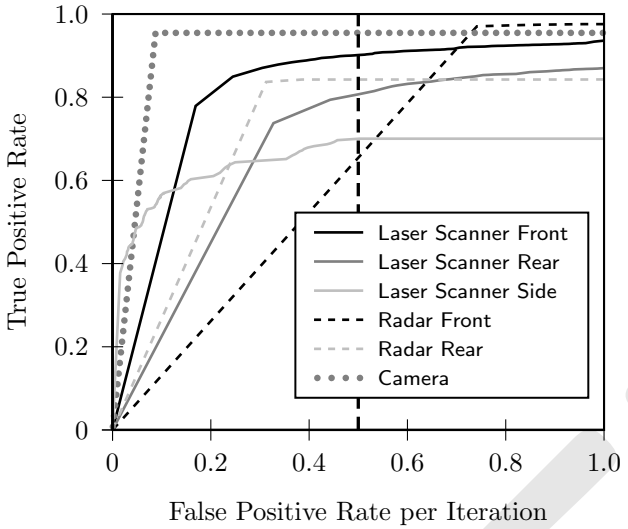


Figure B.1: ROC curves of sensor-level detection performance for the sensors on the test vehicle used for evaluation. The vertical line depicts the $f = 0.5$ false positive rate per iteration value from which the trust probabilities are derived.

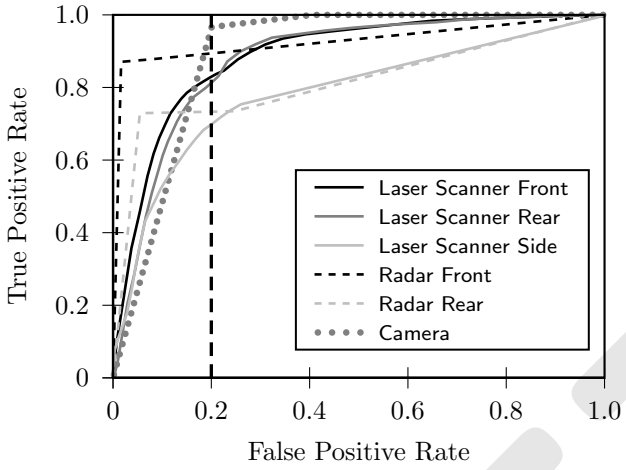
B.2 Classification Trust Probability

The trust probability for the classification performance of a sensor is derived in a similar manner as for the existence trust probability. However, since the classification vector $\mathbf{c}$ consists of several classification hypothesis, a separate ROC curve is generated for each class type for every sensor. Since this thesis focuses mainly on highway scenarios, a trust probability is derived only for the classes C_{Car}, C_{Truck} and $C_{\text{Motorcycle}}$. However, the same process applies for the other classes of the classification hypothesis set $\mathcal{C}$. Figure B.2 shows the ROC curves for the mentioned three classes.

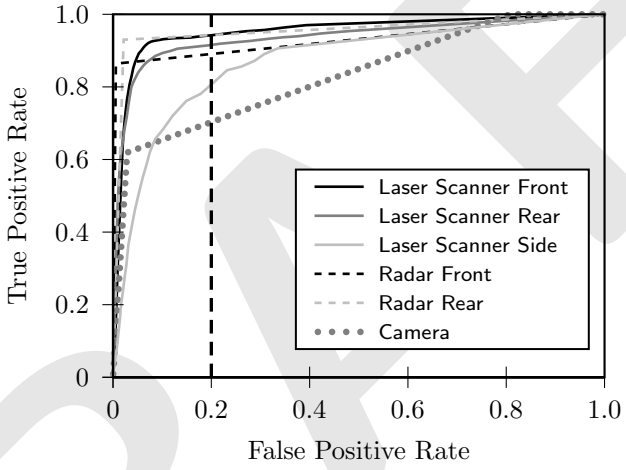
For deriving the classification trust probabilities, a common false positive rate of $f = 0.2$ is used, where the trust probability for each sensor and each class C_i takes on the value of the ROC curve where it intersects the line for $f = 0.2$. The results of which are shown in Table B.2. Again, for the ultrasonic sensor, no evaluation was performed and therefore a low trust probability of 0.10 for all classes was chosen. These trust probability values were used during all of the results from Chapter 7.

Table B.2: Derived weighting probabilities for each sensor's individual classification performance.

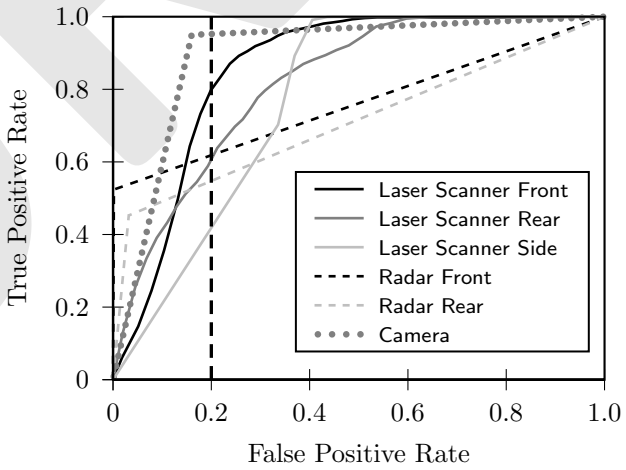
Sensor	$p^i_{\text{trust}}(C_{\text{Car}})$	$p^i_{\text{trust}}(C_{\text{Truck}})$	$p^i_{\text{trust}}(C_{\text{Motorcycle}})$
Laser Scanner front	0.83	0.94	0.80
Laser Scanner rear	0.81	0.92	0.60
Laser Scanner side	0.70	0.80	0.40
Radar Front	0.89	0.88	0.62
Radar Rear	0.73	0.94	0.54
Camera	0.96	0.70	0.98
Ultrasonic	0.10	0.10	0.10



(a) C_{Car}



(b) C_{Truck}



(c) $C_{\text{Motorcycle}}$

Figure B.2: ROC curves of the classification performance for the sensors on the test vehicle used for evaluation. Each class is shown in a separate ROC curve, where (a) shows C_{Car}, (b) C_{Truck} and (c) $C_{\text{Motorcycle}}$. The false positive rate of $f = 0.2$ for deriving the trust probabilities is depicted by the vertical dashed line.

C Evaluation Scenario Descriptions

Sensor data during several highway scenarios were recorded using the prototype test vehicle described in Section 7.1. The data from these scenarios were used to train and evaluate the algorithms presented in this thesis. This appendix gives a short description of the recorded scenarios.

C.1 Training Data

Data from scenarios on a closed test track (denoted with "TT") and real highway traffic (denoted with "RT") were used for training and parametrizing some of the algorithms presented in this thesis. Mainly the training of sensor-level classification algorithms and the parametrization of existence and classification fusion algorithms made use of the data described in this section.

A total of 16 scenarios were used as training data sets, summarized in Table C.1. Each of scenarios was semi-automatically labeled such that vehicles in the scenario, their type and their approximate location is saved for each processing cycle. Fifteen of the training scenarios were conducted on a closed test track, focusing on the collection motorcycle object data. Another scenario, recorded in real highway traffic was used to collect data on typical highway traffic objects. In total, 15,920 cycles of data was collected, which corresponds to over 25 minutes of recorded data. A total of 55,990 car, 9,977 truck and 10,552 motorcycle samples make up the complete training data set. Note that these numbers do not correspond to any defined field-of-view.

Figure C.1 shows some camera images from the training data, where the top row of the figure shows example scenes from the real highway traffic scenario (T-RT-S1) and the bottom rows shows some examples of the motorcycle scenarios on a closed test track.

C.2 Evaluation Data

A different set of data was used in Chapter 7 to evaluate the algorithms presented in this thesis. Two types of data was collected: test track scenarios with differential Global Positioning System (GPS) ground truth and real highway traffic scenarios with labeled vehicle data.

C.2.1 Test Track

On a closed test track, several overtaking scenarios were performed with the host vehicle and a target vehicle. Both vehicles are equipped with differential GPS in order to obtain ground truth data evaluating the surround object detection algorithms presented in this thesis. The ground truth consists of highly accurate position and velocity information of

Table C.1: Overview of the data sets used for algorithm training and parametrization.

Scenario	# Cycles	Car Samples	Truck Samples	Motorcycle Samples
T-RT-S1	5,368	54,701	9,977	0
T-TT-S1	1,265	340	0	1,265
T-TT-S2	580	0	0	580
T-TT-S3	1,101	380	0	1,101
T-TT-S4	562	0	0	562
T-TT-S5	1,109	320	0	1,109
T-TT-S6	876	0	0	876
T-TT-S7	484	0	0	484
T-TT-S8	326	0	0	326
T-TT-S9	222	0	0	222
T-TT-S10	203	0	0	203
T-TT-S11	529	0	0	529
T-TT-S12	973	0	0	973
T-TT-S13	807	249	0	807
T-TT-S14	567	0	0	567
T-TT-S15	948	0	0	948
Total	15,920	55,990	9,977	10,552

the two vehicles. A total of 12 overtaking maneuvers were performed and recorded, as summarized in Table C.2. Each of the scenarios varies the overtaking maneuver's constellation slightly, where the host and target vehicles perform the maneuver with different velocities in addition to the target and host vehicle taking turns being the overtaking vehicle.

In total, 4,891 cycles are recorded across the data sets and are available for evaluation. Figure C.2 shows an excerpt of such an overtaking scenario as seen from a camera sensor and the 3D visualization software, where the target vehicle has just passed the host vehicle-

C.2.2 Real Traffic

For the detection and classification performance, labeled scenarios from real highway traffic were used for evaluation. In these scenarios, the location and type of traffic objects (vehicles, trucks and motorcycles) were semi-automatically labeled by a person using a labeling software in order to generate a ground truth for the true objects and their type in the host vehicle's environment. The labeling process is not able to provide the position accuracy as with a differential GPS system and therefore these data sets are only used in order to evaluate detection and classification performance and not state estimation performance.

