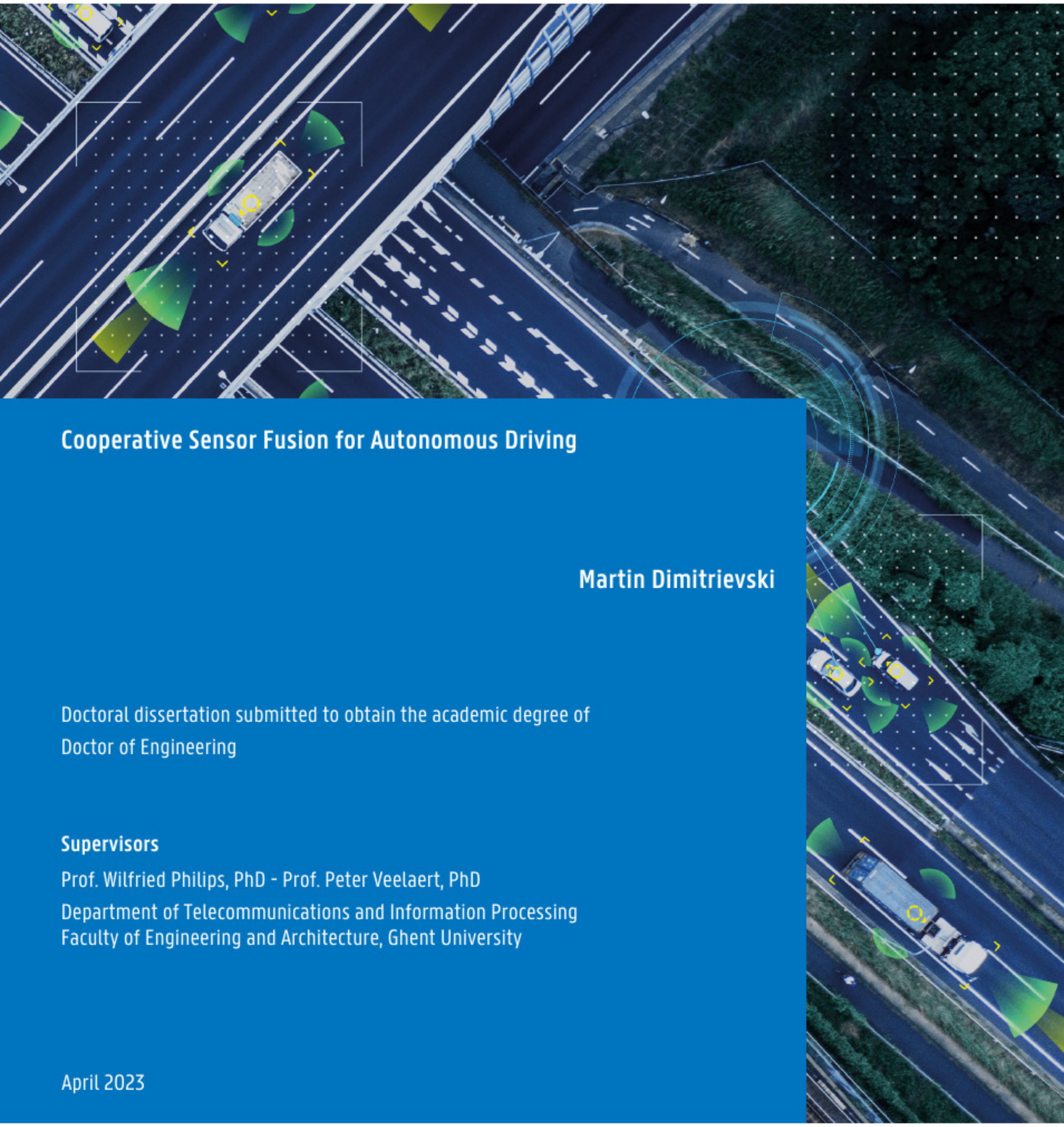
FACULTY OF ENGINEERING
AND ARCHITECTURE



Cooperative Sensor Fusion for Autonomous Driving

Martin Dimitrievski

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Engineering

Supervisors

Prof. Wilfried Philips, PhD - Prof. Peter Veelaert, PhD

Department of Telecommunications and Information Processing
Faculty of Engineering and Architecture, Ghent University

April 2023

GHENT
UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

Cooperative Sensor Fusion for Autonomous Driving

Martin Dimitrievski

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Engineering

Supervisors

Prof. Wilfried Philips, PhD - Prof. Peter Veelaert, PhD

Department of Telecommunications and Information Processing
Faculty of Engineering and Architecture, Ghent University

April 2023

GHENT UNIVERSITY

ISBN 978-94-6355-701-6
NUR 984, 983
Wettelijk depot: D/2023/10.500/33

Members of the Examination Board

Chair

Prof. Patrick De Baets, PhD, Ghent University

Other members entitled to vote

Prof. Bart Dhoedt, PhD, Ghent University
Prof. Bart Goossens, PhD, Ghent University
Sebastian Grünwedel, PhD, CARIAD, Germany
Prof. Hiep Luong, PhD, Ghent University
Prof. Adrian Munteanu, PhD, Vrije Universiteit Brussel

Supervisors

Prof. Wilfried Philips, PhD, Ghent University
Prof. Peter Veelaert, PhD, Ghent University

Acknowledgement

I wish to express my gratitude to my supervisors, whose guidance and selfless support have been instrumental throughout the years of my doctoral research. Their continuous assistance not only helped me understand the theoretical framework but also enabled me to apply seemingly abstract methodologies to real-world problems and make meaningful interpretations of the results. Furthermore, I am deeply appreciative of the members of the Jury for their thorough review and valuable suggestions, which added the final touch to the dissertation.

I would also like to acknowledge the various funding organizations for their genuine interest to venture into the unknown and provide me with the resources necessary for this work. In addition, I would like to express my sincere appreciation to all the other reviewers and academics I've come to know through my conference travels. I am particularly grateful to Reviewer 2 for their constructive criticism, which identified the weaknesses in my study but greatly enhanced the quality of my research in the end.

Although the journey of a PhD is not easy, at IPI I was surrounded by a team of thoughtful, open-minded and appreciative colleagues, some of whom I now consider as close friends. Despite our cultural differences, we worked together to make an environment where everyone felt comfortable seeking advice and providing support. Likewise, the technical and administrative staff worked with much understanding and anticipation, which helped make our team greater than the sum of its parts.

Finally, I'd like to thank my long-standing friends, who have been with me since before embarking on this academic journey. I appreciate that you never asked how my research was doing, but were always willing to listen to me rattle on about it. Above all, I thank my family and my girlfriend for giving me their unconditional love. This thesis would not have been possible without your constant encouragement and support.

Ghent, April 2023
Martin Dimitrievski

Table of Contents

List of Figures

List of Tables

List of Acronyms

AP	average precision
API	application programming interface
ACF	aggregated channel features
ADAM	adaptive moment estimation
ADAS	advanced driver assistance systems
CFAR	constant false alarm rate
CHM	contraharmonic mean filter
CNN	convolutional neural network
DAG	directed acyclical graph
DBN	dynamic Bayes network
ESS	effective sample size
FLOPS	floating point operations per second
FMCW	frequency modulated continuous wave
FOV	field of view
GPS	global position system
GPU	graphics processing unit
HMM	hidden Markov model
ICP	iterative closest point
INS	inertial navigation sensor
IOU	intersection over union

KF	Kalman filter
kNN	k-nearest neighbors
LIDAR	light detection and ranging
LSTM	long short-term memory
MAE	mean absolute error
MAP	mean average precision / maximum a posteriori
MCMC	Markov chain Monte Carlo
MHT	multiple hypothesis tracker
MOTP	multi-object tracking precision
MOTA	multi-object tracking accuracy
MT/ML	mostly tracked/mostly lost
MRF	Markov random field
MIMO	multiple input multiple output
NMS	non-maximum suppression
RADAR	radio detection and ranging
RANSAC	random sample consensus
RFS	random finite set
RGB-D	red, green, blue and depth
RJMCMC	reversible jump Markov-chain Monte Carlo
RMSE	root mean squared error
ROI	region of interest
SI(R)S	sequential importance (re)sampling
SLAM	simultaneous localization and mapping
SOM	switching observation model
SGD	stochastic gradient descent
TOF	time of flight
TID	track initialization delay
V2E	vehicle to everything
VRU	vulnerable road user

Samenvatting

De snelle groei van de markt voor consumentenelektronica maakt verschillende cruciale technologieën mogelijk die innovaties in de richting van autonoom rijden faciliteren. Goedkope sensoren, artificiële neurale netwerken en snelle draadloze communicatie vormen de kern van een nieuwe autorevolutie, waardoor voertuigen onderling kunnen communiceren, kunnen zien, interpreteren, beslissen en hun kennis kunnen uitwisselen. Op het moment van schrijven van dit proefschrift worden high-end auto's al uitgerust met semi-autonome systemen, die menselijke bestuurders helpen veiliger te rijden. Er bestaat echter nog steeds een enorme kloof in de richting van volledig autonoom rijden, en de aandacht van de menselijke bestuurder achter het stuur is nog steeds vereist.

Het onderzoeksonderwerp van dit proefschrift is de perceptie van weggebruikers in de omgeving van een rijdend voertuig. Er wordt speciale aandacht besteed aan de systeemprestaties onder reële omstandigheden. De taken van het observatiesysteem omvatten het herkennen en classificeren van interessante objecten, het inschatten van hun posities en het voorspellen van hun bedoelingen. In de context van autonome voertuigen moet het observatiesysteem nauwkeurig en betrouwbaar zijn in verschillende soorten weersomstandigheden en verkeerssituaties. Dit betekent dat de algoritmen onveranderlijk moeten zijn voor veranderingen in verlichting, atmosferische omstandigheden, elektromagnetische interferentie, sensorbewegingen, onoverzichtelijke achtergronden, enz. Hieronder bestuderen we meerdere be-staande algoritmen en stellen we meetbare verbeteringen en nieuwe methoden voor, met als doel een technologiedemonstrator te bouwen van een perceptiesysteem dat volledig autonoom rijden mogelijk maakt. De specifieke onderwerpen die in dit proefschrift worden behandeld, zijn ego-lokalisatie, objectdetectie, tracking en intentievoorspelling.

Ego-lokalisatie is de taak van het inschatten van de verandering in positie en oriëntatie van het voertuig terwijl het door de omgeving beweegt. Nauwkeurige zelflokalisatie is nodig om het waarnemingssysteem in de loop van de tijd overeen te laten komen met overeenkomstige waarnemingen. Traditionele lokalisatiesystemen, zoals systemen op basis van satelliettriangulatie, missen de nodige precisie voor autonome voertuigen. Alternatieve, op camera's gebaseerde technieken kunnen nauwkeurige schattingen van egobewegingen opleveren, maar alleen overdag wanneer de weersomstandigheden het mogelijk maken om kwaliteitsvolle foto's te maken. In plaats van te vertrouwen op externe satellietsignalen of camerabeelden die onder veel omstandigheden onbetrouwbaar kunnen zijn, stellen we een nieuwe egolokalisatiemethode voor die gebaseerd is op de registratie van probabilistische 2D-kaarten die zijn opgebouwd

uit lidar-metingen. Om aan de real-time vereiste te voldoen, schatten we de egobeweging als de relatieve offset tussen opeenvolgende 2D-kaarten met behulp van het Phase-Only Correlation-algoritme. Deze methode maakt gebruik van de Fouriertransformatie van de 2D-kaarten en is zeer robuust tegen willekeurige variaties van ruis en outliers. Experimentele resultaten laten zien dat onze methode een bijna perfecte schatting van ego-beweging heeft. Dit kan het temporeel volgen van objecten vereenvoudigen, omdat de onzekerheid over egobeweging grotendeels wordt weggenomen.

In de context van autonome voertuigen omvat objectdetectie het classificeren van verkeersdeelnemers en het lokaliseren ervan ten opzichte van het voertuig. Het meeste werk in de literatuur is gewijd aan de classificatie en lokalisatie van de relevante objecten in RGB-beelden. Objectdetectie op basis van camera's biedt echter gedeeltelijke informatie omdat objectgroottes en -afstanden niet rechtstreeks kunnen worden afgeleid uit een tweedimensionaal beeld. In deze dissertatie onderzoeken we alternatieve manieren om afstand te bepalen op camerapixels met behulp van 3D-metingen van radar en lidar. De belangrijkste technische uitdagingen komen voort uit de schaarse afstandsgegevens die, wanneer ze op het camerabeeld worden geprojecteerd, resulteren in een schaarse dieptekaart. We stellen verschillende nieuwe technieken voor diepteaanvulling voor, gebaseerd op het principe van geleide signaalreconstructie. De voorgestelde methoden extraheren contextuele informatie op hoog niveau uit het camerabeeld en gebruiken deze om de voltooiing van ontbrekende diepte te begeleiden, waarbij objectvormen en randen behouden blijven. Experimentele evaluatie toont state-of-the-art nauwkeurigheid bij het invullen van de diepte in termen van afstandsfouten, wat leidt tot verbeterde objectdetectie.