A total of three highway scenarios were recorded for evaluation, summarized in Table C.3. The scenarios were recorded on highways just outside of Munich, Germany with moderate to heavy traffic conditions. One of the scenarios, E-RT-S3, was specifically recorded with a motorcycle constantly in the surrounding environment in order to increase the samples available for motorcycle object evaluation. In total, 12,881 cycles of data were recorded across the three data sets, which contained 88,385 car samples, 24,496 trucks and 3,047 motorcycles. In total, this corresponds to over 20 minutes of data from driving on highways

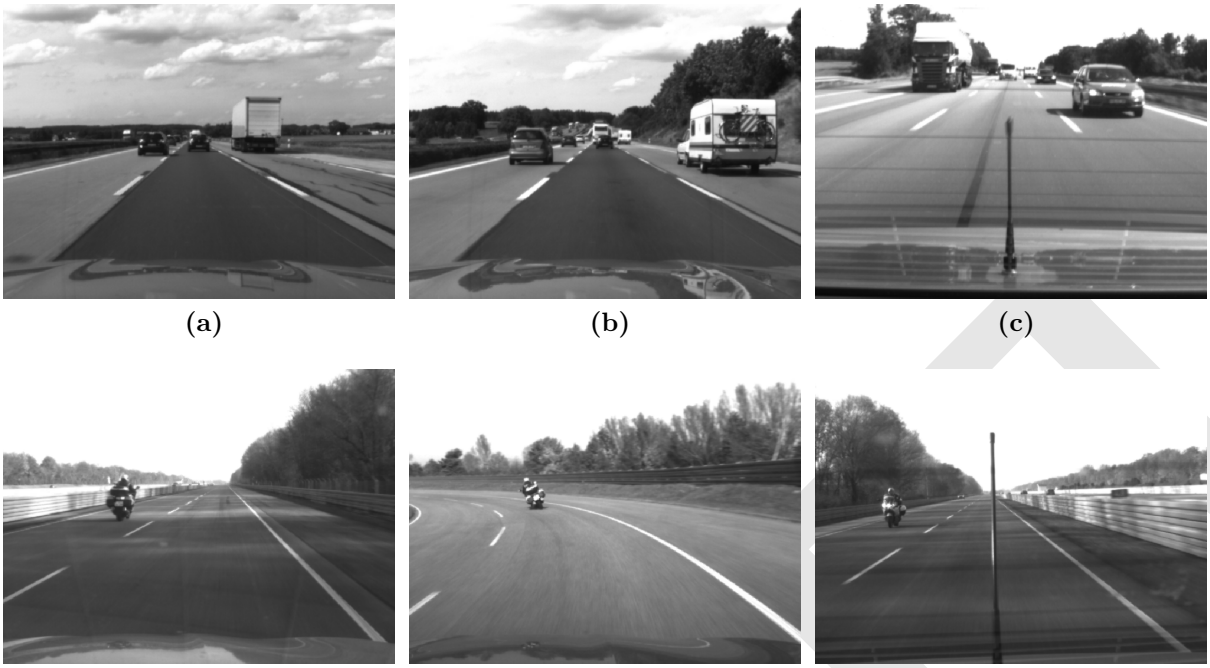


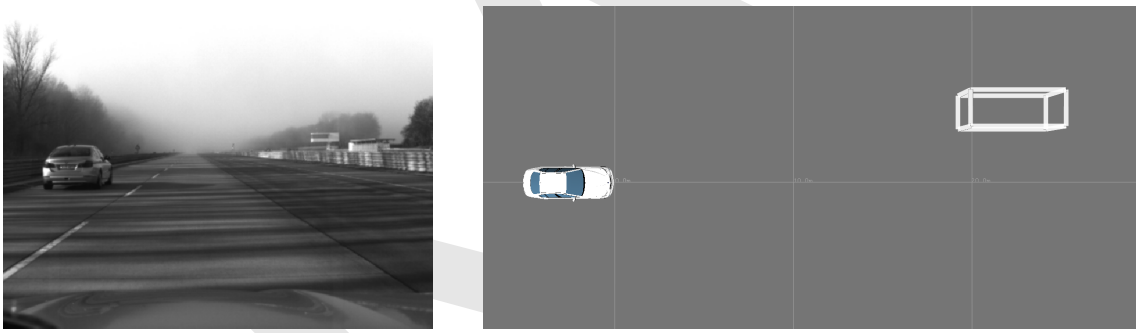
Figure C.1: Examples camera images from the training scenarios. The images (a)-(c) show scenes from the real highway traffic scenario and the images (d)-(f) shows some scenes from the motorcycle data sets.

with real traffic. As with the training data, note that these numbers do not correspond to any defined field-of-view.

In order to get an impression of some of the scenes recorded, Figure C.3 shows images as recorded from a camera sensor from the host-vehicle. Each row of the image refers to one of three data sets, where the images for the top row (a)-(c) are for data set E-RT-S1 and so on. Note that the traffic scenes are a good mix of moderate traffic with normal sized vehicles, but also traffic scenes with more heavy traffic, especially dense traffic with trucks on the right lane. Note that the third scenario, E-RT-S3, for the majority of the time contains a motorcycle in the vicinity of the host vehicle.

Table C.2: Summary of the overtaking scenarios with precise ground truth data for evaluation.

Scenario	# Cycles	Overtaking Vehicle	v_{host}	v_{target}
E-TT-S1	365	target	80	100
E-TT-S2	339	host	90	80
E-TT-S3	397	target	80	90
E-TT-S4	317	host	100	80
E-TT-S5	513	host	90	80
E-TT-S6	581	target	90	80
E-TT-S7	516	host	90	80
E-TT-S8	536	target	90	80
E-TT-S9	428	host	90	80
E-TT-S10	407	target	90	80
E-TT-S11	226	host	110	80
E-TT-S12	266	target	110	80
Total	4,891			



(a) **(b)**

Figure C.2: A scene from an overtaking maneuver scenario used for evaluation, as seen from a camera sensor (a) and the 3D visualization software (b), where the ground truth for the target vehicle is represented as a white wire-frame box.

Table C.3: Overview of the data sets used for real-traffic evaluation of detection and classification performance.

Scenario	# Cycles	Car Samples	Truck Samples	Motorcycle Samples
E-RT-S1	4,960	30,499	9,817	0
E-RT-S2	5,365	45,381	7,130	712
E-RT-S3	2,556	12,505	7,549	2,335
Total	12,881	88,385	24,496	3,047

Figure C.3: Example camera images for the highway scenarios used for evaluation. The top row (a)-(c) is from scenario E-RT-S1, (d)-(f) from E-RT-S2 and (g)-(i) from E-RT-S3.

References

[1] C. Adam, R. Schubert, and G. Wanielik, "Radar-based extended object tracking under clutter using generalized probabilistic data association," in *IEEE Intelligent Transportation Systems Conference*, The Hague, Netherlands, October 2013, pp. 1408–1415.

[2] D. Ahrens, "Parkassistent mit Längs- und Querfürung," in *5. Tagung Fahrerassistenz*, Munich, Germany, May 2012.

[3] R. Altendorfer and S. Matzka, "A confidence measure for vehicle tracking based on a generalization of bayes estimation," in *IEEE Intelligent Vehicles Symposium*, San Diego (CA), USA, June 2010, pp. 766–772.

[4] S. Aly, L. Hassan, A. Sagheer, and H. Murase, "Partially occluded pedestrian classification using part-based classifiers and restricted boltzmann machine model," in *IEEE International Conference on Intelligent Transportation Systems*, The Hague, Netherlands, October 2013, pp. 1065–1070.

[5] M. Ardelt, P. Waldmann, F. Homm, and N. Kaempchen, "Strategic decision-making process in advanced driver assistance systems," in *6th IFAC Symposium Advances in Automotive Control*, Munich, Germany, July 2010.

[6] M. Ardelt, "Hybrid control strategies for advanced safety and driver assistance systems," Ph.D. dissertation, Technische Universität München, May 2012.

[7] M. Ardelt, C. Coester, and N. Kaempchen, "Highly automated driving on freeways in real traffic using a probabilistic framework," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1576–1585, December 2012.

[8] M. Ardelt and P. Waldmann, "Hybrid control concept for highly automated driving on freeways," *Automatisierungstechnik*, vol. 59, no. 12, pp. 738–750, December 2011.

[9] Audi USA. (2015, January) Mission accomplished: Audi A7 piloted driving car completes 550-mile automated test drive. [Online]. Available: https://www.audiusa.com/newsroom/news/

[10] Audi USA News. (2010, November) Autonomous Audi TTS Pikes Peak achieves this years goal. [Online]. Available: http://www.audiusanews.com/

[11] J. Aue, M. Schmid, T. Graf, and J. Effertz, "Improved object tracking from detailed shape estimation using object local grid maps with stereo," in *IEEE Intelligent Transportation Systems Conference*, The Hague, Netherlands, October 2013, pp. 330–335.

[12] J. Aue, M. Schmid, T. Graf, J. Effertz, and P. Muehlfellner, "Object tracking from medium level stereo camera data providing detailed shape estimation using local grid maps," in *IEEE Intelligent Vehicles Symposium*, Gold Coast, Australia, June 2013, pp. 1330–1335.

[13] AutoNOMOUS Labs. [Online]. Available: http://autonomos.inf.fu-berlin.de/

[14] Q. Baig, O. Aycard, T. Dung Vu, and T. Fraichand, "Fusion between laser and stereo vision data for moving objects tracking in intersection like scenario," in *IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, June 2011, pp. 362–367.

[15] Y. Bar-Shalom, "On the track-to-track correlation problem," *IEEE Transactions on Automatic Control*, vol. 26, no. 2, pp. 571–572, April 1981.

[16] ——, *Multitarget-Multisensor Tracking: Principles and Techniques*, 3rd ed. Storrs, CT: Yaakkov Bar-Shalom, 1995.

[17] Y. Bar-Shalom and L. Campo, "The effect of the common process noise on the two-sensor fused-track covariance," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 22, no. 6, pp. 803–805, November 1986.

[18] Y. Bar-Shalom, P. Willett, and X. Tian, *Tracking and Data Fusion: A Handbook of Algorithms*. Storrs, CT: Yaakkov Bar-Shalom, 2011.

[19] Y. Bar-Shalom, "Update with out-of-sequence measurements in tracking: Exact solution," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 3, pp. 769–778, July 2002.

[20] Y. Bar-Shalom and H. Chen, "Multisensor track-to-track association for tracks with dependent errors," *Journal of Advances in Information Fusion*, vol. 1, no. 1, pp. 3–14, July 2006.

[21] ——, "Track-to-track association using attributes," *Journal of Advances in Information Fusion*, vol. 2, no. 1, pp. 49–59, 2007.

[22] Y. Bar-Shalom, H. Chen, and M. Mallick, "One-step solution for the multistep out-of-sequence measurement problem in tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 1, pp. 27–37, January 2004.

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

[24] Y. Bar-Shalom and P. J. Lanzkron, "A two-step method for out-of-sequence measurements," in *IEEE Aerospace Conference*, March 2004, pp. 2036–2041.

[25] Y. Bar-Shalom and X.-R. Li, *Estimation and Tracking: Principles, Techniques and Software*. Norwood, MA: Artech House, 1993.