Bij het gebruik van meerdere sensoren (camera's, lidar, radar) om objecten te detecteren, kan fusie van informatie een significante verbetering van de detectieresultaten opleveren, ver-geleken met detectie op elke sensor zelf. Fusie is vooral belangrijk 's nachts of bij barre weersomstandigheden, waar individuele sensoren waarschijnlijk slecht presteren. Traditionele technieken voor sensorfusie leveren bevredigende resultaten op onder nominale omstandigheden, maar falen bij gecompromitteerd zicht. Om deze problemen op te lossen, stellen we een nieuwe coöperatieve fusiemethode voor, waarbij de sensoren hun gelokaliseerde vertrouwen uitwisselen om de detectie in gebieden van de scène met een gecompromitteerd zicht te verbeteren. De coöperatieve sensoren blijven voorwaardelijk onafhankelijk, wat een eenvoudige fusie van detectiewaarschijnlijkheden mogelijk maakt, zoals standaaard bij late fusie. Om de effectiviteit van coöperatieve fusie te evalueren, hebben we de precisie van objectdetectie grondig getest in verschillende camera/radar/lidar configuraties met behulp van meerdere datasets. De experimentele resultaten tonen aan dat coöperatieve fusie beter presteert dan late fusie bij grensgevallen zoals weinig licht en objectocclusie. Bovendien biedt coöperatieve fusie een aanzienlijk hogere robuustheid tegen sensorstoringen dan vroege fusie.

Objecttracking bevestigt de locatie van gedetecteerde objecten van meerdere detecties die over een tijdsperiode zijn gemeten. Door meerdere detecties in de loop van de tijd te aggregeren en hun vertrouwen te vergroten, worden foutieve detecties geleidelijk weggegooid, wat leidt tot een groter vertrouwen de aanweigheid van wegge-

bruikers in de directe omgeving. De voorkeursmethode voor het volgen van weggebruikers in de literatuur is Bayesiaanse filtering, die een reeks priors en waarschijnlijkheidsmodellen gebruikt om de kansverdeling van gevolgde objecten te verbeteren. Onze tracker breidt dit model uit tot het volgen van een onbekend aantal objecten met behulp van detecties van meerdere onvolmaakte (mogelijk defecte) sensoren. We stellen een probabilistisch model voor dat automatisch zijn waarschijnlijkheidsfuncties aanpast aan de lokale sensorkarakteristieken. Op deze manier kan de voorgestelde tracker omgaan met een tijdelijke verandering in detectiekwaliteit die optreedt in gebieden met occlusie, ruis of gecompromitteerd zicht. In situaties van volledige afwezigheid van detectie, gebruikt de voorgestelde tracker detecties met een lage betrouwbaarheid om de positie van weggebruikers toch te voorspellen. Experimentele evaluatie, zowel in simulatie als op vier real-world datasets, laat significante verbeteringen zien in vergelijking met andere optimale trackers. De voorgestelde tracker vertoonde state-of-the-art trackingprestaties onder algemene omstandigheden, en was vooral effectief in grensgevallen die de nauwkeurigheid van andere trackers uit de literatuur belemmeren.

Naast de demonstratiesoftware is het interdisciplinaire onderzoek dat in dit doctoraat is gedaan, geïntegreerd in verschillende systemen voor autonoom rijden die een meetbaar voordeel opleveren voor de samenleving. Delen van de methoden die in dit proefschrift worden uitgelegd, helpen de prototyperobots aan te drijven die zijn ontwikkeld binnen vijf onderzoeksprojecten die worden gefinancierd door bedrijven, evenals door lokale overheid en EU-financieringsprogramma's. Dit onderzoek resulteerde verder in twee internationale tijdschriftpublicaties en tien publicaties in de proceedings van internationale conferenties.

Summary

The rapid growth of the consumer electronics market is advancing several key technologies that facilitate innovations toward autonomous driving. Affordable sensors, mobile neural computing, as well as fast wireless communication are at the core of a new automotive revolution, where vehicles are becoming interconnected digital devices able to see, interpret, act and exchange knowledge. As of the time of writing this dissertation, high-end cars are already being equipped with semi-autonomous systems, assisting human drivers in safer driving. However, a huge gap still exists toward reaching fully autonomous vehicles, and the attention of the human driver behind the steering wheel is still required.

The core research topic addressed in this dissertation is the perception of road users present in the surroundings of a moving vehicle, under real-world constraints. The tasks of the vehicle's perception system include recognition and categorization of objects of interest, estimation of their positions, and prediction of their intentions. In the context of autonomous vehicles, the perception system needs to be accurate and reliable in diverse types of weather conditions and traffic situations. This means that the algorithms should be robust to illumination changes, atmospheric conditions, electromagnetic interference, sensor motion, cluttered backgrounds, etc. We study a wide variety of existing algorithms and propose significant improvements and novel methods, with the goal of building a technology demonstrator of a perception system that could enable fully autonomous driving. The topics covered in this thesis are ego-localization, object detection, tracking, and prediction.

Ego-motion estimation refers to the task of estimating the change in position and orientation of the vehicle as it moves through the environment. Accurate self-localization is necessary for the perception system to be able to match corresponding observations through time. Traditional localization systems such as ones based on satellite triangulation do not provide sufficient precision for autonomous vehicles. Alternative, camera-based solutions, can provide accurate ego-motion estimation, but only during daytime when the viewing conditions allow the capture of quality images. Instead of resorting to external satellite signals, or camera images which can be unreliable under many circumstances, we propose a novel ego-localization method based on the registration of 2-D probabilistic maps constructed from lidar measurements. To comply with the real-time requirements, we estimate the ego-motion as the relative offset between consecutive 2-D maps using the Phase-Only Correlation algorithm. This method uses the Fourier representation of the 2-D maps and is highly robust to measurement noise and outliers. Experimental results show that our method provides

an almost perfect estimate of the ego-motion, which simplifies the tracking of object detections over time by alleviating the need to model the ego-motion uncertainty.

In the context of autonomous vehicles, object detection involves classifying instances of road users and localizing them relative to the vehicle. Most of the effort in the literature is devoted to the classification and localization of the relevant objects in RGB image data. However, camera-based object detection provides only partial information because object sizes and distances cannot directly be inferred from a two-dimensional image. In this dissertation, we explore alternative ways for ranging camera pixels using 3-D measurements from radar and lidar. The main technical challenges come from the sparsity of range data which, when projected onto the camera image, results in a sparse depth map. We propose several novel depth completion techniques based on the principle of guided signal reconstruction. The proposed methods extract high-level contextual information from the camera image and use it to guide the completion of missing depth, preserving object shapes and edges. Experimental evaluation shows state-of-the-art depth completion accuracy in terms of distance errors, which leads to improved object detection.

When detecting objects using multiple sensors (cameras, lidar, radar), fusion of information can bring significant improvement in detection results, compared to relying on each sensor individually. Fusion is especially important at night or in harsh weather, where individual sensors tend to underperform. Traditional techniques for sensor fusion provide satisfactory results under nominal operation but fail in cases of poor visibility. To overcome these issues, we propose a novel cooperative fusion method, where the sensors exchange their localized confidences to improve the detection in areas of the scene with compromised viewing. The cooperating sensors remain conditionally independent, which allows for the easy fusion of detection likelihoods in a standard, late-fusion manner. To evaluate the effectiveness of cooperative fusion, we have thoroughly tested the precision of object detection in various camera-radar-lidar configurations using multiple datasets. The experimental results show that cooperative fusion significantly outperforms late fusion in border cases such as low light and object occlusion. Moreover, cooperative fusion offers significantly higher robustness to sensor failures than early fusion.

Object tracking corroborates the location of detected objects from multiple observations over time. By aggregating multiple observations over time and increasing their confidence, faulty detections are gradually discarded, leading to higher confidence in the actual road users in the surroundings. The preferred method for tracking road users in the literature is Bayesian filtering, which uses a set of priors and likelihood models to update the probability distribution of tracked objects. Our tracker extends this model to tracking an unknown number of objects using detections from multiple imperfect (potentially faulty) sensors. We propose a probabilistic model which automatically adapts its likelihood functions to the local sensor characteristics. This way, the proposed tracker can cope with a transient decrease in detection quality which happens in regions with occlusion, clutter, or poor visibility. Furthermore, in situations of complete absence of detection, the proposed tracker uses sub-threshold detection information to better predict the position of road users. Experimental evaluation, both

in simulation and on four real-world datasets, shows significant improvements over other optimal trackers. The proposed tracker showed state-of-the-art tracking performance under general conditions, and it was especially effective in border cases that hinder the performance of other trackers from the literature.

Beyond software demonstrators, the interdisciplinary research done in this PhD was integrated into several solutions for autonomous driving that are making a measurable benefit to society. Parts of the methods explained in this dissertation help drive the prototype robots developed within five research projects funded by companies as well as local government and EU funding programs. This research further resulted in two international journal publications and ten publications in the proceedings of international conferences.

1 Introduction

1.1 Motivation

The physical harm and mental stress brought on by automobile accidents are a huge global public health issue. Every year, traffic accidents alone result in the untimely death of almost 100,000 persons throughout Europe. Additionally, it is projected that 2.4 million people each year suffer injuries necessitating hospitalization. Besides having a well developed road infrastructure, Belgium experiences an above average rate of injuries and fatalities from traffic accidents throughout the European Union [1, 2]. Due to the unexpected need for emergency services and other competing priorities, the COVID-19 pandemic increased this strain on the hospital sector. The World Health Organization report [3] estimates that road traffic accidents result in significant economic losses to society that can amount to up to 3% of the gross domestic product of any particular country. There is strong evidence that accidents caused by motor vehicles can be avoided, making the current state of affairs even more undesirable. Over 90% of traffic accidents, according to reports on road safety, are the result of driver error [4, 5]. Even when a crash is caused primarily by a vehicle malfunction, issues with the road, or other environmental variables, certain additional human factors, such as inattention, distraction, or speeding, frequently contribute to the crash and the severity of the injuries. Reducing unnecessary driving through incentives that discourage individual car ownership and promote the use of public transit is a straightforward method of lowering traffic accidents. In places with high rates of car ownership and sparse populations, this change may not always be welcomed or economically sustainable. Thus, the number of cars on the road is not likely to decrease in the near future.

Even if we don't take road safety into account, inefficient driving leads to traffic congestion, which has serious detrimental effects on both the economy and people's mental health. According to the recent INRIX[1] global traffic scorecard report, the urban areas with the highest number of hours lost in congestion during peak commute periods compared to off-peak conditions in 2021 are: London (148 hours lost), Paris (140), Brussels (134), Moscow (108) and New York (102). This is time lost completely due to inefficient traffic which is on top of the normal commute time. The economic implications to the cities are estimated to be $8.3B (New York), $5.8B (Chicago), $3.3B (Philadelphia), 823M € (Berlin), etc.

These are expenses that can be cut by fewer people owning cars, as well as through general traffic system optimization, increased use of mass transit, and recently, telecommuting. Instead of immediately discouraging people from owning automobiles, we might attempt to improve the efficiency of the transportation system by designing smarter, autonomous vehicles that do not require a human driver. But for such objectives to be accomplished, there must be strong collaboration between business, government, and academics. Fully autonomous vehicles have been shown to benefit society, and the author believes that the development of reliable and predictable perception systems will hasten their introduction.

The consumer electronics industry is expanding quickly, pushing a number of key technologies that are essential for enabling intelligent transportation systems. At the center of a new automotive revolution, where vehicles are becoming networked digital devices capable of seeing, interpreting, acting, and sharing their knowledge and experiences, are affordable sensors, quick wireless connection, and mobile neural computing. High-end vehicles currently come with a variety of advanced safety features as standard equipment, including adaptive cruise control, automatic parking, automotive night vision, collision avoidance, emergency braking, hill descent, lane departure assistance, traffic sign recognition, vehicle to everything communication, etc. However, due to a number of technical and regulatory issues, intelligent vehicles available today cannot yet do fully autonomous driving.

We are in a vulnerable position given the current status of intelligent transportation systems because they may provide users with a false sense of security, which could lead to distractions and increased risk—exactly what we are trying to avoid. System whose accuracy, and more critically, uncertainty, can be better understood are thus clearly needed. Large datasets that can be used to train autonomous robots have benefited to some extent from advances in sensors and computers, but more progress in our fundamental understanding of artificial intelligence and robotics is still required. Deep learning-trained artificial neural networks have demonstrated ground-breaking perception accuracy and a remarkable capacity for adaptation to unforeseen circumstances. However, there is still a significant discrepancy between the accuracy reported in scientific literature and what is actually achieved in deployed systems. This thesis makes an attempt to close this gap by applying algorithms from well-established probabilistic theory to the practical problems encountered in everyday life. The main innovation

[1] https://inrix.com/scorecard/

is the addition of sensor-sensor feedback loops, which enhance a vehicle's ability to perceive its surroundings under poor viewing conditions, such as sensor failures, unexpected noise patterns, and ambiguities brought on by object occlusion. Some of the algorithms were already being used in prototype vehicles, and the claimed increases in accuracy and robustness were peer-reviewed and painstakingly tested on actual data.

1.2 Problem statement

An autonomous vehicle is a type of mobile robot that can navigate on the ground without human input by using its sensors and control systems. The precise definition of an autonomous vehicle varies across the literature and refers to various degrees of automation, from assistive to fully autonomous. The driving automation features in intelligent vehicles can be divided into five levels, starting with level 1 (driver assistance features) and going all the way up to level 5 (features that enable full driving automation), according to the report [6] published by the Society of Automotive Engineers (SAE). In all cases, an autonomous car improves upon a human-driven car by applying four fundamental technologies: ego-localization, environment perception, and map building, path planning and decision-making, and motion control. The concepts in this thesis apply to all levels of autonomous driving in a self-driving car, meaning a vehicle using car automation to achieve partial or complete driving autonomy.