[26] Y. Bar-Shalom and E. Tse, "Tracking in a cluttered environment with probabilistic data association," *Automatica*, vol. 11, no. 5, pp. 451–460, September 1975.

[27] A. Bartels, S. Steinmeyer, S. Brosig, and C. Spichalsky, "Lane Keeping Support," in *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*, 2nd ed., H. Winner, S. Hakuli, and G. Wolf, Eds. Wiesbaden, Germany: Vieweg+Teubner, 2009, pp. 562–571.

[28] A. Bartels, T.-B. To, S. Karrenberg, and A. Weiser, "Highly automated driving on motorways," *Automobiltechnische Zeitschrift*, pp. 652–657, September 2011.

[29] J. Becker, "Fusion of data from the object-detecting sensors of an autonomous vehicle," in *IEEE International Conference on Intelligent Transportation Systems*, Tokyo, Japan, October 1999, pp. 362–367.

[30] J. Becker and A. Simon, "Sensor and navigation data fusion for an autonomous vehicle," in *IEEE Intelligent Vehicles Symposium*, Dearborn (MI), USA, October 2000, pp. 1224–1228.

[31] J. Becker and M. Stämpfle, "Fusion and filtering of multidimensional objects for driver assistance systems," in *IEEE Intelligent Vehicles Symposium*, Parma, Italy, June 2004, pp. 613–618.

[32] M. Benmimoun, C. Kessler, A. Zlocki, and A. Etemad, "euroFOT: Field operational test and impact assessment of advanced driver assistance systems - first results," in *20. Aachner Kolloquium Fahrzeug- und Motorentechnik*, Aachen, Germany, October 2011.

[33] M. Benmimoun, A. Pütz, A. Zlocki, and L. Eckstein, "Ergebnisse der Wirkungsanalyse von Adaptive Cruise Control (ACC) und Forward Collision Warning (FCW) im Rahmen eines Feldversuchs," in *28. VDI/VW-Gemeinschaftstagung*, Wolfsburg, Germany, October 2012, pp. 153–167.

[34] C. Bergenhem, Q. Huang, A. Benmimoun, and T. Robinson, "Challenges of platooning on public motorways," in *17th World Congress on Intelligent Transportation Systems*, Busan, Korea, October 2010.

[35] M. Bertozzi, L. Bombini, A. Broggi, M. Buzzoni, E. Cardarelli, S. Cattani, P. Cerri, A. Coati, S. Debattisti, A. Falzoni, R. I. Fedriga, M. Felisa, L. Gatti, A. Giacomazzo, P. Grisleri, M. C. Laghi, L. Mazzei, P. Medici, M. Panciroli, P. P. Porta, P. Zani, and P. Versari, "VIAC: an out of ordinary experiment," in *IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, June 2011, pp. 175–180.

[36] M. Bertozzi, A. Broggi, G. Conte, and A. Fascioli, "The experience of the argo autonomous vehicle," in *SPIE Aerosense Conference*, Orlando (FL), USA, April 1998, pp. 218–229.

[37] D. Bertsekas and D. Castanon, "The auction algorithm for the transportation problem," *Annals of Operations Research*, vol. 20, pp. 67–96, 1989.

[38] C. Bishop, *Neural Networks for Pattern Recoginition*. New York: Oxford University Press, 1996.

[39] ——, *Pattern Recognition and Machine Learning*. New York, NY: Springer, 2006.

[40] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Norwood, MA: Artech House, 1999.

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

[42] C. Blanc, L. Trassoudaine, Y. Le Guilloux, and R. Moreira, "Track to track fusion method applied to road obstacle detection," in *IEEE International Conference on Information Fusion*, Stockholm, Sweden, June 2004.

[43] G. O. Blog. (2014) Just press go: designing a self-driving vehicle. [Online]. Available: http://googleblog.blogspot.com/2014/05/just-press-go-designing-self-driving.html

[44] H. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with markovian switching coefficients," *IEEE Transactions on Automatic Control*, vol. 33, no. 8, pp. 780–783, August 1988.

[45] H. Blom and E. Bloem, "Combining IMM and JPDA for tracking multiple maneuvering target in clutter," in *5th International Conference on Information Fusion*, Annapolis (MD), USA, July 2002, pp. 705–712.

[46] BMW Group PressClub Global. [Online]. Available: http://www.press.bmwgroup.com/

[47] ——. (2009, July) Stopping safely in an emergency. [Online]. Available: http://www.press.bmwgroup.com/

[48] ——. (2010, October) Innovation days connected drive meets efficient dynamics. [Online]. Available: http://www.press.bmwgroup.com/

[49] ——. (2011, August) Ready for takeover! [Online]. Available: http://www.press.bmwgroup.com/

[50] ——. (2013, June) Bmw connecteddrive 2013. [Online]. Available: http://www.press.bmwgroup.com/

[51] ——. (2014, January) Bmw at the consumer electronics show (ces) in las vegas 2014. [Online]. Available: http://www.press.bmwgroup.com/

[52] ——. (2015, December) BMW Innovations at the 2015 Consumer Electronics Show (CES) in Las Vegas. [Online]. Available: http://www.press.bmwgroup.com/

[53] G. A. Borges and M. J. Aldon, "Line extraction in 2d range images for mobile robotics," *Journal of Intelligent and Robotic Systems*, vol. 40, no. 3, pp. 267–297, ? 2004.

[54] Bosch Group. (2015, September) Cars that drive themselves - highway pilot technically viable in five years. [Online]. Available: http://www.bosch-presse.de/

[55] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*. ACM Press, 1992, pp. 144–152.

[56] A. Broggi, M. Buzzoni, S. Debattisti, P. Grisleri, M. Laghi, P. Medici, and P. Versari, "Extensive tests of autonomous driving technologies," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1403–1415, September 2013.

[57] A. Broggi, M. Bertozzi, A. Fascioli, C. G. L. Bianco, and A. Piazzi, "The ARGO autonomous vehicle's vision and control systems," *International Journal of Intelligent Control and Systems*, vol. 3, no. 4, pp. 409–441, November 1999.

[58] A. Broggi, A. Cappalunga, C. Caraffi, S. Cattani, S. Ghidoni, P. Grisleri, P. P. Porta, M. Posterli, and P. Zani, "TerraMax vision at the urban challenge 2007," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 4, pp. 194–205, March 2010.

[59] C. Bruges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[60] Bureau of Transportation Statistics, "National transportation statistics 2013," United States Department of Transportation, Tech. Rep., 2011.

[61] California Senate Bill 1298, September 2012.

[62] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[63] K. C. Chang, R. K. Saha, and Y. Bar-Shalom, "On optimal track-to-track fusion," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, no. 4, pp. 1271–1276, October 1997.

[64] K.-C. Chang and Y. Bar-Shalom, "Joint probabilistic data association for multitarget tracking with possibly unresolved measurements," in *American Control Conference*, San Francisco (CA), USA, June 1983, pp. 466–471.

[65] ——, "Joint probabilistic data association for multitarget tracking with possibly unresolved measurements and maneuvers," *IEEE Transactions on Automatic Control*, vol. 29, no. 7, pp. 585–594, 1984.

[66] H. Cheng, N. Zheng, X. Zhang, J. Qin, and H. van de Wetering, "Interactive road situation analysis for driver assistance and safety warning systems: Framework and algorithms," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 1, pp. 157–167, March 2007.

[67] W.-S. Choi, Y.-S. Kim, S.-Y. Oh, and J. Lee, "Fast iterative closest point framework for 3d LIDAR data in intelligent vehicle," in *IEEE Intelligent Vehicles Symposium*, Alcalá de Henares, Spain, June 2012, pp. 1029–1034.

[68] C. Chong, S. Mori, W. H. Barker, and K. Chang, "Architectures and algorithms for track association and fusion," *IEEE Trans. on Aerosp. and Electron. Syst.*, vol. 15, no. 1, pp. 5–13, January 2000.

[69] Continental AG Press Portal. (2012, March) Continental tests highly-automated driving. [Online]. Available: http://www.conti-online.com/

[70] H. Cramer, U. Scheunert, and C. Wanielik, "Multi sensor fusion for object detection using generalized feature models," in *6th International Conference of Information Fusion*, Cairns, Australia, July 2003, pp. 2–10.

[71] ——, "Multi sensor data fusion using a generalized feature model applied to different types of extended road objects," in *7th International Conference on Information Fusion*, Stockholm, Sweden, July 2004, pp. 664–671.

[72] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego (CA), USA, June 2005, pp. 886–893.

[73] D. Damböck, M. Farid, L. Tönert, and K. Bengler, "Übernahmezeiten beim hochautomatisierten Fahren," in *5. Tagung Fahrerassistenz*, Munich, Germany, May 2012.

[74] R. Danescu, F. Oniga, and S. Nedevschi, "Modeling and tracking the driving environment with a particle-based occupancy grid," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1331–1342, December 2011.

[75] M. Darms and H. Winner, "A modular system architecture for sensor data processing of ADAS applications," in *IEEE Intelligent Vehicles Symposium*, Las Vegas (NV), USA, June 2005, pp. 729–734.

[76] M. Darms, "Eine Basis-Systemarchitektur zur Sensordatenfusion von Umfeldsensoren für Fahrerassistenzsysteme," Ph.D. dissertation, Technische Universität Darmstadt, July 2007.

[77] M. Darms, P. Rybski, and C. Urmson, "Classification and tracking of dynamic objects with multiple sensors for autonomous driving in urban environments," in *IEEE Intelligent Vehicles Symposium*, Eindhoven, Netherland, June 2008, pp. 1197–1202.

[78] M. Darms, P. Rybski, C. Baker, and C. Umerson, "Obstacle detection and tracking for the urban challenge," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 6, pp. 475–485, September 2009.

[79] Defense Advanced Research Projects Agency (DARPA). Grand Challenge 04. [Online]. Available: http://archive.darpa.mil/grandchallenge04/

[80] ——. Grand Challenge 05. [Online]. Available: http://archive.darpa.mil/grandchallenge05/

[81] ——. Urban Challenge. [Online]. Available: http://archive.darpa.mil/grandchallenge/

[82] Delphi Automotive PLC. (2015, April) Delphi successfully completes first coast-to-coast automated drive. [Online]. Available: http://investor.delphi.com/investors/press-releases/

[83] Deutsches Institute für Normung, "DIN 70000; Road vehicles; vehicle dynamics and road-holding ability; vocabulary (ISO 8855:1991, modified)," 1994.

[84] J. Dickmann, F. Diewald, M. Mählisch, J. Klappstein, S. Zuther, S. Pietzsch, S. Hahn, and M. Munz, "Environmental perception for future integrated safety systems," in *Enhanced Safety Vehicles Conference*, Stuttgart, Germany, June 2009, pp. 1–12.

[85] E. Dickmanns, R. Behringer, D. Dickmanns, T. Hildebrandt, M. Maurer, F. Thomanek, and J. Schiehlen, "The seeing passenger car VaMoRs-P," in *IEEE Intelligent Vehicles Symposium*, Paris, France, October 1994, pp. 68–73.

[86] M. Dittmer, J. Kibbel, H. Salow, and V. Willhoeft, "Team-LUX: DARPA urban challenge 2007," Defense Advanced Research Projects Agency, Tech. Rep., June 2007.

[87] O. Drummond, "On track and tracklet fusion filtering," *Proceedings of SPIE*, vol. 4728, pp. 176–195, August 2002.

[88] R. Duda, P. Hart, and D. Stork, *Patter Classification*. New York, NY: Wiley, 2001.

[89] J. Effertz, "Sensor architectures and data fusion for robotic perception in urban environments at the 2007 DARPA urban challenge," *Lecture Notes in Computer Science*, vol. 4931, pp. 275–290, 2008.

[90] M. Enzweiler, M. Hummel, D. Pfeiffer, and U. Franke, "Efficient stixel-based object recognition," in *IEEE Intelligent Vehicles Symposium*, Alcalá de Henares, Spain, June 2012, pp. 1066–1071.

[91] F. Erbs, B. Schwarz, and U. Franke, "Stixmentation - probabilistic stixel based traffic scene labeling," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2012, pp. 71.1–71.12.

[92] ——, "From stixels to objects - a conditional random field based approach," in *IEEE Intelligent Vehicles Symposium*, Gold Coast, Australia, June 2013, pp. 586–591.

[93] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *2nd International Conference on Knowledge Discovery and Data Mining*, Portland (OR), USA, August 1996, pp. 226–231.

[94] European New Car Assessment Programme, "Euro NCAP rating review - report from the ratings group," European New Car Assessment Programme, Tech. Rep., June 2012.

[95] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[96] A. Fischer, "Inside google's quest to popularize self-driving cars," Popular Science, September 2013. [Online]. Available: http://www.popsci.com/cars/article/2013-09/google-self-driving-car

[97] Florida Committee Substitute House Bill 1207, April 2012.

[98] N. Floudas, P. Lytrivis, H. Avgoustidis, M. Ahrholdt, G. Thomaidis, and A. Amditis, "Track based multi sensor data fusion for collision mitigation," in *11th International Conference on Information Fusion*, Cologne, Germany, June 2008, pp. 1–8.

[99] N. Floudas, A. Polychronopoulos, O. Aycard, J. Burlet, and M. Ahrholdt, "High-level sensor data fusion approaches for object recognition in road environments," in *IEEE Intelligent Vehicles Symposium*, Istanbul, Turkey, June 2007, pp. 136–141.

[100] N. Floudas, A. Polychronopoulos, M. Tsogas, and A. Amditis, "Multi-sensor coordination and fusion for automotive safety applications," in *9th International Conference on Information Fusion*, Florence, Italy, July 2006, pp. 1–8.

[101] N. Floudas, P. Lytrivis, A. Polychronopoulos, and A. Amditis, "On the track-to-track association problem in road environments," in *10th International Conference on Information Fusion*, Quebec, Canada, July 2011, pp. 1–8.

[102] U. Franke, D. Gavrilla, S. Görzig, F. Lindner, F. Paetzold, and C. Wöhler, "Autonomous driving goes downtown," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 6, pp. 40–48, November/December 1998.

[103] U. Franke, D. Pfeiffer, C. Rabe, C. Knoeppel, M. Enzweiler, F. Stein, and R. G. Herrtwich, "Making bertha see," in *ICCV Workshop on Computer Vision for Autonomous Driving*, Sydney, Australia, December 2013.

[104] D. Fränken and A. Hüpper, "Improved fast covariance intersection for distributed data fusion," in *8th International Conference on Information Fusion*, Philadelphia (PA), USA, July 2005, pp. 154–160.

[105] Freie Universität Berlin. (2012, October) "made in germany" from Freie Universität Berlin first autonomous car to drive around mexico city. [Online]. Available: http://www.fu-berlin.de/en/presse/informationen/fup/2012/fup_12_286/index.html

[106] K. Fuerstenberg and M. Dittmer, "Data fusion and ego-estimation using laserscanners," in *IEEE Intelligent Transportation Systems*, Shanghai, China, October 2003, pp. 1224–1228.

[107] K. Fuerstenberg and K. Dietmayer, "Object tracking and classification for multiple active safety and comfort applications using a laser scanner," in *IEEE Intelligent Vehicles Symposium*, Parma, Italy, June 2004, pp. 802–807.

[108] K. Fuerstenberg, D. Linzmeier, and K. Dietmayer, "Pedestrian recognition and tracking of vehicles using a vehicle based multilayer laser scanner," in *10th World Congress on Intelligent Transportation Systems*, Madrid, Spain, November 2003.

[109] J. Funke, P. Theodosis, R. Hindiyeh, G. Stanek, K. Kritatakirana, C. Gerdes, D. Langer, M. Hernandez, and B. Müller-Bessler, "Up to the limits: Autonous audi tts," in *IEEE Intelligent Vehicles Symposium*, Alcalá de Henares, Spain, June 2012, pp. 541–547.

[110] P. Furgale, U. Schwesinger, M. Rufli, W. Derendarz, H. Grimmett, P. Mühlfellner, S. Wonneberger, J. Timpner, S. Rottmann, B. Li, B. Schmidt, T. Nguyen, E. Cardarelli, S. Cattani, S. Brüning, S. Horstmann, M. Stellmacher, H. Mielenz, K. Köser, M. Beermann, C. Häne, L. Heng, G. Lee, F. Fraundorfer, R. Iser, R. Triebel, I. Posner, P. Newman, L. Wolf, M. Pollefeys, S. Brosig, J. Effertz, C. Pradalier, and R. Siegwart, "Toward automated driving in cities using close-to-market sensors," in *IEEE Intelligent Vehicles Symposium*, Gold Coast, Australia, June 2013, pp. 809–816.

[111] T. Gasser, "Ergebnisse der Projektgruppe Automatisierung: Rechtsfolgen zunehmender Fahrzeugautomatisierung," in *5. Tagung Fahrerassistenz*, Munich, Germany, May 2012.

[112] ——, "Rechtsfolgen zunehmender fahrzeugautomatisierung," Bundesanstalt für Straßenwesen, Tech. Rep., 2012.

[113] J. Gayko, "Lane Keeping Support," in *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*, 2nd ed., H. Winner, S. Hakuli, and G. Wolf, Eds. Wiesbaden, Germany: Vieweg+Teubner, 2009, pp. 554–561.

[114] G. Geduld, "Lidarsensorik," in *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*, 2nd ed., H. Winner, S. Hakuli, and G. Wolf, Eds. Wiesbaden, Germany: Vieweg+Teubner, 2009, pp. 172–185.

[115] S. Gehrig, A. Barth, N. Schneider, and J. Siegemund, "A multi-cue approach for stereo-based object confidence estimation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, October 2012, pp. 3055–3060.

[116] A. Gepperth, M. Ortiz, and B. Heisele, "Partially occluded pedestrian classification using part-based classifiers and restricted boltzmann machine model," in *IEEE International Conference on Intelligent Transportation Systems*, The Hague, Netherlands, October 2013, pp. 348–353.

[117] C. Gläser, T. P. Michalke, L. Bürkle, and F. Niewels, "Environment perception for inner-city driver assistance and highly-automated driving," in *IEEE Intelligent Vehicles Symposium*, Dearborn (MI), USA, June 2014, pp. 1270–1275.

[118] C. Gold, D. Damböck, L. Lorenz, and K. Bengler, ""Take Over!" How long does it take to get the driver back into the loop?" in *57th Human Factors and Ergonomics Society (HFES) International Annual Meeting*, San Diego (CA), USA, September 2013.

[119] R. Grewe, A. Hohm, S. Lüke, and H. Winner, "Umfeldmodelle Standardisierte Schnittstellen für Assistenzsysteme," *Automobiltechnische Zeitschrift*, pp. 334–338, May 2012.

[120] M. Grinberg, F. Ohr, and J. Beyerer, "Feature-based probabilistic data association (FBPDA) for visual multi-target detection and tracking under occlusions and split and merge effects," in *IEEE International Conference on Intelligent Transportation Systems*, St. Louis (MI), USA, October 2009, pp. 1–8.

[121] M. Grinberg, F. Ohr, D. Willersinn, and J. Beyerer, "Feature-based probabilistic data association and tracking," in *7th International Workshop on Intelligent Transportation*, Hamburg, Germany, March 2010, pp. 29–34.

[122] E. Guizzo, "How google's self-driving car works," The New York Times, October 2011. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works

[123] HAVEit. [Online]. Available: http://www.haveit-eu.org/

[124] T. Herpel, C. Lauer, R. German, and J. Salzberger, "Multi-sensor data fusion in automotive applications," in *3rd International Conference on Sensing Technology*, Tainan, Taiwan, November 2008, pp. 206–211.

[125] M. Himmelsbach and H.-J. Wuensche, "Tracking and classification of arbitrary objects with bottom-up/top-down detection," in *IEEE Intelligent Vehicles Symposium*, Alcalá de Henares, Spain, June 2012, pp. 577–582.

[126] A. Houenou, P. Bonnifait, V. Cherfaoui, and J.-F. Boissou, "A track-to-track association method for automotive perception systems," in *IEEE Intelligent Vehicles Symposium*, Alcalá de Henares, Spain, June 2012, pp. 704–710.

[127] T. Huck, A. Westenberger, M. Fritzsche, T. Schwarz, and K. Dietmayer, "Precise timestamping and temporal synchronization in multi-sensor fusion," in *IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, June 2011, pp. 242–247.

[128] M. Hurley, "An information theoretic justification for covariance intersection and its generalization," in *5th International Conference on Information Fusion*, Annapolis (MD), USA, July 2002, pp. 8–11.

[129] International Organization for Standardization, "ISO 26262; Road Vehicles - Functional Safety," 2011.

[130] P. Jeevan, F. Harchut, B. Mueller-Bessler, and B. Huhnke, "Realizing autonomous valet parking with automotive grade sensors," in *IEEE International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, October 2010, pp. 3824–3829.

[131] T. Jochem, D. Pomerleau, B. Kumar, and J. Armstring, "PANS: A portable navigation platform," in *IEEE Intelligent Vehicles Symposium*, Detroit (MI), USA, September 1995, pp. 107–112.