One of the main tasks in autonomous driving is the prevention and reduction of the severity of collision with obstacles or other road users through the use of scene perception and interpretation. A perception system needs to build accurate representation of the position and intent of all objects in the environment including itself. Even for the most basic collision warning tasks, which provide only a warning when the car gets too close to an object, the system requires real-time detection with high accuracy covering all weather and traffic conditions. Static objects need to be quickly detected and ranged while objects in motion need to also be tracked over time. Then, by comparing the current traffic situation to a known map of the environment and putting its own position on this map, the self-driving system needs to compute a trajectory with optimal safety, comfort and energy consumption. Such a set of requirements poses a variety of real-world as well as theoretical difficulties, both in terms of hardware and software. The issue of real-time road user perception in all weather conditions will be addressed in this thesis. This book divides the perception problem into three tasks—ego-localization, object detection, and tracking—each of which is covered in a different chapter.

Ego-localization is the task of estimating one's location and orientation relative to a reference point in the environment. For a perception system, this essential piece of information is used to match current observations to observed objects from the past. A well performing ego-localization method needs to be both accurate as well as robust to allow for the uninterrupted autonomous navigation of the vehicle. Currently installed satellite positioning does not have the accuracy needed for obstacle detection, tracking and collision avoidance. Moreover, satellite reception is not always available,

especially while driving in "urban canyons" or when driving in tunnels. State-of-the-art methods from the literature extend satellite-based ego-localization with data from the onboard perception sensors for more precise positioning relative to known markers in the environment. These methods result in centimeter accurate positioning but the accuracy can deteriorate at night, in bad weather and generally anytime when the viewing conditions are poor.

Object detection refers to the task of estimating the position, shape and category of objects of interest from sensor measurements. In the literature, this task is often described as the instantaneous interpretation of data taken at a single time instance. Since an autonomous vehicle has an array of sensors, object detection in this context is performed by fusing the measurements of all sensors synchronized to a given time instance. A significant part of the research was focused towards the reconstruction of accurate depth images for the ranging of objects detected by a camera. The goal of the proposed methods is to reconstruct or complete depth values for each pixel in the camera image, which trivializes the process of object ranging because the distance to an object can simply be looked up within the image area of a bounding box.

The two main challenges of sensor fusion in object detection are first: achieving maximal confidence in the detected objects against the background or clutter, and second: reducing positional uncertainty. These challenges are often achieved by applying early or late sensor fusion on the aggregated sensor data or on individually processed sensor information respectively. This thesis proposes the concept of *cooperative fusion*, an improvement on the paradigm of late-fusion where the strengths of one sensor are used to mitigate the weaknesses of another by allowing an interaction between sensors using sensor-agnostic feedback loops.

Object tracking refers to the task of aggregating detection information processed over a longer time period. The main objective in tracking is to maximize the confidence in the presence and location of the objects of interest. By integrating multiple observations of the same object over time, object tracking exploits the stochastic nature of sensor noise and improves the confidence of perceived objects. In autonomous driving, the car encounters many other road users which greatly complicates the association of current and past observed evidence. For optimal results, the object tracker needs to assign observations to the correct tracks and deal with appearing and disappearing objects. Moreover, multi-sensor tracking needs to also decide how to best fuse noisy, and potentially missing observations from multiple sensors over time. This thesis proposes a probabilistic tracker with several adaptive strategies for aggregating multi-sensor data, predicting the object motion and reconstructing missing observations.

Combining the above-mentioned algorithms, the proposed perception system produces a picture of the environment which consists of all observed road users, their immediate intentions as well as the confidence/uncertainty in their position and the likelihood of their existence. This comprehensible output data can be used as the input for vehicle control systems, ensuring the best possible use of the sensor data available in a variety of weather and traffic conditions.

The work in this thesis begins with Chapter 2 which explains the probabilistic con-

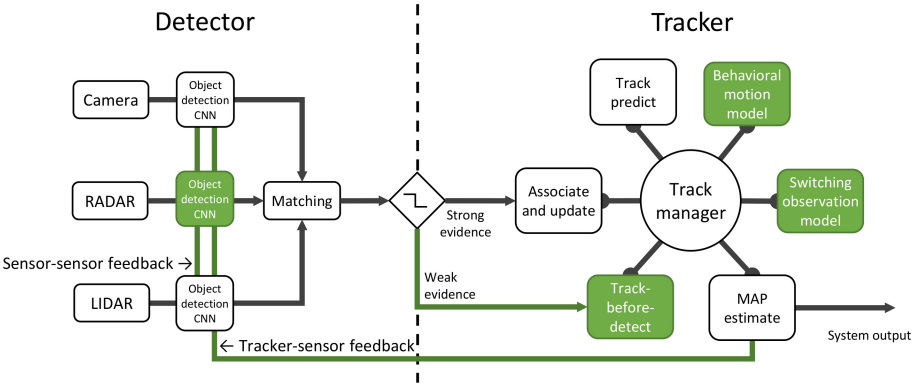


Figure 1.1: General system diagram of proposed perception system with contributions indicated in green.

cepts and models that will be used throughout the rest of the text. The remaining body of work consists of three significant chapters that present contributions to three related topics in environmental perception for autonomous vehicles. Knowledge and practical experience from one chapter is incrementally applied into the next one: starting from a novel solution to the ego-localization problem, to object detection and finally object tracking. In Figure 1.1 we see a general overview of the complete perception system built as a graph of interacting algorithms and feedback loops. Indicated in green, the graph shows the most significant areas of contributions which will be explained in detail. The three studies are presented as separate chapters following a similar structure: a problem statement, overview of the literature and remaining challenges, then details of the proposed innovation and finally, experimental evaluation.

The work presented in Chapter 3 focuses on the task of ego-localization using on-board perception sensors in situations where satellite-based global positioning is inaccurate or altogether unavailable. The loss of satellite-based localization signal is a challenging border case which happens frequently in dense urban centers and is frequently identified as an area where more research is needed. The proposed innovation uses measurements from on-board sensors and takes over when satellite-based localization fails. Based on the registration of local environmental maps constructed from live lidar scans, the proposed ego-localization algorithm works without a GPS signal, it is robust to ambient light changes; effective in both daytime as well as during the night. Testing the algorithm in a batch of laboratory and real-world experiments showed state-of-the-art ego-localization accuracy as well as exceptional robustness. The estimated vehicle trajectories are accurate even in the presence of incorrect range measurements as well as outlier noise simulating operation in extreme conditions such as glare, rain, snow, fog, etc. The proposed algorithm has a high technology readiness level, being already deployed and tested on real hardware.

In Chapter 4 we present details about the work on the problems of depth image

prediction and completion. The inclusion of ranging sensors in the sensor arrays of autonomous vehicles significantly improves their capability to perceive the environment. However, matching the content seen by the cameras and the range sensors is not straight forward due to their large mismatch in sampling resolution. The main challenge in this topic is the accurate interpolation of missing depth values around object edges. In this chapter we propose several techniques that project and upscale a lidar point cloud to the resolution of a camera image using semantic information to guide the interpolation process. This line of research is especially useful in multi-sensor applications with limited computing resources where it is not feasible to run object detection on the point cloud data. For example, in a camera-lidar sensor setup, the camera image provides most of the object classification information while the lidar point cloud provides ranging of detected objects in the image. Due to the sparsity of lidar point clouds, small image bounding boxes are rarely covered by a lidar point making their ranging ambiguous. Depth completion methods are therefore highly beneficial for the accurate ranging of small bounding boxes which are usually sparsely sampled by the lidar. This chapter proposes four different depth reconstruction methods with varying accuracy and computational complexity. The most accurate method was designed as part of a public depth completion competition and achieves high accuracy while the fastest method uses classical signal processing filtering to achieve relatively accurate depth reconstruction at a very low computational cost.

The work presented in Chapter 5 analyzes various sensor fusion algorithms for the task of instantaneous object detection. Fusing multi-modal data is challenging due to the need of a common representation, the resulting data sparsity as well as robustness to sensor failures. This study focuses on the combination of cameras and range (lidar and radar) sensors, using the camera image for classification and the range data for estimation of the position of detected objects. It outlines several depth completion algorithms which apply early camera-lidar fusion to produce pixel-accurate depth maps, which, together with the camera image can be used for accurate 3-D object detection. Experimental evaluation on camera and lidar data captured by an autonomous vehicle prototype show that the proposed early-fusion techniques achieve state-of-the-art depth reconstruction in terms of pixel-level accuracy. A second contribution presented in this chapter is the *cooperative* fusion of camera and radar information for object detection with increased robustness. The proposed method of cooperating object detectors applies information feedback which transmits attention cues from one sensor to the other, improving detection precision and recall. At the same time, when one of the sensors fails, detection accuracy does not deteriorate beyond the baseline accuracy of the other sensor.

Finally, the study presented in Chapter 6 deals with the task of multi-object tracking in real-world autonomous driving settings. A special attention is given to the tracking of objects with unpredictable behavior, such as pedestrians, where the introduction of a novel behavioral motion model and non-parametric statistics led to improvements beyond the state-of-the-art. The proposed tacker was deployed on multi-sensor data captured in the real-world where its robustness to sensor failures was thoroughly tested. These experiments led to the discovery of an important gap in the tracking lit-

erature, namely tracking under faulty sensors and missing detections, an effect that is frequently observed but rarely discussed among authors. The implications of missing detections can be confounding in border cases of tracking, causing non-convergence and tracking loss in safety-critical situations. This chapter presents a second innovation in tracking aimed at border cases of missing detections where lost information is reconstructed from sub-threshold detections by tracking before detection. Experimental evaluation of the proposed method shows state-of-the-art tracking performance over multiple datasets and multiple sensor configurations even with a loss of detection as severe as 50%.

1.3 Novelties

The work explained in this thesis has been published in 2 peer-reviewed open-access articles in the MDPI journal "Sensors" and 10 peer-reviewed papers in international conferences from which 5 were published in the proceedings of the flagship conferences of the IEEE Intelligent Transportation Systems Society. The main contributions of this thesis can be summarized as follows:

- *Robust and accurate Lidar-based vehicle odometry method based on registration of occupancy maps.*
  Operating on Lidar data, which is invariant to ambient light conditions, the proposed algorithm estimates the ego-motion by registering consecutive Lidar scans using a 2-D bird eye view representation (an occupancy map). The registration is done by computing the phase-only correlation between consecutive 2-D maps using the Fourier-Mellin transform, resulting in accurate estimation of the vehicle translation and rotation. At the time of writing, this novel odometry method showed state of the art accuracy on the KITTI odometry benchmark and moreover, remains accurate under most weather conditions that can break most visual odometry methods.

- *Depth completion methods for computing accurate depth images from camera and Lidar data.*
  The first method is a Lidar-only depth completion algorithm which relies on semantical segmentation to guide a multi-lateral depth restoration filter. Experimental evaluation of this method showed promising results, however its performance remains limited due to the difficulty in obtaining high quality segmentation from the low resolution Lidar input. Next, we propose a fusion-based depth completion method which uses camera images to better guide the Lidar depth completion. To that end, we designed two novel convolutional neural networks which take camera and sparse depth images as input and produce a dense depth image as output. Since convolutional neural networks cannot easily handle sparse inputs, we propose to pre-process the sparse input depth converting it to dense depth using morphological dilation. The first network employs a pre-processing block consisting of morphological operators with manually tuned

support kernels. In the second approach, we propose a neural network that performs the morphological pre-processing using a contraharmonic mean filter whose parameters are learnable and part of the neural network model. At the time of writing, this last depth completion method showed state of the art reconstruction accuracy on the KITTI depth completion benchmark.

- *Cooperative sensor fusion method for object detection using sensor-to-sensor feedback loops.*
  This thesis proposes the novel concept of cooperative fusion for improved object detection by two or more sensors. The proposed algorithm uses low-bandwidth data links between the individual sensors which allows the exchange of detection information at runtime. We call these links feedback loops since detection information from one sensor is used as prior information to enhance the detection in the other sensor and vice versa. Practically, we implemented the cooperative feedback in two directions: range sensors to camera, and camera to range sensors where the decision boundary of object detection adapts locally to the availability of feedback information from the cooperating sensors. The main benefit of this novelty is that the multi-sensor system retains the maximal robustness of late fusion with an increase of performance similar to that of early fusion approaches. The concept was tested and shown to be effective in several sensor configurations including cameras, Radar and Lidar.

- *Multi-object tracking in the presence of sensor failures and missing detections.*
  While convergence and optimality of traditional filtering-based object trackers has been widely studied, little is known of the tracking performance under real-world constraints. We propose a particle filter tracker which uses an adaptive (switching) observation model that can switch between sensor modes of operation at runtime. The benefit of this approach is that optimal tracking can be sustained even in situations of intermittent sensor failures such as degradation due to low light, glare, clutter, occlusion, etc. Furthermore, we propose a novel two-step tracking update algorithm which uses confident detections to update well matching track hypotheses, but switches to weak detection cues to update hypotheses which are unsupported by confident detections. The novelty of the approach is that we use an efficient structure which stores the weak multi-sensor detection cues, enabling the tracker to operate in real-time. These improvements were thoroughly tested on 4 datasets and different sensor configurations of cameras, Radar and Lidar.

Publications in international journals

The following research papers have been published in peer-reviewed scientific journals as the first author.