[132] S. Julier and J. Uhlmann, "General decentralized data fusion with covariance intersection (CI)," in *Handbook of Data Fusion*, D. Hall and J. Llinas, Eds. Boca Raton, FL: CRC Press, 2001, pp. 12–1–12–25.

[133] ——, "Using covariance intersection for SLAM," *Robotics and Autonomous Systems*, vol. 55, no. 1, pp. 3–20, January 2007.

[134] ——, "A new extension of the kalman filter to nonlinear systems," *Proceedings of SPIE*, vol. 3, no. 1, pp. 182–193, 1997.

[135] S. Julier, J. Uhlmann, and H. Durrant-Whyte, "A new approach for filtering nonlinear systems," in *American Control Conference*, vol. 3, Seattle (WA), USA, June 1995, pp. 1628–1632.

[136] R. Jungnickel and F. Korf, "Object tracking and dynamic estimation in evidential grids," in *IEEE Intelligent Transportation Systems Conference*, Qingdao, China, October 2013, pp. 2310–2316.

[137] N. Kaempchen, K. Fuerstenberg, A. Skibicki, and K. Dietmayer, "Sensor fusion for multiple automotive active safety and comfort applications," in *International Forum on Advanced Microsystems for Automotive Applications*, Berlin, Germany, March 2004, pp. 137–163.

[138] N. Kaempchen, P. Waldmann, F. Homm, and M. Ardelt, "Umfelderfassung für den Nothalteassistent – ein System zum automatischen Anhalten bei plötzlich reduzierter Fahrfäigkeit des Fahrers," in *Automatisierungs-, Assistenzsysteme und eingebettete Systeme für Transportmittel*, Braunschweig, Germany, February 2010, pp. 121–137.

[139] N. Kaempchen, "Feature-level fusion of laser scanner and video data for advanced driver assistance systems," Ph.D. dissertation, University of Ulm, June 2007.

[140] N. Kaempchen, M. Bühler, and K. Dietmayer, "Feature-level fusion for free-form object tracking using laserscanner and video," in *IEEE Intelligent Vehicles Symposium*, Las Vegas (NV), USA, June 2005, pp. 453–458.

[141] N. Kaempchen, K. Weiss, M. Schaefer, and K. Dietmayer, "IMM object tracking for high dynamic driving maneuvers," in *IEEE Intelligent Vehicles Symposium*, Parma, Italy, June 2004, pp. 825–830.

[142] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[143] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schröder, M. Thuy, M. Goebel, F. von Hundelshausen, O. Pink, C. Frese, and C. Stiller, "Team AnnieWAY's autonomous system for the 2007 DARPA urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 615–639, September 2008.

[144] S. Karush, "They're working: Insurance claims data show which new technologies are preventing crashes," *Status Report of the Insurance Institute for Highway Safety — Highway Loss Data Institute*, vol. 52, no. 5, July 2012.

[145] R. Katzwinkel, R. Auer, S. Brosig, M. Rohlfs, V. Schöning, F. Schroven, F. Schwitters, and U. Wuttke, "Einparkassistenz," in *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*, 2nd ed.,

H. Winner, S. Hakuli, and G. Wolf, Eds. Wiesbaden, Germany: Vieweg+Teubner, 2009, pp. 471–477.

[146] D. Kellner, J. Klappstein, and K. Dietmayer, "Grid-based DBSCAN for clustering extended objects in radar data," in *IEEE Intelligent Vehicles Symposium*, Alcalá de Henares, Spain, June 2012, pp. 365–370.

[147] H. Kennedy, "Clutter-based test statistics for automatic track initation," *ACTA Automatica Sinica*, vol. 34, no. 3, pp. 266–273, March 2008.

[148] K. Klasing, D. Wollherr, and M. Buss, "A clustering method for efficient segmentation of 3d laser data," in *IEEE International Conference on Robotics and Automation*, Pasadena (CA), USA, May 2008, pp. 4043–4048.

[149] D. Koks and S. Challa, "An introduction to Bayesian and Dempster-Shafer data fusion," Defence Science and Technology Organization, Edinburgh, Australia, Tech. Rep. DSTO-TR-1436, 2005.

[150] K. Kritayakirana and J. Gerdes, "Autonomous cornering at the limits: Maximizing a "g-g" diagram by using feedforward trail-braking and throttle-on-exit," in *Proceedings of the International Federation of Automatic Control (IFAC) Symposium on Advances in Automotive Control*, Munich, Germany, July 2010.

[151] R. Kümmerle, D. Hähnle, D. Dolgov, S. Thrun, and W. Burgard, "Autonomous driving in a multi-level parking structure," in *IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009, pp. 3395–3400.

[152] M. Kurdej, J. Moras, V. Cherfaoui, and P. Bonnifait, "Map-aided evidential grids for driving scene understanding," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 30–41, 2015.

[153] R. Labayrade, C. Royere, and D. Aubert, "A collision mitigation system using laser scanner and stereovision fusion and its assessment," in *IEEE Intelligent Vehicles Symposium*, Las Vegas (NV), USA, June 2005, pp. 441–446.

[154] L. Lamard, R. Chapius, and J.-P. Boyer, "A comparison of two different tracking algorithms is provided for real application," in *IEEE Intelligent Vehicles Symposium*, Alcalá de Henares, Spain, June 2012, pp. 414–419.

[155] ——, "Dealing with occlusions with multi targets tracking algorithms for the real road context," in *IEEE Intelligent Vehicles Symposium*, Alcalá de Henares, Spain, June 2012, pp. 371–376.

[156] ——, "CPHD filter addressing occlusions with pedestrians and vehicles tracking," in *IEEE Intelligent Vehicles Symposium*, Gold Coast, Australia, June 2013, pp. 1125–1130.

[157] Y.-C. Lee and T. Hsiao, "Object tracking via the probability-based segmentation using laser range images," in *IEEE Intelligent Vehicles Symposium*, San Diego (CA), USA, June 2010, pp. 197–202.

[158] S. Lefebvre and S. Ambellouis, "Vehicle detection and tracking using mean shift segmentation on semi-dense disparity maps," in *IEEE Intelligent Vehicles Symposium*, Alcalá de Henares, Spain, June 2012, pp. 855–860.

[159] X. Li, "Integrated real-time estimation of clutter density for tracking," *IEEE Transactions on Signal Processing*, vol. 48, no. 10, pp. 2797–2805, October 2000.

[160] D. Linzmeier, "Real-time detection of pedestrians from a moving vehicle using thermopile and radar sensor fusion," Ph.D. dissertation, Technische Universität Chemnitz, July 2006.

[161] C. Löper, C. Brunken, G. Thomaidis, S. Lapoehn, P. Fouopi, H. Mosebach, and F. Köster, "Automated valet parking as part of an integrated travel assistance," in *IEEE International Conference on Intelligent Transportation Systems*, The Hague, Netherlands, October 2013, pp. 2341–2348.

[162] R. MacLachlan, "Tracking moving objects from a moving vehicle using a laser scanner," Carnegie Mellon University, Tech. Rep. CMU-RI-TR-05-27, 2005.

[163] P. C. Mahalanobis, "On the generalized distance in statistics," *Proceedings of the National Institute of Sciences of India*, vol. 2, no. 1, pp. 49–55, 1936.

[164] R. Mahler, *Data Fusion Hand Book*. Boca Raton, FL: CRC Press, 2001, ch. Random Set Theory for Target Tracking and Identification.

[165] ——, "Multitarget bayes filtering via first-order multitarget moments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1152–1178, October 2003.

[166] M. Mählisch, "Filtersynthese zur simultanen Minimierung von Existenz-, Assoziations- und Zustandsunsicherheiten in der Fahrzeugumfelderfassung mit heterogenen Sensordaten," Ph.D. dissertation, University of Ulm, August 2009.

[167] M. Mählisch, R. Hering, W. Ritter, and K. Dietmayer, "Heterogeneous fusion of video, lidar, and ESP data for automotive ACC vehicle tracking," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Heidelberg, Germany, September 2006, pp. 139–1444.

[168] M. Mählisch, T. Kauderer, W. Ritter, and K. Dietmayer, "Feature-level video and multibeam lidar sensor fusion for full-speed acc state estimation," in *4th International Workshop on Intelligent Transportation*, Hamburg, Germany, March 2007.

[169] M. Mählisch, W. Ritter, and K. Dietmayer, "De-cluttering with integrated probabilistic data association for multisensor multitarget ACC vehicle tracking," in *IEEE Intelligent Vehicles Symposium*, Istanbul, Turkey, June 2007, pp. 178–183.

[170] ——, "Feature level video and lidar sensor fusion for acc stop&go using joint integrated probabilistic data association," in *6th European Congress and Exhibition on Intelligent Transport Systems and Services*, Aalborg, Denmark, June 2007.

[171] M. Mählisch, R. Schweiger, W. Ritter, and K. Dietmayer, "Sensorfusion using spatio-temporal aligned video and lidar for improved vehicle detection," in *IEEE Intelligent Vehicles Symposium*, Tokyo, Japan, June 2006, pp. 424–429.

[172] M. Mählisch, M. Szczot, O. Löhlein, M. Munz, and K. Dietmayer, "Simultaneous processing of multitarget state measurements and object individual sensory existence evidence with the joint integrated probabilistic data association filter," in *5th International Workshop on Intelligent Transportation*, Hamburg, Germany, March 2008.

[173] J. Markoff, "Smarter than you think: Google cars drive themselves, in traffic," The New York Times, October 2010. [Online]. Available: http://www.nytimes.com/2010/10/10/science/10google.html

[174] S. Matzka and R. Altendorfer, "A comparison of track-to-track fusion algorithms for automotive sensor fusion," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Seol, Korea, August 2008, pp. 189–194.

[175] M. Maurer, R. Behringer, S. Furst, F. Thomanek, and E. D. Dickmanns, "A compact vision system for road vehicle guidance," in *13th International Conference on Pattern Recognition*, Vienna, Austria, August 1996, pp. 313–317.

[176] L. McGee and S. Schmidt, "Discovery of the kalman filter as a practical tool for aerospace and industry," National Aeronautics and Space Administration, Tech. Rep. 86857, 1985.

[177] D. Meissner, S. Reuter, and K. Dietmayer, "Road user tracking at intersections using a multiple-model PHD filter," in *IEEE Intelligent Vehicles Symposium*, Gold Coast, Australia, June 2013, pp. 377–382.

[178] A. Mendes, L. Bento, and U. Nunes, "Multi-target detection and tracking with a laserscanner," in *IEEE Intelligent Vehicles Symposium*, Parma, Italy, June 2004, pp. 796–801.