1. “Behavioral pedestrian tracking using a camera and lidar sensors on a moving

vehicle", Martin Dimitrievski, Peter Veelaert and Wilfried Philips (2019) SENSORS. 19(2), DOI: 10.3390/s19020391. [7]

2. "Cooperative multi-sensor tracking of vulnerable road users in the presence of missing detections", Martin Dimitrievski, David Van Hamme, Peter Veelaert and Wilfried Philips (2020) SENSORS. 20(17), DOI: 10.3390/s20174817. [8]

Publications in international conferences

The following research papers have been published in international peer-reviewed scientific conferences as the first author.

1. "Robust matching of occupancy maps for odometry in autonomous vehicles", Martin Dimitrievski, David Van Hamme, Peter Veelaert and Wilfried Philips, Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. 3. p.626-633, DOI: 10.5220/0005719006260633. [9]

2. "Semantically aware multilateral filter for depth upsampling in automotive LiDAR point clouds", Martin Dimitrievski, Peter Veelaert and Wilfried Philips, in the 2017 IEEE Intelligent Vehicles Symposium (IV) p.1058-1063, DOI: 10.1109/IVS.2017.7995854. [10]

3. "High resolution depth reconstruction from monocular images and sparse point clouds using deep convolutional neural network", Martin Dimitrievski, Bart Goossens, Peter Veelaert and Wilfried Philips (2017), Unconventional and Indirect Imaging, Image Reconstruction, and Wavefront Sensing in Proc. of SPIE 10410, DOI: 10.1117/12.2273959. [11]

4. "Information feedback loop for improved pedestrian detection in an autonomous perception system", Martin Dimitrievski, Peter Veelaert and Wilfried Philips, in the 21st International Conference on Intelligent Transportation Systems (ITSC), pages 3119–3124, 2018 p.3119-3124, DOI: 10.1109/ITSC.2018.8569968. [12]

5. "Learning morphological operators for depth completion", Martin Dimitrievski, Peter Veelaert and Wilfried Philips, in the 19th International Conference, ACIVS 2018, p.450-461, DOI: 10.1007/978-3-030-01449-0_38. [13]

6. "People tracking by cooperative fusion of RADAR and camera sensors", Martin Dimitrievski, Lennert Jacobs, Peter Veelaert and Wilfried Philips (2019) IEEE Intelligent Transportation Systems Conference - ITSC 2019. p.509-514, DOI:10.1109/ITSC.2019.8917238. [14]

7. "Tracking road users by cooperative fusion of radar and camera sensors", Martin Dimitrievski, David Van Hamme, Lennert Jacobs, Peter Veelaert, Heidi Steendam and Wilfried Philips (2019) 26th Symposium on Communications and Vehicular Technology in the Benelux (SCVT 2019), Abstracts. [15]

8. “Weakly supervised deep learning method for vulnerable road user detection in FMCW radar”, Martin Dimitrievski, Ivana Shopovska, David Van Hamme, Peter Veelaert and Wilfried Philips (2020) 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), Proceedings. In IEEE International Conference on Intelligent Transportation Systems-ITSC, DOI: 10.1109/ITSC45102.2020.9294399. [16]

9. “Automatic labeling of vulnerable road users in multi-sensor data”, Martin Dimitrievski, Ivana Shopovska, David Van Hamme, Peter Veelaert and Wilfried Philips (2021) 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). p.2623-2630, DOI: 10.1109/ITSC48978.2021.9564692. [17]

10. “Perception system based on cooperative fusion of lidar and cameras”, Martin Dimitrievski, David Van Hamme, and Wilfried Philips (2022) 2022 IEEE SENSORS Conference (SENSORS2022),
DOI: 10.1109/ENSORS52175.2022.9967331. [18]

Supervised theses

- “Pedestrian detection using heterogeneous sensor data”, Master’s thesis by Dwight Kerkhove, Ghent University, 2016
- “Multicameratracking van weggebruikers op kruispunten”, Master’s thesis by Nicolas Gelders, Ghent University, 2017
- “Object detection in LiDAR point clouds for autonomous vehicles”, Master’s thesis by Alexander Vandenbulcke, Ghent University, 2017
- “Semantical reconstruction of depth images with deep convolutional neural networks”, Master’s thesis by Hao Qin, 2018
- “Automatisatie van zinktransport op galvanisatie”, Master’s thesis by Ben Vyvey, Ghent University, 2019
- “Simultaneous localization and mapping using automotive”, Master’s thesis by Adam Tassier, Ghent University 2020
- “A Deep Learning Approach for Moving People Detection Based on FMCW Radar”, Master’s thesis by Adriaan Van Haecke, Ghent University, 2021
- “Towards Reliable Environmental Perception for Autonomous Vehicles”, Master’s thesis by Tristan Dhoedt, Ghent University, 2022

2

Environmental perception framework

2.1 Introduction

In this chapter we present the concepts and principles which will serve as the foundation for the proposed systems. Throughout the thesis we will be discussing the concepts of vehicle intelligence which enable a vehicle to operate autonomously by perceiving the environment and taking responsive actions. It comprises four fundamental technologies: environment perception and modeling, localization and map building, path planning and decision-making and motion control [19]. This thesis is focused on the tasks of environment perception and localization as deemed to be the main requirements for autonomous driving.

One main requirement to intelligent vehicles is that they need to be able to perceive and understand their surroundings in real time. Environmental perception, in this context, refers to the task of interpreting the type, position and size of objects relevant to the autonomous vehicle and tracking them through time. Since the vehicle is also in motion, it is also crucial that the perception system knows its own position and speed with respect to the environment. This way the objects of interest can be detected and tracked in a global coordinate system which in turn will minimize the complexity of path planning and decision making algorithms.

The proposed systems are inspired by the biological perception system of the human brain which interprets the information using stereo vision, contextual information and prior experiences. Mirroring human perception, most systems in the literature usually comprise of a camera and a ranging device. The camera is used to interpret the scene from the visual content while range data is also needed to localize the interpreted objects. In this chapter we will discuss the parameterization of the environment

and modeling the sensor measurements which represent information processed from the raw sensor data, for example a box around a pedestrian in the camera image. Additionally, we will present the concepts of sensor fusion which bring together information from multiple sensors in order to improve the confidence or localization of perceived objects.

Object detection and tracking provides the autonomous vehicle with the position and type of objects interest. Very importantly, too, the system has to do this under various weather conditions in order to enable safe driving at all times. Relying on a single camera for detecting objects can be risky because camera images become noisy at night, lose contrast during haze, fog or rain, glare when looking at bright light sources, camera lenses get dirty over time, etc. Moreover, measuring distance to objects with cameras is ambiguous due to the loss of depth in the image formation process. Additional sensors like Radar or Lidar provide the system with another source of detection information as well as accurate ranging. These sensors operate outside of the visible light spectrum and therefore do not suffer from the typical degradation of a camera in bad weather.

Specifically, automotive Radars use modulated electromagnetic radiation in the millimeter to centimeter wavelength to measure the distance and velocity of objects. At these wavelengths effects such as clutter from fog/rain/mist/dust are minimized making the Radar robust in situations where cameras images fail. Similarly, automotive Lidar uses modulated infra-red laser pulses to measure the distance to objects in the environment. Using its own modulated light, enables the Lidar sensor to be unhindered by the infra-red radiation of the sun, while also being effective at night. Currently, Lidar sensors offer a higher sampling resolution than Radar, but this is not an intrinsic benefit of one technology over the other. The two technologies are in their early development and since they operate on different principles we deem that both should be considered when building a perception system.

It is clear to see that the high pixel resolution of cameras offers an advantage when detecting and classifying objects under good lighting conditions. In these situations the additional Radar and Lidar information mainly improves in the localization of objects, however, when the viewing conditions deteriorate we are compelled to rely heavily on the Radar and Lidar. A fusion system must therefore be designed to optimally combine the classification and ranging information from all sensors under all viewing conditions. This means that camera detection performance must not deteriorate when fused with Radar and Lidar under good viewing conditions. Similarly, when the conditions are poor for the camera, the fusion system must realize that its interpretation of the environment is unreliable.

The material that follows provides the probabilistic architecture for fusing categorization and location data from various sensors. The content is presented in an incremental manner, beginning with simpler notions for tracking a single object and progressing to the Random Finite Sets (RFS) for tracking multiple objects by the end of the section. In Section 2.2, we will begin by describing the prevailing sensor fusion ideas. The sensor models and scene parameterization are then explained in Section 2.3 and Section 2.4. We offer details on the probabilistic modeling of belief in object

existence and placement in the scene in Section 2.5 and Section 2.6. Finally, we expand these single-object concepts to a unified framework for tracking multiple objects in Section 2.7.

2.2 Sensor fusion concepts

Environmental perception can be achieved using passive and active sensors and sensing principles. Passive sensors observe the environment by sensing the radiation emitted by the environment. Active sensors, on the other hand, emit their own radiation into the environment and infer the scene structure by observing the reflections. Analysis algorithms can then be applied on this data to extract useful information. Examples of passive sensing are monocular and stereo depth estimation, while active depth sensors include Radar, Sonar, Lidar and time of flight. Historically, much of the depth sensing in computer vision applications has been based on extracting disparity information from stereo images. This is clearly evident on the evaluation page on the KITTI Stereo 2015 benchmark [20] which currently lists more than 270 algorithms. Even though stereo vision, mimicking human binocular perception, has been effective in many applications, it possesses shortcomings in flat image regions where the lack of image features makes it impossible to estimate the distance. Moreover, since disparity is inversely proportional to the perceived depth, small disparity errors may give large depth errors, especially for distant objects. Authors have also successfully used time-of-flight cameras in order to recover dense depth information without the use of a second RGB image [21]. However, time-of-flight cameras have limited operating range, especially in bright light conditions, and as such are not in the immediate focus for autonomous vehicles research.

Active range sensors such as Light Detection and Ranging (Lidar), scan the environment by shining infra-red laser beams and measuring the reflection delay in order to determine the correct distance of objects. The accuracy of lidar sensors is limited by the optical transparency of the medium i.e., the atmospheric conditions and overall clarity of the air. For example: rain droplets and mist have, on many occasions, presented a significant challenge to the Velodyne VLP-16 and Ouster OS1-128 lidars which we had available for experimentation. Rain droplets, and accumulated puddles of water on the ground, act as specular reflectors and distort or return little of the laser light back to the lidar detector. Lidar measurements in rain are commonly polluted by random reflections from seemingly empty areas while wet surfaces register as empty space. Mist and fog, on the other hand, act as ideal light diffusers and show up as large objects, often difficult to distinguish from actual solid material. In nominal conditions, however, these sensors can operate reliably in outdoor environments with usable ranges of up to 200m.

Multiple-input multiple-output (MIMO) Radar technology uses coupled radio-wave transmitters and receivers to estimate the range and velocity of an object measuring the time delays in the radiated and received energy patterns. Most commercial automotive radars operate either in the K-band, around 24GHz, or the W-band,

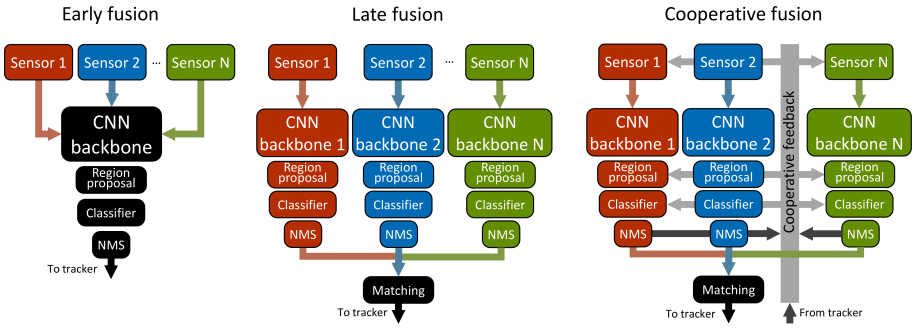


Figure 2.1: Comparison of fusion concepts.

around 77GHz of the EM spectrum. The typical wavelengths of the carrier signal in these bands ranges from a couple of millimeters to a centimeter, which makes the propagation largely unaffected by atmospheric conditions such as rain, mist, snow or fog. Owning to these unique properties, radars have been installed in numerous land, maritime and airborne platforms for the tasks of object detection and tracking. Due to safety regulations in automotive applications, the power output of installed radar devices is limited, resulting in maximum target detection ranges between 100m and 200m. While classical signal processing can be applied efficiently to detect large objects, discriminating people (also referred to as vulnerable road users throughout the thesis) from clutter in traffic environments remains a difficult task. This is manly due to the fact that people are poor radar energy reflectors and they move slowly relative to the static environment. Additionally, the effects of multipath propagation of radar signals are difficult to model explicitly due to the unknown and ever changing scene geometry. Yet, detecting moving people in radar data can be performed based on the unique pattern of motion of the human body.

It is important to keep in mind that depth perception can also be achieved implicitly through statistical modeling of observations that have a weak relationship to depth. For instance, monocular depth reconstruction attempts to recreate the depth information lost during the creation of a camera image. These algorithms attempt to reassemble depth using pixel and contour data, temporal analysis of camera motion, or offline learning. However, the majority of the literature and commercial products have utilized alternative sensing modalities since, as of right now, the accuracy of state-of-the-art monocular depth reconstruction algorithms is insufficient for automotive applications. As was mentioned in the beginning, in order to meet the strict accuracy, cost, and safety requirements, autonomous cars must rely on the fusion of many data sources. As a result, the majority of modern autonomous vehicles use a variety of sensors that are installed on the roof or dispersed throughout the chassis. This facilitates the availability of multi-modal data, which drives our research into multi-modal object detection.

In order to robustly estimate the class and range of objects in the environment we

need some level of information fusion between multiple sensors. The method used to achieve this fusion can have a substantial impact on the accuracy, robustness, complexity, and ultimately the system safety. Early or low-level fusion, Figure 2.1 left, integrates raw sensor measurements from multiple sources using a common feature extractor (shown as a CNN backbone), region of interest (ROI) extractor, classifier and non-maximum suppression (NMS). The output of early fusion is a list of objects described by their full 3-D position $\mathbf{u}_l$, shape $\mathbf{s}_l$ and activation, usually class confidence a_l. Therefore, the task of selection, matching and fusion across sensors is completely learned from training data. Using a large enough dataset, contemporary deep-learning-based early fusion methods achieve the highest accuracy at the cost of reduced robustness and interpretability of errors. A non-maximum suppression algorithm is usually applied on the classifier output in order to reduce multiple detections of a single object. NMS works by averaging shapes and detection scores of detections with a significant spatial overlap. Early fusion CNNs usually output a single output score which represents the confidence of the model for that specific task. For example, an early fusion object detector will output a detection score a_l representing the confidence of detecting an object based on the multi-sensor data input. Depending on how the model is trained, the fused confidence score can acquire any real number which does not directly represent the probability of detection. When such outputs are then used in probabilistic frameworks, this effect becomes significant and necessitates additional modeling.

Late or high-level fusion, Figure 2.1 middle, applies sensor-specific processing pipelines for each sensor and NMS on the activations of each individual sensor pipeline. The list of outputs from each of the sensors consists of the position in sensor-specific coordinates, its shape and features, and is usually supplemented with the respective confidence scores. As in early fusion, the confidence scores of individual sensors can be any real number and do not represent the detection probability. In order to merge single-sensor outputs into a multi-sensor estimate, late fusion methods apply statistical models to detections which are in close spatial proximity. Usually, this is done by projecting all sensor outputs onto a common representation and computing the joint-likelihood of detection. The benefit of this approach is that the individual object detectors can be trained independently, which is more practical, e.g. reuse of already trained networks.. The downside to late fusion compared to early fusion is that part of the information is lost in each individual processing pipeline which is then impossible to to recover by the fusion logic.

Intermediate-fusion is a broad concept which covers any fusion technique that merges multi-sensor information at an intermediate stage (using latent data representations) during the analysis process. Sensor data is usually processed by sensor-specific feature and region of interest extractors. Then, all regions of interest are matched into a common representation and classified by a common classifier. The fusion algorithm in this case has access to a richer output from the individual sensor pipelines than in late fusion, and therefore it can make a better informed classification. The fusion of detection scores in this example is delegated to the common classifier and is more akin to early fusion where the output of the network is a single activation score per

object. Intermediate fusion techniques usually balance between complexity and accuracy given the requirements of the application. In practice, most intermediate fusion methods are trained end-to-end and can be thought of a special case of early fusion since they share similar failure cases.

In this thesis, we further explore the idea of cooperative fusion for object detection, Figure 2.1 right, where the system is based on the late fusion design but adds an additional feedback line conveying cross-sensor attention cues as well as historical (from the tracker) signals. Each individual sensor has the option to tap into this feedback line at any time and utilize it to modify its parameters in response to the environment. For instance, the lidar is more effective at detecting objects at night and can indicate their existence on the feedback line. The lens aperture, sensitivity, or integration time of a camera may then be increased using this information. The location of previously tracked objects can also be added by the tracker to the feedback line, letting the detectors know where to look for a successful detection. Within these constraints, cooperative fusion can still be accomplished in a Bayesian framework where the individual sensors maintain their independence and use the feedback line as a probabilistic prior. Despite the fact that this cross-sensor information sharing may appear to point toward an early or intermediate fusion design, it is important to keep in mind that the feedback line only contains highly processed information and not data or features. In the case of a faulty sensor or corrupt feedback information, a cooperative fusion method can still revert to the baseline mode of operation as in late fusion.

2.3 Observations and hypotheses

Sensor fusion for environmental perception combines observational evidence from multiple sensors to form a more confident understanding of the state of its surroundings. Within the context of this thesis, we will use the term *sensor data* to refer to the raw sensor outputs and *sensor measurements* to refer to the processed sensor information. Sensor data is therefore the RGB camera pixel values organized as 2-D image arrays, the Radar range-azimuth-Doppler values organized as 3-D Radar cubes and the Lidar point cloud organized as a list of 3-D coordinates and reflectance values. Sensor measurements, are thus an interpretation of the sensor data and represent the observations for our environmental perception system. For the task of localization and object detection/tracking, a measurement can be thought of as a list of detections generated by the sensor processing algorithm. A detection generally consists of a position and shape (in sensor coordinates) and an activation value indicating how confident the sensor is about the detection.

The perception system generates hypotheses about the existence and position of objects of interest (in our case road users) and uses the observations to either confirm or reject them. We hereby assume two or more smart sensors, e.g., a Lidar and a camera equipped with a neural network, which capture data frames at each time instance t. They then compute a possibly empty list of detections. There is always exactly one

measurement $\mathrm{Z}_k(t)$ per sensor k and per time instant t which we define as the set:

$$\mathrm{Z}_k(t) \triangleq \left\{ \left(\mathbf{z}_1^{(k)}, k \right), ..., \left(\mathbf{z}_m^{(k)}, k \right) \right\}, \tag{2.1}$$

where each element is a tuple containing sensor-specific measurements:

$$\mathbf{z}^{(k)} = \left(\mathbf{u}^{(k)}, \mathbf{s}^{(k)}, a^{(k)}, \mathbf{f}, k \right). \tag{2.2}$$

This measurement has a variable length depending on the number of detections. Depending on the practical implementation of the detector, the elements can be ordered according to a specific property such as their activation score or the position in the data frame. This ordering property can be useful in some algorithms where for example, we want to give preference to detections with higher confidence.

A detection $\mathbf{z}^{(k)}$ is a tuple containing various pieces of information: the location $\mathbf{u}^{(k)}$ and size $\mathbf{s}^{(k)}$ of the detection in a sensor-specific coordinate system, for example a bounding box. We will use the index j to distinguish between detections $\mathbf{z}_j^{(k)}$ in the same sensor. The index k distinguishes between detections from different sensors. The detections can also contain features $\mathbf{f}$, which can be any piece of information that could be useful to track objects from frame to frame or that could help to better predict how reliable the detection really is in some more detailed analysis. All these features are extracted from the data (image, radar cube, lidar point cloud) such as texture, reflectivity, Doppler pattern, etc.

Each detection $\mathbf{z}_j^{(k)}$ also contains one or more reliability scores or activations $a_j^{(k)}$ which pertain to the bounding box defined by $\left(\mathbf{u}_j^{(k)}, \mathbf{s}_j^{(k)} \right)$. Throughout this text we will also use the functional form for the activation pertaining to a specific location and shape in sensor coordinates: $a_j^{(k)} = a^{(k)} \left(\mathbf{u}_j^{(k)}, \mathbf{s}_j^{(k)} \right)$. This reliability score is an estimate of the probability that the sensor made a mistake, a quality index indicating the degree of camouflage (e.g., pedestrian has almost the same color as the background), the degree of occlusion, etc.

A road user $(\mathbf{x}, \mathbf{g})$ is a tuple of the road user's pose $\mathbf{x}$ in world coordinates, and a feature vector $\mathbf{g}$ defining the identity of this road user by describing the road user in more detail. We use the Cartesian coordinate system to explain the pose which consist of its location and orientation. In a 3-D coordinate system the location is described as the 3-D vector (x, y, z), where the elements are the offsets of the object center with respect to the origin, while the orientation is the 3-D vector $(\theta_x, \theta_y, \theta_z)$, where each element describes the angles that the object's orientation vector makes with the coordinate axes. Similarly, in 2-D coordinates, the road user is described by its location on (x, y) on the coordinate plane and its orientation is explained by a single scalar θ. In the literature this later convention is often referred to as a 2.5-D system. The orientation of road users is always linked to their direction of travel, unless otherwise noted. The orientation of the most recent known motion is retained by static objects.

Throughout the thesis we will use the generic vector notation $\mathbf{x}$ to explain the state of a road user, referring to it as the object location, but the orientation and shape are always present in the system. The feature vector $\mathbf{g}$ can contain the height $\mathbf{h}$, but also things like the dominant color, the velocity, the material i.e., anything which is useful to predict what the road user should look like in the camera (e.g., size of bounding box and color), or in the lidar (e.g., reflectance, shape). In practice $\mathbf{g}$ will probably contain less detail then the corresponding observational features $\mathbf{s}$ and $\mathbf{f}$, which is why we have chosen to all denote them as $\mathbf{g}$. Depending on the type of sensor used to observe the state, its location and orientation relate to the measurements according to:

$$\mathbf{z}_t = h_t^{(k)}\left(\mathbf{x}_t, \mathbf{g}\right) + \mathbf{w}_t^{(k)}, \tag{2.3}$$

where $h_t^{(k)}\left(.\right)$ is a non-linear sensor-specific function and $\mathbf{w}_t^{(k)}$ is sensor-specific noise. Note that both the function and the noise are time-varying meaning that the statistics of the observed measurements can change through time. In Section 6.6 we give a detailed analysis on the shape of these models for lidar, radar and camera detectors. Furthermore, we propose a novel switching observation model which conforms to the changing characteristics of the observations over time.

We use a tack management algorithm to spawn, merge and remove hypotheses as well as assign measurements to hypotheses. The algorithm is based on the independent/local tracking principle (details follow in Section 2.7) where each road user is modeled as an individual state variable and individual sensor measurements are optimally assigned to each road user. This concept assumes that the interaction of individual road users does not influence the sensor observations enough to cause significant ambiguities in the association. By taking this assumption the track management has a significantly reduced algorithmic complexity and can be implemented to run in real-time.

2.4 Parameterization of the scene

We will hereby choose appropriate variables to model the aspects of the scene we are interested in. Primarily this is the location of road users and some distinguishing features (shape and size). First we will present the definition of the scene model and some of its properties, and then in Section 2.5, Section 2.5 and Section 2.7 we will show how these variables can be optimally estimated.

The scene is specified in terms of the number of road users and their locations. Implicitly, this also assumes that only 1 road user can be in any given location. We model the scene in terms of occupancy of a 2-D surface. The physical assumption is that all road users are on a "floor" surface (road; planar or not). For each $\mathbf{x}$ on that floor surface we define a function $\mathrm{o}(\mathbf{x})$ whose value is zero everywhere except at $\mathbf{x}$. The values of $\mathrm{o}(\mathbf{x})$ at an occupied position $\mathbf{x}$ varies throughout the literature, where the value 1 meaning occupied has been widely used in [22].

The vector $\mathbf{x}$ is scene related and as it lies on a 2D surface it is possible to represent

it with 2 coordinates. We assume that occupancy is point-like in nature where the shape of the object can be generally recovered from its class attribute. This allows us to ignore the shape of the occupant (car, bicycle, pedestrian, frontal or side view...), leading to simpler modeling of o(**x**). In practice, occupancy can be considered sparse (impulse-like), and we might as well represent o(**x**) as a set of occupied positions.

We also use variables to model the shape **g** of road users. This information is needed to evaluate likelihood of the measurements. For example a small road user should appear smaller in the picture, so person height and width could be included in **g**. Similarly, materials and surface properties could be included, or even scene related features such as "distance between **x** and and the road surface. Of course these variables only make sense for occupied positions, but we still can denote them as **g**(**x**) to uniquely associate them with an occupied position **x**. In that case the value of **g**(**x**) should be treated as "don't care" for non-occupied positions **x** and it should actually never be needed in numerical computations. Even in cases where individual road users have been identified and are being tracked, the notation **g**(**x**) makes some sense, as there can be only one road user in each position **x**.

The occupancy density $p_{\mathrm{O}(X)}\left(\mathrm{o}(\mathbf{x}) \neq 0\right)$ is related to positional uncertainty. When we wish to evaluate the probability density $p_X\left(\mathbf{x}\right)$ of the road user position **x** in a larger local neighborhood $V_{\mathbf{x}}$, knowing for sure that such a road user exists in $V_{\mathbf{x}}$ (and there is only one), then:

$$p_X\left(\mathbf{x}\right) = \int_{\mathbf{x}\in V_{\mathbf{x}}} p_{\mathrm{O}(X)}\left(\mathrm{o}(\mathbf{x}) \neq 0\right) dV_{\mathbf{x}}, \tag{2.4}$$

expresses the probability of a road user being near $\mathbf{x}$, with "near" defined by the size of $dV_{\mathbf{x}}$, see Figure 2.2 for an illustration. From a physical point of view, not all $\mathrm{o}(\mathbf{x})$ functions are possible. Specifically, road users must be at minimum distances from each other. This means that that some $\mathrm{o}(\mathbf{x})$ functions have prior probability equal to zero i.e., those for which multiple **x** points with $\mathrm{o}(\mathbf{x}) \neq 0$ are too close together. This type of requirement implies that the random variables $\mathrm{o}(\mathbf{x})$ cannot be treated as independent, because $p_{\mathrm{O}(X)}\left(\mathrm{o}(\mathbf{x})\right)$ and $p_{\mathrm{O}(X)}\left(\mathrm{o}(\mathbf{x}')\right)$ cannot both be nonzero if **x** and $\mathbf{x}'$ are too close. Locally, the scene is often sparsely populated and we can assume that for far enough **x** and $\mathbf{x}'$ the two variables $\mathrm{o}(\mathbf{x})$ and $\mathrm{o}(\mathbf{x}')$ are independent. Also, if they are far enough, any observations of the associated road users can also often be considered as independent.