[179] Michigan Senate Bill 0169, December 2013.

[180] J. A. Mieghem, H. I. Avi-Itzhak, and R. D. Melen, "Straight line extraction using iterative total least squares methods," *Journal of Visual Communication and Image Representation*, vol. 6, no. 1, pp. 59–68, ? 1995.

[181] K. Min and J. Choi, "Design and implementation of autonomous vehicle valet parking system," in *IEEE International Conference on Intelligent Transportation Systems*, The Hague, Netherlands, October 2013, pp. 2082–2087.

[182] R. Möbus and U. Kolbe, "Multi-target multi-object tracking, sensor fusion of radar and infrared," in *IEEE Intelligent Vehicles Symposium*, Parma, Italy, June 2004, pp. 732–737.

[183] R. Möbus, A. Joos, and U. Kolbe, "Multi-target multi-object radar tracking," in *IEEE Intelligent Vehicles Symposium*, Columbus (OH), USA, June 2003, pp. 489–494.

[184] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffman, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, and A. Petrovskaya, "Junior: The stanford entry in the urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, September 2008.

[185] F. Moosmann, O. Pink, and C. Stiller, "Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion," in *IEEE Intelligent Vehicles Symposium*, Xi'an, China, June 2009, pp. 215–220.

[186] J. Moras, V. Cherfaoui, and P. Bonnifait, "Credibilist occupancy grids for vehicle perception in dynamic environments," in *IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 84–89.

[187] D. Müller, J. Pauli, M. Meuter, L. Ghosh, and S. Müller-Schneiders, "A generic video and radar data fusion system for improved target selection," in *IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, June 2011, pp. 679–684.

[188] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, March 1957.

[189] M. Muntzinger, "Zustandsschätzung mit chronologisch ungeordneten Sensordaten für die Fahrzeugumfelderfassung," Ph.D. dissertation, University of Ulm, November 2011.

[190] M. Muntzinger, S. Zuther, and K. Dietmayer, "Probability estimation for an automotive pre-crash application with short filter settling times," in *IEEE Intelligent Vehicles Symposium*, Xi'an, China, June 2009, pp. 411–416.

[191] M. Munz, M. Mählisch, and K. Dietmayer, "Generic centralized multi sensor data fusion based on probabilistic sensor and environment models for driver assistance systems," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 1, pp. 6–17, 2010.

[192] M. Munz, "Generisches Sensorfusionsframework zur gleichzeitigen Zustands- und Existenzschätzung fr die Fahrzeugumfelderkennung," Ph.D. dissertation, University of Ulm, Juli 2011.

[193] M. Munz and K. Dietmayer, "Using Dempster-Shafer-based modeling of object existence evidence in sensor fusion systems for advanced driver assistance systems," in *IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, June 2011, pp. 776–781.

[194] M. Munz, K. Dietmayer, and M. Mählisch, "A sensor independent probabilistic fusion system for driver assistance systems," in *12th International IEEE Conference on Intelligent Transportation Systems*, St. Louis (MO), USA, October 2009, pp. 1–6.

[195] ——, "Generalized fusion of heterogeneous sensor measurements for multi target tracking," in *13th International Conference on Information Fusion*, Edinburgh, Scotland, July 2010, pp. 1–8.

[196] M. Munz, M. Mählisch, J. Dickmann, and K. Dietmayer, "Probabilistic modeling of sensor properties in generic fusion systems for modern driver assistance systems," in *IEEE Intelligent Vehicles Symposium*, San Diego (CA), USA, June 2010, pp. 760–765.

[197] D. Musicki and R. Evans, "Joint integrated probabilistic data association: Jipda," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 3, pp. 1094–1099, July 2004.

[198] D. Musicki, R. Evans, and S. Stankovic, "Integrated probabilistic data association," in *31st IEEE Conference on Decision and Control*, Tucson (AZ), USA, December 1992, pp. 3796–3798.

[199] ——, "Integrated probabilistic data association," *IEEE Transactions on Automatic Control*, vol. 39, no. 6, pp. 1237–1241, June 1994.

[200] K. Naab, "Sensorik- und Signalverarbeitungsarchitekturen für Fahrerassistenz und Aktive Sicherheit," in *Aktive Sicherheit durch Fahrerassistenz*, Garching, Germany, March 2004.

[201] K. Naab and A. Frey, "Efficient in-vehicle sensor and signal processing architectures for driver assistance and active safety systems," in *12th World Congress on Intelligent Transportation Systems*, San Francisco (CA), USA, November 2005.

[202] National Highway Traffic Safety Administration, "National motor vehicle crash causation survey," United States Department of Transportation, Tech. Rep. DOT HS 811 059, July 2008.

[203] ——, "Distracted driving and driver, roadway, and environmental factors," United States Department of Transportation, Tech. Rep. DOT HS 811 380, September 2010.

[204] ——, "Traffic safety facts research note - distracted driving 2011," United States Department of Transportation, Tech. Rep. DOT HS 811 737, April 2013.

[205] Nevada Revised Statutes Chapter 482A - Autonomous Vehicles, March 2012.

[206] T.-N. Nguyen, B. Michaelis, A. Al-Hamadi, M. Tornow, and M.-M. Meinecke, "Stereo-camera-based urban environment perception using occupancy grid and object tracking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 154–165, March 2012.

[207] M. Noll and P. Rapps, "Ultraschallsensorik," in *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*, 2nd ed., H. Winner, S. Hakuli, and G. Wolf, Eds. Wiesbaden, Germany: Vieweg+Teubner, 2009, pp. 110–122.

[208] A. Novoselsky, S. Sklarz, and M. Dorfan, "Track to track fusion using out-of-sequence track information," in *10th Conference on Information Fusion*, Québec City, Canada, July 2007, pp. 1–5.

[209] D. Nuss, M. Stuebler, and K. Dietmayer, "Consistent environmental modeling by use of occupancy grid maps, digital road maps, and multi-object tracking," in *IEEE Intelligent Vehicles Symposium*, Dearborn (MI), USA, June 2014, pp. 1371–1377.

[210] D. Nuss, B. Wilking, J. Wiest, H. Deusch, S. Reuter, , and K. Dietmayer, "Decision-free true positive estimation with grid maps for multi-object tracking," in *IEEE International Conference on Intelligent Transportation Systems*, The Hague, Netherlands, October 2013, pp. 28–34.

[211] S. Ohl and M. Maurer, "A contour classifying Kalman filter based on evidence theory," in *14th IEEE International Conference on Intelligent Transportation Systems*, Washington (DC), USA, October 2011, pp. 1392–1397.

[212] ——, "Ein Kontur schätzendes Kalmanfilter mithilfe der Evidenztheorie," in *7. Workshop Fahrerassistenzsysteme*, Walting, Germany, March 2011, pp. 83–94.

[213] S. Ohl, "Fusion von Umfeld wahrnehmenden Sensoren in städtischer Umgebung," Ph.D. dissertation, Technische Universität Braunschweig, June 2014.

[214] R. Osborne, Y. Bar-Shalom, and P. Willett, "Track-to-track association with augmented state," in *14th International Conference on Information Fusion*, Chicago (IL), USA, July 2011, pp. 1–8.

[215] U. Ozguner, B. Baertlein, C. Cavello, D. Farkas, C. Hatipoglu, S. Lytle, J. Martin, F. Paynter, K. Redmill, S. Schneider, E. Walton, and J. Young, "The OSU demo '97 vehicle," in *IEEE Conference on Intelligent Transportation Systems*, Boston (MA), USA, November 1997, pp. 502–507.

[216] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Autonomous Robots*, vol. 26, no. 2-3, pp. 123–139, April 2009.

[217] D. Pfeiffer and U. Franke, "Modeling dynamic 3d environments by means of the stixel world," *IEEE Intelligent Transportation Systems Magazine*, vol. 3, no. 3, pp. 24–36, 2011.

[218] D. Pfeiffer, S. Gehrig, and N. Schneider, "Exploiting the power of stereo confidences," in *IEEE Conference on Computer Vision and Pattern Recognition*, Portland (OR), USA, June 2013, pp. 297–304.

[219] S. Pietzsch, T. D. Vu, O. Aycard, T. Hackbarth, N. Appenrodt, J. Dickmann, and B. Radig, "Results of a precrash application based on laser scanner and short-range radars," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 4, pp. 584–593, December 2009.

[220] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*. MIT Press, 1999, pp. 61–74.

[221] J. C. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin dags for multiclass classification," in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds. MIT Press, 2000, vol. 12, pp. 547–553. [Online]. Available: http://papers.nips.cc/paper/1773-large-margin-dags-for-multiclass-classification.pdf

[222] C. Premebida and U. Nunes, "A multi-target tracking and GMM-classifier for intelligent vehicles," in *IEEE International Conference on Intelligent Transportation Systems*, Toronto, Canada, September 2006, pp. 313–318.

[223] U. K. Rakowsky, "Fundamentals of the Dempster-Shafer theory and its applications to system safety and reliability modelling," in *Proceedings of the ESRA Summer Safety and Reliability Seminars*, vol. 2, July 2007, pp. 283–295.

[224] S. Rauch, T. Schaller, A. Savkin, and P. Hecker, "Hochgenaue Fahrzeugeigenlokalisierung und kollektives Erlernen hochgenauer digitaler Karten," in *12. Symposium Automatisierungssysteme, Assistenzsysteme und eingebettete Systeme für Transportmittel*, Braunschweig, Germany, Februar 2011.

[225] F. Rauskolb, K. Berger, C. Lipski, M. Magnor, K. Cornelsen, J. Effertz, T. Form, F. Graefe, S. Ohl, W. Schumacher, J.-M. Wille, P. Hecker, T. Nothdurft, M. Doering, K. Homeier, J. Morgenroth, L. Wolf, C. Basarke, C. Berger, T. Gülke, F. Klose, and B. Rumpe, "Caroline: An autonomously driving vehicle for urban environments," *Journal of Field Robotics*, vol. 25, no. 9, pp. 674–724, 2008.

[226] D. Reid, "An algorithm for tracking multiple targets," *IEEE Transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, December 1979.

[227] F. Rhaume and A. Benaskeur, "Out-of-sequence measurements filtering using forward prediction," Defence R&D Canada Valcartier, Tech. Rep. TR 2005-485, 2007.

[228] Robert Bosch GmbH, *Safety, Comfort and Convenience Systems*. West Sussex, England: John Wiley & Sons Ltd., 2006.

[229] T. Robinson, E. Chan, and E. Coelingh, "Operating platoons on public motorways: An introduction to the sartre platooning programme," in *17th World Congress on Intelligent Transportation Systems*, Busan, Korea, October 2010.