In the following we will verify hypotheses of presence/absence of road users with certain features **g** at certain locations **x** by computing their prior and likelihoods and thus select the most probable ones (a posteriori). In principle this should be done for all possible combinations (**x**, **g**). Even when only considering sparse occupancy maps, this would involve evaluating priors and likelihoods for all possible scene locations (e.g., on a discretized grid). When the system starts, and we have very little prior information, we can restrict all computations to (**x**, **g**) combinations compatible with at least one of the detections in at least one sensor.

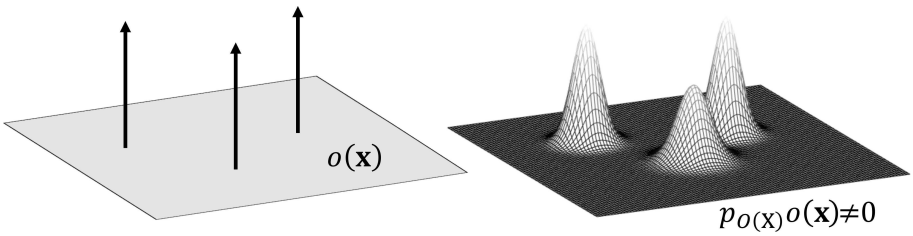


Figure 2.2: Example of the scene parameterization model with three road users. Left: the model of occupancy function with the definition that each road occupies exactly one position, which is, e.g., the projection of the center of mass on the ground plane. Right: a possible prior occupancy density, which is proportional to the probability that one of the binary variables $\mathrm{o}(\mathbf{x}')$ with $\mathbf{x}'$ near $\mathbf{x}$ is non-zero, i.e., that a location near $\mathbf{x}$ is occupied.

2.5 Belief in the object's existence

Having defined the measurements in Eq. (2.1) and how they depend on the state variables in Eq. (2.3) we will estimate the dynamic object state based on these uncertain measurements using the Bayes filter. This filter assumes the state variables are memory-less (Markov property) and additionally uses the motion patterns of the object to make a prediction of the state, which is then corrected from the observed evidence. Because of the Markov assumption, the probability of the current true state given the immediately previous one is conditionally independent of the other earlier states. Similarly, the measurement at time t is dependent only upon the current state, so is conditionally independent of all other states given the current state. The details and correctness of the Bayes filter will be analyzed in detail in Section 6.3.

We use the term belief to reflect the system's internal knowledge about the state of the objects of interest. We will denote belief over a state vector $\mathbf{x}$ by $\mathrm{bel}(\mathbf{x})$, which is an abbreviation for the posterior $p_{X|Z}(\mathbf{x}|\mathbf{z})$. It can be expressed in terms of two parts: the *belief in object existence* regardless of the details of its position and shape, and the *belief in the objects position* i.e., given that an object is present, what is the uncertainty of its position over the ground surface. Beliefs are represented through conditional probability distributions. A belief distribution assigns a probability (or density value) to each possible hypothesis with regards to the true state. Belief distributions are posterior probabilities over state variables conditioned on the available data.

Given the proposed scene model, the goal of sensor observations is to compute likelihoods for each possible $\mathrm{o}(\mathbf{x})$ scene occupancy. For example, for a camera, this likelihood computation must derive a single number $p_{I|O,G}(i|\mathrm{o},\mathbf{g})$ from the image $i(\mathbf{u})$, modeled as a function of picture coordinates, the occupancy function $\mathrm{o}(\mathbf{x})$ and the shape feature function $\mathbf{g}(\mathbf{x})$. The likelihood value should be high if the observations agree very well with the hypothesized $\mathrm{o}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$; they must be very small if they don't agree.

Object detection analysis extracts for each picture location $\mathbf{u}$ evidence for the presence of a road user in the corresponding world locations $\mathbf{x}$. The evidence usually does

not apply to a single point but rather to a region of interest, e.g., a bounding box around the road user part of the image. That region of interest is also in essence a function of $\mathbf{x}$ and the shape $\mathbf{g}$. Object detectors are usually trained to output a maximum a posteriori probability estimate of "presence at image coordinates $\mathbf{u}$". Also, often only the binary decision itself is available, and not the posterior probability of detection. In many cases the final stages of such a detector (e.g., YOLO [23, 24]) apply some thresholding to an activation value to reach their decision. By analyzing the statistics of these activations, it may in some cases be possible to estimate posterior probabilities from these activations, and with information on the training set to convert them to likelihoods.

For each sensor, we define an activation function $a(\mathbf{u})$, which is some value which the smart sensor computes and then thresholds to reach decision on presence/absence of a road user at or near $\mathbf{u}$. As explained above, this value may be missing for negative detections in actual algorithms. Some sensors will compute activation not only in terms of location but also scale or size $\mathbf{s}$.

Sensor likelihoods serve to qualify the strength of the evidence for a specific scene configuration. They answer the question: how well do the current sensor observations agree with a hypothesized occupancy map $\mathrm{o}(\mathbf{x})$ and road user shape features $\mathbf{g}(\mathbf{x})$. Assuming that we do not use the images, radar cubes or point clouds directly, but rather the activations $a(\mathbf{u})$ as "observations" produced by a smart sensor, the likelihood is a probability density functional: $p_{A|O,G}\left(a|\mathrm{o},\mathbf{g}\right)$. Its arguments are three functions $a(.)$, $\mathrm{o}(.)$ and $\mathbf{g}(.)$. In practical applications, $a(\mathbf{u})$ is an input, $\mathrm{o}(\mathbf{x})$ and $g(\mathbf{x})$ are the unknowns to compute: they need to be selected such that the product of sensor likelihoods and priors is maximized. Occupancy functions such as $\mathrm{o}(\mathbf{x})$ pose one complication: in reality the spatial resolution of observations is limited and we can only perform inference at the level of local neighborhoods. That means, we can never hope to obtain $\mathrm{o}(\mathbf{x})$ in all detail but rather we can only analyze derived measures of presence and absence, such as $H\left(\mathbf{x},\mathbf{g}\right) \triangleq \max_{x\in\Omega(\mathbf{x},\mathbf{g})}\mathrm{o}\left(\mathbf{x}'\right)$, which is non-zero if at least one road user with physical features $\mathbf{g}$ is present in $\Omega(\mathbf{x},\mathbf{g})$ which is a specific region near $\mathbf{x}$, and 0 if no road user is present in that region. In the following for the values of $H\left(\mathbf{x},\mathbf{g}\right)$, we will often write H_1 and H_0, to make the link with detection theory, where they are the standard names for the two detection cases. The shape of the region $\Omega(\mathbf{x},\mathbf{g})$ will often not be specified, but it is assumed to be in accordance with the sensor resolutions. $H\left(\mathbf{x},\mathbf{g}\right)$ obviously cannot vary arbitrarily as a function of $\mathbf{x}$ and $\mathbf{g}$. It should be composed of Dirac-delta peaks: if a road user of a given size is in a specific location, no other road user of another size can be in the same location, or even in a too-nearby location. In the following, we will sometimes make abstraction of the $\mathbf{g}$ dependency, but in principle it is always present. A practical method needs to estimate two likelihood functions:

$$p_{A,U|H,X,G}\left(a_k,\mathbf{u}_k|H_1,\mathbf{x},\mathbf{g}\right), \tag{2.5}$$

for presence and

$$p_{A,U|H,X}\left(a_k,\mathbf{u}_k|H_0,\mathbf{x},\mathbf{g}\right), \tag{2.6}$$

for absence, where k is a sensor index. Note that the latter means: "no road user present at location $\mathbf{x}$ and of specific size/shape/color as defined by $\mathbf{g}$." In other words, H_0 includes the possibility that a road user with physical features differing from $\mathbf{g}$ is present at $\mathbf{x}$. In reality, considering hypotheses depending on both $\mathbf{x}$ and $\mathbf{g}$ will be too complex. So instead we prefer a simpler hypothesis $H(\mathbf{x})$ which has the value H_1 if a road user of any size is present at $\mathbf{x}$ and H_0 otherwise. In this case the two likelihood distributions no longer depend on $\mathbf{g}$. With a slight abuse of notation we can simplify the equations to:

$$p_{A,U|H,X}\left(a_k, \mathbf{u}_k | H_1, \mathbf{x}\right), \tag{2.7}$$

for presence and

$$p_{A,U|H,X}\left(a_k, \mathbf{u}_k | H_0, \mathbf{x}\right), \tag{2.8}$$

for absence. Practically, these densities densities could be learned as follows:

1. Create training labels based on expert annotation on a dataset. Project each ground truth road user location $\mathbf{x}$ and size $\mathbf{g}$ into sensor coordinates, thus delineating a picture ROI $\Omega(\mathbf{x}, \mathbf{g})$ of "relevant" activation locations $\mathbf{u}$ and sizes $\mathbf{s}$.

2. Summarize the activation in $\Omega(\mathbf{x}, \mathbf{g})$ e.g., by picking the strongest one, or a weighted average. This also involves computing a single summarized $\mathbf{u}_k$ and $\mathbf{s}_k$ and a summarized activation a_k.

3. For each (binned) combination of $\mathbf{u}_k$ and $\mathbf{s}_k$, maintain a histogram $h(\alpha_k; \mathbf{u})$ of the a_k values. After normalization $h(\alpha_k; \mathbf{u})$ can serve as a relatively accurate approximation of $p_{A,U|H,X}\left(a_k, \mathbf{u}_k, \mathbf{s}_k | H_1, \mathbf{x}\right)$.

This of course only produces the likelihood density for presence. For absence, we need a similar training set, but this time of locations void of road users. Apart from this, the approach is similar and results in an approximation of $p_{A,U|H,X}\left(a_k, \mathbf{u}_k | H_0, \mathbf{x}\right)$. Note that the likelihood densities contain $\mathbf{x}$ and (sometimes) $\mathbf{g}$ as parameters. This is for good reason: it can indeed be inspected that road user detection becomes more difficult at larger distances. In extreme cases, one could even expect that the detector outputs very often very low activations in distant regions and this irrespective of road user presence. Then the two likelihood functions for H_1 and H_0 will be very similar and Dirac-impulse like. On the other hand, for closer distances, activation values will tend to be high for H_1 and low for H_0 . Similar considerations apply to $\mathbf{g}$. For instance, people dressed in camouflage uniforms will tend to be more difficult to detect. Small people will tend to more difficult to detect. Bicycles will be easier to detect with radar than pedestrians. In order to simplify the modeling and computations in practical applications, the following assumptions can be taken:

- Ignore the dependency on $\mathbf{x}$ and $\mathbf{g}$. In that case two simple histograms of S values should be computed on a small training set. Not much data is needed for that, which opens the possibility to continuously re-estimate the likelihoods, to adapt the changing circumstances (e.g., weather). This is the approach taken later in Section 6.3.1.

- Cluster $(\mathbf{x}, \mathbf{g})$ into a small number of context classes e.g., "close-by and large", "close-by and small", "far away and large", etc. This requires estimating two histograms per context class and obviously requires more training data.

- Based on physical principles, derive a model equation for the functional shape of the two likelihoods and estimate the model parameters from training data.

Let us assume a system composed of K sensors, each producing an observation in the form of an activation image/volume $a(\mathbf{u}, \mathbf{s})$, where $\mathbf{u}$ is a specific point in the image/volume and $\mathbf{s}$ describes the geometry and size and or shape of (e.g., bounding box) the object to which this observation applies. For instance, a smart camera examines all locations $\mathbf{u}$ for evidence on the presence of a road user at that location, but with $\mathbf{s}$ specific size and shape of a bounding box. The activation is high if there is such an evidence and low otherwise. In practice, the sensor may rather output a list of tuples $\mathbf{z}_j^{(k)} : (\mathbf{u}_j, \mathbf{s}_j, a_j)^{(k)}, j = 0, 1....$ describing strong activation peaks, where k is a sensor index. In that case, it is to be understood that:

$$a(\mathbf{u}, \mathbf{s}) \approx \sum_J a_j^{(k)} \mathrm{d}\left(\mathbf{u} - \mathbf{u}_j^{(k)}, \mathbf{s} - \mathbf{s}_j^{(k)}\right) + a_0, \tag{2.9}$$

with $\mathrm{d}(.)$ being a distance function between bounding box parameters $\mathbf{u}$ and $\mathbf{s}$ (e.g., Jaccard Index in image plane), and a_0 is a small constant which models the activation in areas without road users, usually, $a_0 \approx 0$. The formulation in Eq. (2.9) gives a practical way to evaluate the likelihood at an arbitrary coordinate $\mathbf{u}$ and shape $\mathbf{s}$ from the detected bounding boxes with location $\mathbf{u}_j$ and shape $\mathbf{s}_j$. The shape and eventual parameters of the distance function will be specific to the employed sensor and can be learned offline from training data.

To compute the belief for road user existence, with or without sensor fusion, in general we need the likelihood $p_{A^{(k)}|O,G}\left(a^{(k)}|o, \mathbf{g}\right)$, but, we will rather adopt a local view, and find evidence for either of two hypotheses about the location **x** and road user size/features **g**: a road user is present near the specified location and with features similar to the specified one (H_1) or no such road user is present (H_0). The exact meaning of "near" and "similar" is defined by the earlier introduced $\Omega(\mathbf{x}, \mathbf{g})$ and on other details such has the definition of "presence in" (center of gravity in, completely in, overlapping with...). We can then condition on the binary hypothesis variable h at $(\mathbf{x}, \mathbf{g})$, which equals either H_0 or H_1 . Moreover, for $h = H_1$, we can assume that the likelihood is very small for all (**x**, **g**), except those which project near to one of the "activation blobs" produced by the sensor. With an abuse of notation, and dropping the sensor index k for notational simplicity only, the H_1 likelihood has the following form, in which only one specific blob j near (**x**, **g**) occurs:

$$p_{A,U,S|H,X,G}\left(a_j, \mathbf{u}_j, \mathbf{s}_j | H_1, \mathbf{x}, \mathbf{g}\right). \tag{2.10}$$

The simplest (but sub-optimal) approach is to assume that this functional form is independent of the scene variables **x**, **g** and the detection location and shape ($\mathbf{u}_j$

and $\mathbf{s}_j$). In that case the equation simplifies to: $p_{A|H}\left(a_j|H_1\right)$, and this can be estimated from a simple histogram of the activations of true positive ground truth samples. The practical implementation of the fusion algorithm uses this assumption, as is later described in detail in Section 6.3.1. A slightly better version would keep the dependency on **x** and then compute different histograms for different parts of the scene. For example, we expect that the confidence of an object detector decreases with distance because of the constant resolution of the sensor data (a far away object appears smaller in the camera image). Thus, we can safely expect that the function Eq. (2.10) rolls-off with distance and the steepness can be modeled by repeating the steps 1-3 from above by binning over $\|\mathbf{x}\|$- the distance to the sensor. This modeling results to a set of histograms $h(\alpha_k; \mathbf{u}, \|\mathbf{x}\|)$

A complication occurs because sometimes the sensor will not output a detection (false negative). This case is easily handled by assuming that the sensor then still outputs a value ϵ – the "below threshold value". Therefore we can define a, which equals a_j if the sensor detected an activation blob with activation value a_j, or ϵ (signifying sub-threshold) if no blob was detected.

Without detailed knowledge of the scene structure it is safe to assume assume that false positives do not depend much on the location **u** within the image or lidar cloud, and perhaps not even much on **s**. Also, as the H_0 hypothesis stipulates there are no road users at that location, obviously the result cannot depend on **x** and **g** either. Hence the model simplifies to $p_{A|H}\left(a_j|H_0\right)$, in which a equals a_j if the sensor detected an activation blob with activation value a_j, or the special value (signifying sub-threshold) if no blob was detected. This too can be estimated from an activation histogram.

Computing the multi-sensor likelihood $p_{A^{(k)}|O,G}\left(a^{(k)}|o,\mathbf{g}\right)$ even in a local view, depends on matching the activations across each of the K sensors. This matching can be done by projecting activations into the sensor domain which has the least positional ambiguity. Within this domain, the joint-likelihood is computed as the product of individual sensor likelihoods of the detections that match the closest to the projection of $\mathrm{o}\left(\mathbf{x}\right), \mathbf{g}$. When one of the sensors is a camera the projection is usually done on the image plane. The matching of Lidar and Radar, on the other hand, can be done by projecting of the Lidar activations onto the Radar plane.

When fusing multiple sensors, we must consider all their activations as well as the priors in order to maximize the belief of existence and reach an optimal detection decision. Such a decision can be made for all parts of the scene: for each **x** we can find the relevant part of the image $(\mathbf{u}, \mathbf{s})$, radar or lidar space and the corresponding sensor activation $a(\mathbf{u}, \mathbf{s})$ (this value may be ϵ). Using the sensor-specific likelihood functions we can compute $\mathrm{llr}^{(k)}(a(\mathbf{u}, \mathbf{s}))$. According to Bayesian theory the belief in the road user presence is the log posterior probability ratio, defined as:

$$\mathrm{bel}\left(H; a^{(0)}, ..., a^{(K-1)}\right) = \ln \frac{p_{H|a^{(0)},...,a^{(K-1)}}\left(H_1|a^{(0)}\left(\mathbf{u},\mathbf{s}\right), ..., a^{(K-1)}\left(\mathbf{u},\mathbf{s}\right)\right)}{p_{H|a^{(0)},...,a^{(K-1)}}\left(H_0|a^{(0)}\left(\mathbf{u},\mathbf{s}\right), ..., a^{(K-1)}\left(\mathbf{u},\mathbf{s}\right)\right)}, \tag{2.11}$$

which determines the optimal maximum a posteriori outcome. If it is positive we

should conclude road user presence, else absence. In case we wish to penalize false positives and false negatives differently, we can also compare it to a threshold larger or smaller than 0, to minimize the cost of a wrong decision. Assuming that the K sensors operate on a different sensing principle (such as lidar, radar and camera) the observations of one sensor are not conditionally dependent on the other sensors and only depend on the state, thus according to the Bayes rule, this equates:

$$\mathrm{bel}\left(H; a^{(0)}, ..., a^{(K-1)}\right) = \ln \frac{p_H\left(H_1; \mathbf{x}\right)}{p_H\left(H_0; \mathbf{x}\right)} + \sum_{k=0}^{K-1} \mathrm{llr}^{(k)}(a^{(k)}(\mathbf{u}, \mathbf{s})), \tag{2.12}$$

where the first term is the log-prior ratio. We need to note that the definition of the priors in Bayesian filtering varies throughout the literature. The prior probability distribution defines the density function explaining the belief in the state of a random variable before observing any evidence. In object tracking over time, however, it is often the case that the belief summarized from the past (including the information from the observations) is considered as prior probability density before making an observation in the present. It will therefore be high in regions where we expect road user presence i.e., close in value to the log poster probability ratio of road user presence found after processing all data at the previous time instance $t-1$. The values $\mathrm{llr}(a^{(k)}(\mathbf{x}))$ come from the sensors and are well defied, even when a sensor outputs no detection corresponding to the 3D coordinates $\mathbf{x}$.

The hypotheses $H(\mathbf{x})$ in this case becomes: $H(\mathbf{x}) \triangleq \max_{\mathbf{x}' \in \Omega(\mathbf{x})} \mathrm{o}\left(\mathbf{x}'\right)$, which has the value H_0 if no road user exists in $\Omega\left(\mathbf{x}\right)$ and the value H_1 if one exists. Because of the assumption that very locally, only one road user can exist if $\Omega\left(\mathbf{x}\right)$ is about the same size as a typical road user, we have:

$$p_H\left(H_1; \mathbf{x}\right) = \int_{\Omega(\mathbf{x})} \mathrm{o}\left(\mathbf{x}\right) dV_{\mathbf{x}}, \tag{2.13}$$

and $p_H\left(H_0; \mathbf{x}\right) = 1 - p_H\left(H_1; \mathbf{x}\right)$.

Due to practical limitations, we can only store a limited amount of hypotheses in the computer memory which limits the extent of $\mathrm{o}\left(\mathbf{x}\right)$. As the ego-vehicle is moving forward, the hypothesis space $\mathrm{o}\left(\mathbf{x}\right)$ needs to also be able to evaluate the newly observed regions that come into view of the sensors. Therefore, we need to make a decision on how to practically model this limited hypothesis space $\mathrm{o}\left(\mathbf{x}\right)$. The two options are to use hypotheses in a local coordinate system attached to the ego-vehicle, or use a global coordinate system with hypotheses attached to Geo-locations. A local coordinate system will span the maximum sensor range and needs not be extended as the vehicle is moving. However, at each time step all of the hypotheses have to be corrected for ego-motion in order to match with the local sensor observations. Since the number of hypotheses is generally much larger than the number of observations, we chose to to use a global coordinate system for the hypotheses and only transform the (fewer) observations into global coordinates at runtime. The common reference is chosen as the Geo-location of the ego-vehicle when the system initializes. Newly

observed detections are therefore transformed into this global coordinate system using the vehicle odometry which is explained in the following chapter. This creates another practical problem, namely old hypotheses which are no longer in sensor view become irrelevant and need to be forgotten. To that end we employ a management system that removes unlikely hypotheses and is explained in detail in Section 6.8.

After converting the current observations into global coordinates we can easily evaluate the support of each hypothesis using the log-likelihood ratio. At time t the llr is computed from the llr in the past using the following recursion (dropping the geometrical parameters **x,g** and **u,s** for brevity):

$$\mathrm{llr}(a_t) \leftarrow \mathrm{llr}(a_{t-1}) + \sum_{k=0}^{K-1} \mathrm{llr}^{(k)}(a_t^{(k)}), \tag{2.14}$$

where the initial ratio is an initialization prior that represents the likelihood of observing a road user without looking at the scene:

$$\mathrm{llr}(a_0) = \ln \frac{p_H\left(H_1; \mathbf{x}\right)}{p_H\left(H_0; \mathbf{x}\right)}. \tag{2.15}$$

This ratio can be used as a measure of confidence which ranks hypotheses according to our belief in their existence.

2.6 Belief in the object's position

The belief in the object's position can take into account detection location uncertainties: instead of assigning a non-zero likelihood only when $\mathbf{u}_k$ and $\mathbf{s}_k$ exactly agree with **x** and **g** after projection, we can also assign high or non-zero values to other nearby values of **x** and **g**. This is to take into account that detectors typically produce multiple strong responses and which one is the largest can depend on random influences. The non-maximum suppression therefore risks picking a wrong, nearby location as the maximum. This approach will typically result in models with produce "likelihood blobs" in (**x**, **g**) space. Taking into account detection location uncertainties is necessary, but becomes less important if the noisy detections can be matched with no ambiguity across multiple sensors. For instance, a camera will produce high, non-zero likelihoods not for a single **x** position but rather for a "stripe-like" region, because of the projective nature of image formation. Similarly, a radar will do so for a "banana-shaped" regions along the azimuth. Detection uncertainty will cause these regions to shift, but as long as the shift is small enough, the two regions will still overlap. As a result, at least one position will have high likelihood values for both sensors. Hence, the fusion system will still detect a road user, but perhaps in a slightly wrong position.

The need for the following analysis comes from the fact that in autonomous driving the car will encounter multiple objects of interest that will cause the object detectors to

generate many, potentially ambiguous detections. Matching the detections across the multiple sensors, and moreover, matching them to the correct hypothesis over time, necessitates that we also take into account the positional uncertainty of observations. Most modern object detectors employ non-maximum suppression and produce an activation only at a limited number of sensor coordinates. The position of the true object in sensor coordinates is usually close, but not exactly at the position of the detection after non-maximum suppression. If we were to only use the sensor activation of the detection after non-max suppression, then slight variations in the position of a hypothesis would yield the same likelihood given this closest detection. Therefore, the distance between the hypothesis and the *closest* detection (also the difference in their shapes) can provide additional information for modeling the precise position of the object. In this section we will explain the concept of likelihood based on distance in sensor coordinates for optimizing the belief in the position of an object for which we are certain that it exists.

The tracking of the spatial coordinates is performed similarly by using the Bayesian framework. For a positive hypothesis H_1 defining a road user present in the hypothesis region $\Omega(\mathbf{x}, \mathbf{g})$ we wish to optimize the belief in the spatial position of that road user given the sensor measurements. The theory of Bayesian fusion is quite clear how this should be taken into account: we consider many locations $\mathbf{x}^{(i)}$ near the hypothesized position (within $\Omega(\mathbf{x}, \mathbf{g})$), evaluate Eq. (2.12) for these positions according to the prior and all likelihood models. The most likely spatial position can then be found by finding the mode of the posterior. The following analysis is for only one road user hypothesis while Section 2.7 explains how the principle can be applied when tracking multiple road users. In practice, we use a particle filter (with a finite number of $\mathbf{x}^{(i)}$) to model the uncertainty of measured positions $\mathbf{u}$ and shapes $\mathbf{s}$ of the true road user's position and shape $\mathbf{x}, \mathbf{g}$ respectively. Ignoring the variation of activation scores within a local neighborhood $\Omega(\mathbf{x}, \mathbf{g})$, the posterior distribution of $\mathbf{x}$ for a single road user is given by the following recursion:

$$\begin{gathered} p_{X_t,H|U_{0:t}^{(0)},\ldots,U_{0:t}^{(K-1)}}\left(\mathbf{x}_t, H_1|\mathbf{u}_{0:t}^{(0)},\ldots,\mathbf{u}_{0:t}^{(K-1)}\right) = \\ p_{X_t,H|U_{0:t-1}^{(0)},\ldots,U_{0:t-1}^{(K-1)}}\left(\mathbf{x}_t, H_1|\mathbf{u}_{0:t-1}^{(0)},\ldots,\mathbf{u}_{0:t-1}^{(K-1)}\right) \\ \prod_{k=0}^{K-1} p_{U_t^{(k)}|H,X_t}\left(\mathbf{u}_t^{(k)}|H_1,\mathbf{x}_t\right), \end{gathered} \tag{2.16}$$

where the multiplier is the prior and the multiplicand is a product of the K sensor models (likelihoods with respect to the location measurements). The sensor models are generally considered to be known, either given by the manufacturer or trained from labeled data with known positions of road users. For example, the positional uncertainty of a Lidar detection $\mathbf{u}$ given a positive hypothesis H_1 is accurately modeled by a multi-variate Normal distribution centered around the hypothesis location $\mathbf{x}$; for a

3-D Lidar detection we have:

$$p_{U|H,X}\left(\mathbf{u}|H_1,\mathbf{x}\right)=\frac{\exp\left(-\frac{1}{2}\left(\mathbf{u}-\mathbf{x}\right)^{\mathrm{T}}\Sigma^{-1}\left(\mathbf{u}-\mathbf{x}\right)\right)}{\sqrt{\left(2\pi\right)^{3}\left|\Sigma\right|}}, \tag{2.17}$$

where Σ is a 3x3 covariance matrix, typically estimated from labeled training data. These models are explained in more detail in Section 6.6.