[230] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ: Prentice Hall, 2010.

[231] SAE International, "Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems," SAE International, Tech. Rep. SAE J 3016, January 2014.

[232] F. Sandblom and J. Sörstedt, "Sensor data fusion for multiple configurations," in *IEEE Intelligent Vehicles Symposium*, Dearborn (MI), USA, June 2014, pp. 1325–1331.

[233] SARTRE Project. [Online]. Available: http://www.sartre-project.eu/

[234] F. Saust, J. Wille, B. Lichte, and M. Maurer, "Autonomous vehicle guidance on Braunschweigs inner ring road within the Stadtpilot project," in *IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, June 2011, pp. 169–174.

[235] U. Scheunert, P. Lindner, E. Richter, T. Tatschke, D. Schestauber, E. Fuchs, and G. Wanielik, "Early and multi level fusion for reliable automotive safety systems," in *IEEE Intelligent Vehicles Symposium*, Istanbul, Turkey, June 2007, pp. 196–201.

[236] S. Schneider, M. Himmelsbach, T. Luettel, and H.-J. Wuensche, "Fusion vision and LIDAR - synchronization, correction and occlusion reasoning," in *IEEE Intelligent Vehicles Symposium*, San Diego (CA), USA, June 2010, pp. 388–393.

[237] R. Schubert, C. Adam, E. Richter, S. Bauer, H. Lietz, and G. Wanielik, "Generalized probabilistic data association for vehicle tracking under clutter," in *IEEE Intelligent Vehicles Symposium*, Alcalá de Henares, Spain, June 2012, pp. 962–968.

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

[239] K. Schueler, T. Weiherer, E. Bouzouraa, and U. Hofmann, "360 degree multi sensor fusion for static and dynamic obstacles," in *IEEE Intelligent Vehicles Symposium*, Alcalá de Henares, Spain, June 2012, pp. 692–697.

[240] M. Schütz, N. Appenrodt, J. Dickmann, and K. Dietmayer, "Simultaneous tracking and shape estimation with laser scanners," in *16th International Conference on Information Fusion*, Istanbul, Turkey, July 2013, pp. 885–891.

[241] ——, "Occupancy grid map-based extended object tracking," in *IEEE Intelligent Vehicles Symposium*, Dearborn (MI), USA, June 2014, pp. 1205–1210.

[242] G. Shafer, *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[243] R. Shanker, A. Jones, S. Devitt, K. Huberty, S. Flannery, W. Greene, B. Swinburne, G. Locraft, A. Wood, K. Weiss, J. Moore, A. Schenker, P. Jain, Y. Ying, S. Kakiuchi, R. Hoshino, and A. Humphrey, "Autonomous cars: Self-driving the new auto industrz paradigm," Morgan Stanley Research, Tech. Rep., November 2013.

[244] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Hoboken, NJ: John Wiley & Sons, Inc., 2006.

[245] S. Sivaraman and M. M. Trivedi, "A general active-learning framework for on-road vehicle recognition and tracking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 2, pp. 267–276, June 2010.

[246] SmartSenior. [Online]. Available: http://www.smart-senior.de/

[247] P. Smets, "Constructing the pignistic probability function in a context of uncertainty," in *Fifth Annual Conference on Uncertainty in Artificial Intelligence*, 1989, pp. 29–40.

[248] B. W. Smith, "Automated vehicles are probably legal in the united states," The Center for Internet and Society, Stanford University, Tech. Rep., 2012.

[249] O. T. Solutions. (2014) Rt3000 v2 family. [Online]. Available: http://www.oxts.com/products/rt3000-family/

[250] Stanford News Service. (2010, February) Stanford's robotic Audi to brave Pike's Peak withour a driver. [Online]. Available: http://news.stanford.edu/pr/

[251] Stanford Racing. [Online]. Available: http://cs.stanford.edu/group/roadrunner/old/

[252] Stanford University Dynamic Design Lab. [Online]. Available: http://ddl.stanford.edu/

[253] B. Steux, C. Laurgeau, L. Salesse, and D. Wautier, "Fade: a vehicle detection and tracking system featuring monocular color vision and radar data fusion," in *IEEE Intelligent Vehicles Symposium*, Versailles, France, June 2002, pp. 632–639.

[254] C. Stiller, A. Bachmann, and C. Duchow, "Machinelles Sehen," in *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*, 2nd ed., H. Winner, S. Hakuli, and G. Wolf, Eds. Wiesbaden, Germany: Vieweg+Teubner, 2009, pp. 198–222.

[255] C. Stimming and F. Ahlers, "Object contour tracking in vehicle and infrastructure laserscanner systems," in *7th International Workshop on Intelligent Transportation*, Hamburg, Germany, March 2010, pp. 35–39.

[256] D. Stüker, M. Blumentritt, and C. Prauße, "Modellbasierte Architekturauslegung und -absicherung von Fahrerassistenzfunktionen nach ISO 26262," in *28. VDI/VW-Gemeinschaftstagung*, Wolfsburg, Germany, October 2012, pp. 17–35.

[257] H. Takizawa, K. Yamada, and T. Ito, "Vehicles detection using sensor fusion," in *IEEE Intelligent Vehicles Symposium*, Parma, Italy, June 2004, pp. 238–243.

[258] G. Tanzmeister, J. Thomas, D. Wollherr, and M. Buss, "Grid-based mapping and tracking in dynamic environments using a uniform evidential environment representation," in *IEEE International Conference on Robotics and Automation*, Hong Kong, June 2014, pp. 6090–6095.

[259] Tartan Racing. [Online]. Available: http://www.tartanracing.org/

[260] Technische Universität Braunschweig. (2010, October) First automatic driving in real city traffic world wide. [Online]. Available: http://presse.rz.tu-bs.de/presseinformationen/

[261] A. Teichman, J. Levinson, and S. Thrun, "Towards 3d object recognition via classification of arbitrary object tracks," in *IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 4034–4041.

[262] G. Thomaidis, L. Spinoulas, P. Lytrivis, M. Ahrholdt, G. Grubb, and A. Amditis, "Multiple hypothesis tracking for automated vehicle perception," in *IEEE Intelligent Vehicles Symposium*, San Diego (CA), USA, June 2010, pp. 1122–1127.

[263] 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*, Baden-Baden, Germany, June 2011, pp. 1078–1083.

[264] J. Thomas, K. Stiens, S. Rauch, and R. Rojas, "Grid-based online road model estimation for advanced driver assistance systems," in *IEEE Intelligent Vehicles Symposium*, Seoul, Korea, June 2015, pp. 71–76.

[265] C. Thorpe, T. Jochem, and D. Pomerleau, "The 1997 automated highway free agent demonstration," in *IEEE Conference on Intelligent Transportation Systems*, Boston (MA), USA, November 1997, pp. 496–501.

[266] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: The MIT Press, 2005.

[267] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA grand challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[268] X. Tian and Y. Bar-Shalom, "Exact algorithms for four track-to-track fusion configurations: All you wanted to know but were afraid to ask," in *12th International Conference on Information Fusion*, Seattle (WA), USA, July 2009, pp. 537–544.

[269] ——, "Track-to-track fusion configurations and association in a sliding window," *Journal of Advances in Information Fusion*, vol. 4, no. 2, pp. 146–164, December 2009.

[270] ——, "Algorithms for asynchronous track-to-track fusion," *Journal of Advances in Information Fusion*, vol. 5, no. 2, pp. 128–138, December 2010.

[271] ——, "On algorithms for asynchronous track-to-track fusion," in *13th Conference on Information Fusion*, Edinburgh, Scotland, July 2010, pp. 1–8.

[272] M. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[273] United Nations, "Convention on road traffic," Geneva, Switzerland, September 1949.

[274] ——, "Convention on road traffic," Vienna, Austria, November 1968.

[275] United Nations Economic Commission for Europe, "Statistics of road traffic accidents in europe and north america," United Nations Economic Commission for Europe (UNECE), Tech. Rep., 2011.

[276] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, J. Dolan, D. Duggins, D. Ferguson, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. Howard, A. Kelly, D. Kohanbash, M. Likhachev, N. Miller, K. Peterson, R. Rajkumar, P. Rybski, B. Salesky, S. Scherer, Y. Woo-Seo, R. Simmons, S. Singh, J. Snider, A. Stentz,

W. Whittaker, J. Ziglar, H. Bae, B. Litkouhi, J. Nickolaou, V. Sadekar, S. Zeng, J. Struble, M. Taylor, and M. Darms, "Tartan racing: A multi-model apprach to the DARPA urban challenge," Defense Advanced Research Projects Agency, Tech. Rep., April 2007.

[277] C. Urmson, J. Anhalt, D. Bagnell, C. R. Baker, R. Bittner, M. N. Clark, J. M. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. E. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. M. Snider, A. Stentz, W. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, August 2008.

[278] A. van Zanten and F. Kost, "Bremsbasierte Assistenzfunktionen," in *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme fcr aktive Sicherheit und Komfort*, 2nd ed., H. Winner, S. Hakuli, and G. Wolf, Eds. Wiesbaden, Germany: Vieweg+Teubner, 2009, pp. 198–222.

[279] B. Vanholme, D. Gruyer, B. Lusetti, S. Glaser, and S. Mammer, "Highly automated driving on highways based on legal safety," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 333–347, March 2013.

[280] V. Vapnik, *The Nature of Statistical Learning Theory.* New York, NY: Springer, 2010.

[281] P. Viola and M. Jones, "Robust real-time object detection," in *International Journal of Computer Vision*, 2001, pp. 34–47.

[282] VisLab Intercontinental Autonomous Challenge. [Online]. Available: http://viac.vislab.it/

[283] B. Vo and W. Ma, "The gaussian mixture probability hypothesis density filter," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4091–4004, 2006.

[284] P. Waldmann, "Entwicklung eines Fahrzeugfürungssystems zum Erlernen der Ideallinie auf Rennstrecken," Ph.D. dissertation, Technische Universität Cottbus, December 2008.

[285] P. Waldmann and D. Niehues, "Der BMW Track Trainer - automatisiertes Fahren im Grenzbereich auf der Nürburgring Nordschleife," in *4. Tagung Sicherheit durch Fahrerassistenz*, Garching, Germany, April 2010.

[286] M. Walter, T. Fechner, W. Hellmann, and R. Thiel, "Lane Departue Warning," in *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*, 2nd ed., H. Winner, S. Hakuli, and G. Wolf, Eds. Wiesbaden, Germany: Vieweg+Teubner, 2009, pp. 543–553.

[287] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 889–916, September 2007.

[288] J. Wei, J. Snider, J. Kim, J. Dolan, R. Rajkumar, and B. Litkouhi, "Towards a viable autonomous driving research platform," in *IEEE Intelligent Vehicles Symposium*, Gold Coast, Australia, June 2013, pp. 763–770.

[289] K. Weiss, D. Stueker, and A. Kirchner, "Target modeling and dynamic classification for adaptive sensor data fusino," in *IEEE Intelligent Vehicles Symposium*, Columbus (OH), USA, June 2003, pp. 132–137.

[290] G. Welch and G. Bishop, "An introduction to the kalman filter," University of North Carloina, Tech. Rep. TR-95-041, 2006.

[291] S. Wender and K. Dietmayer, "Statistical approaches for vehicle environment classification at intersections with a laserscanner," in *12th World Congress on IST*, San Francisco (CA), USA, November 2005.

[292] ——, "An adaptable object classification framwork," in *IEEE Intelligent Vehicles Symposium*, Tokyo, Japan, June 2006, pp. 150–155.

[293] S. Wender, M. Schoenherr, N. Kaempchen, and K. Dietmayer, "Classification of laserscanner measurements at intersection scenarios with automatic parameter optimization," in *IEEE Intelligent Vehicles Symposium*, Las Vegas (NV), USA, June 2005, pp. 94–99.

[294] A. Westenberger, T. Huck, M. Fritzsche, T. Schwarz, and K. Dietmayer, "Temporal synchronization in multi-sensor fusion for future driver assistance systems," in *International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication*, Munich, Germany, September 2011, pp. 93–98.

[295] J. Weston and C. Watkins, "Multi-class support vector machines," Royal Holloway University of London, Tech. Rep. CSD-TR-98-04, 1998.

[296] ——, "Support vector machines for multi-class pattern recognition," in *European Symposium on Artificial Neural Networks*, Bruges, Belgium, April 1999, pp. 219–224.

[297] J. Wetmore, "Driving the dream: The history and motivations behind 60 years of automated highway systems in america," *Automotive History Review*, pp. 4–19, Summer 2003.

[298] Wikipedia. (2013, September) Vamp. [Online]. Available: http://en.wikipedia.org/wiki/VaMP

[299] J. Wille, F. Saust, and M. Maurer, "Stadtpilot: Driving autonomously on Braunschweigs inner ring road," in *IEEE Intelligent Vehicles Symposium*, San Diego (CA), USA, June 2010, pp. 506–511.

[300] H. Winner, "Frontalkollisionsschutzsysteme," in *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*, 2nd ed., H. Winner, S. Hakuli, and G. Wolf, Eds. Wiesbaden, Germany: Vieweg+Teubner, 2009, pp. 522–542.

[301] ——, "Radarsensorik," in *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*, 2nd ed., H. Winner, S. Hakuli, and G. Wolf, Eds. Wiesbaden, Germany: Vieweg+Teubner, 2009, pp. 123–171.

[302] H. Winner, B. Danner, and J. Steinle, "Adaptive Cruise Control," in *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*, 2nd ed., H. Winner, S. Hakuli, and G. Wolf, Eds. Wiesbaden, Germany: Vieweg+Teubner, 2009, pp. 478–521.

[303] H. Winner, S. Hakuli, and G. Wolf, *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*, 2nd ed. Wiesbaden, Germany: Vieweg+Teubner, 2009.

[304] S. Wood, J. Chang, T. Healy, and J. Wood, "The potential regulatory challenges of increasingly autonomous motor vehicles," *Santa Clara Law Review*, vol. 52, no. 4, pp. 1422–1502, 2012.

[305] P. Zarchan, *Fundamentals of Kalman Filtering: A Practical Approach*. Reston, VA: American Institute of Aeronautics and Astronautics, Inc., 2009.

[306] H. Zhao, C. Wang, W. Yao, F. Davoine, J. Cui, and H. Zha, "Omni-directional detection and tracking of on-road vehicles using multiple horizontal laser scanners," in *IEEE Intelligent Vehicles Symposium*, Alcalá de Henares, Spain, June 2012, pp. 57–62.

[307] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. Keller, E. Kaus, R. Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knöppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Jobs, H. Fritz, H. Mock, M. Hein, and E. Zeeb, "Making bertha drive - an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.

Publications

[308] M. Aeberhard and N. Kaempchen, "High-level sensor data fusion architecture for vehicle surround environment perception," in *8th International Workshop on Intelligent Transportation*, Hamburg, Germany, March 2011, pp. 173–178.

[309] M. Aeberhard and T. Bertram, "Object classification in a high-level sensor data fusion architecture for advanced driver assistance systems," in *IEEE Conference on Intelligent Transportation Systems*, Las Palmas de Gran Canaria, Spain, September 2015, pp. 416–422.

[310] M. Aeberhard, S. Paul, N. Kaempchen, and T. Bertram, "Object existence probability fusion using Dempster-Shafer theory in a high-level sensor data fusion architecture," in *IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, June 2011, pp. 770–775.

[311] M. Aeberhard, A. Rauch, M. Rabiega, N. Kaempchen, and T. Bertram, "Track-to-track fusion with asynchronous sensors and out-of-sequence tracks using information matrix fusion for advanced driver assistance systems," in *IEEE Intelligent Vehicles Symposium*, Alcalá de Henares, Spain, June 2012, pp. 1–6.

[312] M. Aeberhard, S. Rauch, M. Bahram, G. Tanzmeister, J. Thomas, Y. Pilat, F. Homm, W. Huber, and N. Kaempchen, "Experience, results and lessons learned from automated driving on Germany's highways," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 42–57, 2015.

[313] M. Aeberhard, S. Schlichthärle, N. Kaempchen, and T. Bertram, "Track-to-track fusion with asynchronous sensors using information matrix fusion for surround environment perception," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1717–1726, December 2012.

[314] M. Ardelt, P. Waldmann, N. Kaempchen, F. Homm, and M. Aeberhard, "Fully automatic lane change maneuvers for advanced safety systems in complex traffic situations," in *19th Aachen Colloquium Automobile and Engine Technology*, Aachen, Germany, October 2010, pp. 867–893.

[315] M. Bahram, M. Aeberhard, and D. Wollherr, "Please take over! An analysis and strategy for a driver take over request during autonomous driving," in *IEEE Intelligent Vehicles Symposium*, Seoul, Korea, June 2015, pp. 913–919.

[316] ——, "Please takeover! An analysis and strategy for a driver take over request during autonomous driving," in *IEEE Intelligent Vehicles Symposium*, Seoul, Korea, June 2015, pp. 913–919.

[317] M. Bahram, A. Lohrer, and M. Aeberhard, "Generatives prädiktionsmodell zur frühzeitigen spurwechselerkennung," in *9. FAS-Workshop Fahrerassistenzsysteme*, Walting, Germany, March 2014.

[318] M. Bahram, A. Wolf, M. Aeberhard, and D. Wollherr, "A prediction-based reactive driving strategy for highly automated driving function on freeways," in *IEEE Intelligent Vehicles Symposium*, Dearborn (MI), USA, June 2014, pp. 400–406.

[319] N. Kaempchen, M. Aeberhard, M. Ardelt, and S. Rauch, "Technologies for highly automated driving on highways," *Automobiltechnische Zeitschrift*, pp. 48–52, June 2012.

[320] N. Kaempchen, M. Aeberhard, P. Waldmann, M. Ardelt, and S. Rauch, "Der BMW Nothalteassistent: Hochautomatisiertes Fahren für mehr Sicherheit im Straßenverkehr," *Elektronik Automotive*, vol. 8/9, pp. 26–29, September 2011.

[321] M. Muntzinger, M. Aeberhard, J. Dickmann, and K. Dietmayer, "Time-to-collision estimation for pre-crash with out-of-sequence measurements," in *7th International Workshop on Intelligent Transportation*, Hamburg, Germany, March 2010, pp. 149–154.

[322] M. Muntzinger, M. Aeberhard, F. Schröder, F. Sarholz, and K. Dietmayer, "Tracking in a cluttered environment with out-of-sequence measurements," in *IEEE International Conference on Vehicular Electronics and Safety*, Pune, India, November 2009, pp. 56–61.

[323] M. Muntzinger, M. Aeberhard, S. Zuther, M. Mählish, M. Schmid, J. Dickmann, and K. Dietmayer, "Reliable automotive pre-crash system with out-of-sequence measurement processing," in *IEEE Intelligent Vehicles Symposium*, La Jolla (CA), USA, June 2010, pp. 1022–1027.

[324] S. Rauch, M. Aeberhard, and P. Hecker, "Kollektives Erlernen hochgenauer Straßenmodelle als Grundlage zur Einführung automatisierter Fahrfunktionen," in *6. Tagung Fahrerassistenz*, Munich, Germany, November 2013.

[325] S. Rauch, M. Aeberhard, and N. Kaempchen, "Autonomes Fahren auf der Autobahn - eine Potentialstudie für zukünftige Fahrerassistenzsysteme," in *5. Tagung Fahrerassistenz*, Munich, Germany, May 2012.

Supervised Student Theses

C. Bayer, "Road Course Estimation Using Dynamic Objects for Advanced Driver Assistance Systems," Master Thesis, Technische Universität München, München, Germany, September 2013.

S. Dingler, "Asynchrone Multi-Sensor Assoziation und Fusion für hochautomatisierte Fahrfunktionen – Auf Basis der Informationsmatrix Fusion in einer High-Level Fusionsarchitektur und der Behandlung von räumlich ausgedehnten Objekten," Bachelor Thesis, University of Applied Sciences Esslingen, Esslingen, Germany, June 2012.

M. Henzler, "Classification of Objects in a High-Level Sensor Data Fusion Architecture for Advanced Driver Assistance Systems," Master Thesis, University of Applied Sciences Frankfurt, Frankfurt, Germany, February 2012.

S. Paul, "Objekt-Existenzschätzung zur zuverlässigen Fahrumfelderfassung für hochautomatisierte Fahrerassistenzsysteme," Master Thesis, University of Applied Sciences Munich, Munich, Germany, March 2012.

M. Rabiega, "Development of a Concept for Asynchronous Sensor Data Fusion for Future Driver Assistance Systems," Diploma Thesis, Technische Universität München, München, Germany, November 2011.

S. Schlichthärle, "Track-to-Track Assoziation und Fusion in einer High-Level Fusionsarchitektur für hochautomatisierte Fahrfunktionen," Diploma Thesis, University of Ulm, Ulm, Germany, July 2011.

J. Zheng, "Validierung der Existenzwahrscheinlichkeit aus Objektdetektionsalgorithmen," Master Thesis, Karlsruhe Institute of Technology, Karlsruhe, Germany, June 2015.