When tracking a single object, at time $t=0$ the prior $p_{X_t,H}\left(\mathbf{x}_t,H_1\right)$ is usually assumed uniform making the system initialize from the first set of observation $\mathbf{u}_0^{(k)}$. However, at every consecutive time step t we can make a better prediction about the position of the tracked object if we analyze its motion patterns. If we know the velocity of the object (up to a model), at time t we can compute the expected position of the object from its last known estimate at $t-1$ using $p_{X_t|X_{t-1},H}\left(\mathbf{x}_t|\mathbf{x}_{t-1},H_1\right)$. By applying a motion model $p_{X_t|X_{t-1},H}\left(\mathbf{x}_t|\mathbf{x}_{t-1},H_1\right)$ to the posterior from the previous time step we obtain a more informative information of the likely places where we can find the object in the present. Thus we can use the motion model and the past posterior to compute an estimate in the present and use it as a prior by integrating over all possible $\mathbf{x}_{t-1}$:

$$\begin{aligned}p_{X_t,H|U_{0:t-1}^{(0)},\ldots,U_{0:t-1}^{(K-1)}}\left(\mathbf{x}_t,H_1|\mathbf{u}_{0:t-1}^{(0)},\ldots,\mathbf{u}_{0:t-1}^{(K-1)}\right)&=p_{X_t|X_{t-1},H}\left(\mathbf{x}_t|\mathbf{x}_{t-1},H_1\right)\\ \int p_{X_{t-1},H|U_{0:t-1}^{(0)},\ldots,U_{0:t-1}^{(K-1)}}&\left(\mathbf{x}_{t-1},H_1|\mathbf{u}_{0:t-1}^{(0)},\ldots,\mathbf{u}_{0:t-1}^{(K-1)}\right)d\mathbf{x}_{t-1}.\end{aligned} \tag{2.18}$$

Note that here we are only interested in the positional uncertainty in the positive hypothesis case H_1 while for H_0 the same quantity is meaningless, so in most of the following analysis the hypothesis H_1 is omitted from the notation.

In real-world applications the effects of occlusion, missing detections, soft sensor failures, etc. cause ambiguities in the observations and the posterior distribution of an object's location cannot be accurately explained by Gaussian models. For example, consider a pedestrian that passes behind an occluding object. When this happens we are faced with missing detections and if the pedestrian continues walking he can reappear on the other side of the occluder or he can also turn around and reappear at the same spot that he went missing. We deem that under real-world circumstances, the positional uncertainty of road users (especially vulnerable road users) is best modeled using mixture models which allow the belief to concentrate in multiple locations of the hypothesis space at once. The particle filter formulation, explained in detail in Section 6.4, is a sampling-based implementation of the Bayes filter and offers an effective solution for computing Eq. (2.16) and has well understood performance bounds and convergence. The main benefit of using the particle filter comes from its ability to model the posterior as a multi-modal distribution using weighted Dirac impulses. Thus the posterior distribution of the location of one object is modeled as the sum of the set

of particles:

$$p_{X_t,H|U}\left(\mathbf{x},H_1|\mathbf{u}\right) \approx \sum_{i=1}^{N_{pts}} w^{(i)}\delta\left(\mathbf{x}-\mathbf{x}^{(i)}\right), \tag{2.19}$$

where $w^{(i)}$ are particle weights that sum up to 1 and the Dirac delta $\delta\left(\mathbf{x}-\mathbf{x}^{(i)}\right)$ defines the particle positions. The particle filter algorithm works in two steps, firstly it *predicts* the position of particles (and thus the shape of the posterior) by applying a motion model to each particle, and then it evaluates the likelihood of each particle using the newly observed evidence. The idea is that predicted particles which are well supported by observational evidence contribute more towards the shape of the posterior. So, in order to compute the posterior in Eq. (2.16) at time t we first apply a motion model $p_{X_{t-1},H|X_t}\left(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)},H_1\right)$ to each particle and then update their weights according to the sensor models:

$$w_t^{(i)} \propto w_{t-1}^{(i)} \prod_{k=0}^{K-1} p_{U_t^{(0)},...,U_t^{(K-1)}|H,X_t}\left(\mathbf{u}_t^{(0)},...,\mathbf{u}_t^{(K-1)}|H_1,\mathbf{x}_t^{(i)}\right). \tag{2.20}$$

Note that this gives an overview of how the positional uncertainty is modeled by a vanilla particle filter. In sections 6.6.4 and 6.7 of this thesis we will extend this formulation to a switching observation model particle filter and adapt the algorithm to continue to update with in cases of missing detections.

2.7 Tracking multiple objects

A perception system deployed in the real world needs to detect and track potentially many objects of interest. The material presented in the previous sections implicitly assume that only one object is present in the scene and that the sensor measurement may be directly mapped to the object's hypothesis. However, most applications require the simultaneous tracking of several objects and the measurement to track assignment task is challenging due to sensor noise, missed detections, and false alarms. In this part, we provide the probabilistic framework applicable to multi-object tracking based on Random Finite Set (RFS) statistics, with more details in [25].

Random finite sets provide the means to generalize the single-object Bayes filter using multi-object probability distributions over a random finite set. In Section 2.3 we defined the detections of each sensor as the set Z_k of random observations $\left\{\left(\mathbf{z}_1^{(k)},k\right),...,\left(\mathbf{z}_m^{(k)},k\right)\right\}$ and in Section 2.4 we discussed how the function $o\left(\mathbf{x}\right)$ referring to the scene occupancy can be represented as set of occupied positions without loss of generality. We will therefore use the set state variable:

$$\mathrm{X} = \left\{\mathbf{x}_1,...,\mathbf{x}_n\right\},$$

which consists of n unordered random vectors $\mathbf{x}_1,...,\mathbf{x}_n$ (referring to object states,

i.e. occupied positions), where n; $n \geq 0$, too is a *random variable*. Such a set represents also the uncertainty about the number of objects in a multi-object state and uses random vectors to represent the state of the individual objects. In 3-D tracking problems, the state space of all state vectors $\mathbf{x}$ is $\mathbb{R}^3$ whereas the finite subsets of space $\mathbb{R}^3$ are denoted as $\mathcal{F}\left(\mathbb{R}^3\right)$ and all possible subsets comprising exactly n elements are represented by $\mathcal{F}_n\left(\mathbb{R}^3\right)$.

We use the notation $\pi\left(\mathrm{X}\right)$ to indicate the multi-object probability density function which represents the uncertainty about the multi-object state X and also incorporates the uncertainty about the number of objects represented by X. The generic form of $\pi\left(\mathrm{X}\right)$ can be written as:

$$\pi\left(\mathrm{X}\right) = \begin{cases} \pi\left(\emptyset\right) & \text{if } \mathrm{X} = \emptyset, \\ \pi\left(\{\mathbf{x}_1\}\right) & \text{if } \mathrm{X} = \{\mathbf{x}_1\}, \\ \pi\left(\{\mathbf{x}_1, \mathbf{x}_2\}\right) & \text{if } \mathrm{X} = \{\mathbf{x}_1, \mathbf{x}_2\}, \\ \vdots & \vdots, \end{cases} \tag{2.21}$$

where we can write the probability that the random finite set X contains exactly n vectors as the cardinality distribution over all possible $\mathbf{x}_1, ..., \mathbf{x}_n$:

$$\Pr\left(|\mathrm{X}| = n\right) = \frac{1}{n!} \int \pi\left(\{\mathbf{x}_1, ..., \mathbf{x}_n\}\right) d\mathbf{x}_1, ..., \mathbf{x}_n. \tag{2.22}$$

If the n vectors are independently identically distributed (i.i.d.), then we can use a i.i.d. cluster RFS probability density function: $\pi\left(\mathrm{X}\right) = n!\Pr\left(|\mathrm{X}| = n\right) p\left(\mathbf{x}_1\right) \ldots p\left(\mathbf{x}_n\right)$, where $p\left(\mathbf{x}_i\right)$ denotes the spatial distribution of the i-th object. A commonly used model to represent the uncertainty of the number of objects in the literature [26] is the Poisson distribution:

$$\Pr\left(|\mathrm{X}| = n\right) = \frac{e^{-\lambda}\lambda^n}{n!}, \tag{2.23}$$

where λ is the expected number of objects, making the multi-object Poisson RFS PDF:

$$\pi\left(\mathrm{X}\right) = e^{-\lambda}\lambda^n p\left(\mathbf{x}_1\right) \ldots p\left(\mathbf{x}_n\right). \tag{2.24}$$

To complete the system, we use the Poisson RFS with intensity density: $k\left(z\right) = \lambda_c c\left(z\right)$ for modeling the process of false positives, where λ_c is the expected number of false positives per image and the probability density $c\left(z\right)$ models the variability over the measurement space.

The use of a Bernoulli RFS is another obvious technique to model the uncertainty regarding the presence of a single object. A Bernoulli RFS uses the existence probability r to indicate the existence (as discussed in Section 2.5) of an object with a spatial distribution $p\left(\mathbf{x}\right)$ (as discussed in Section 2.6). Consequently, we can use the probability $1 - r$ for an empty Bernoulli RFS. A Bernoulli distribution with parameter r gives the cardinality distribution of a Bernoulli RFS, and its probability density is

given by:

$$\pi\left(\mathrm{X}\right)=\begin{cases}1-r & \text{if } \mathrm{X}=\emptyset,\\ r\cdot p\left(\mathbf{x}\right) & \text{if } \mathrm{X}=\left\{\mathbf{x}\right\}.\end{cases} \tag{2.25}$$

A simple extension of this method, which may model M objects, will be referred to as multi-Bernoulli RFS. Assuming that the objects are independent of each other, the parameter set $\left\{\left(r_i,p_i\right)\right\}_{i=1}^{M}$completely defines a multi-Bernoulli RFS with the probability density function:

$$\pi\left(\mathrm{X}\right)=\begin{cases}\prod_{j=1}^{M}\left(1-r_j\right) & \text{if } \mathrm{X}=\emptyset,\\ \prod_{j=1}^{n}\left(1-r_j\right)\sum_{1\leq i_1\neq\ldots\neq i_n\leq M}\prod_{j=1}^{n}\frac{r_{i_j}p_{i_j}\left(\mathbf{x}_j\right)}{1-r_{i_j}} & \text{if } \mathrm{X}=\left\{\mathbf{x}_1,\ldots,\mathbf{x}_n\right\};n>0,\end{cases} \tag{2.26}$$

where the probability density function for $\mathrm{X}=\emptyset$ corresponds to the probability that none of the M objects exists, while the probability for the case $n>0$ sums over all permutations of the vectors $\mathbf{x}_i$ [25]. The cardinality distribution of a multi-Bernoulli RFS is simply:

$$\Pr\left(\left|\mathrm{X}\right|=n\right)=\prod_{j=1}^{n}\left(1-r_j\right)\sum_{1\leq i_1\neq\ldots\neq i_n\leq M}\prod_{j=1}^{n}\frac{r_{i_j}}{1-r_{i_j}}, \tag{2.27}$$

where the mean cardinality is estimated as $\hat{N}_{obj}=\sum_{j=1}^{M}r_j$.

The effective multi-object tracking requires a mechanism for track ranking/extraction as well as association. We augment the state of each object by a label l (part of the feature vector $\mathbf{g}$, recall Section 2.3) which lies in a label space $\mathbb{L}$ of positive integers. This augmentation yields a labeled state, and we use the labeled random finite set whose realizations of the labeled multi-object state X may not contain two or more objects with the same label. To that end a distinct label indicator, $\delta_n\left(\left|\left\{l_1,\ldots,l_n\right\}\right|\right)$, is used to make the discrimination between valid and invalid labeled multi-object states. A labeled Multi-Bernoulli Random Finite Set (LMB RFS) is thus a labeled version of the multi-Bernoulli RFS in which instead of the component indices i we use labels l which indicate unique track labels. The multi-object probability density function of a LMB RFS is defined as:

$$\pi\left(\mathrm{X}\right)=\delta_n\left(\left|\left\{l_1,\ldots,l_n\right\}\right|\right)\prod_{i\in\mathbb{L}}\left(1-r_i\right)\prod_{l=1}^{n}\frac{1_{\mathbb{L}}\left(l\right)r_l p_l\left(\mathbf{x}_j\right)}{1-r_l}, \tag{2.28}$$

where the $1_{\mathrm{X}}\left(\mathrm{Y}\right)$ is the inclusion function defined by:

$$1_{\mathrm{X}}\left(\mathrm{Y}\right)=\begin{cases}1 & \text{if } \mathrm{Y}\subseteq\mathrm{X},\\ 0 & \text{otherwise.}\end{cases} \tag{2.29}$$

Using random finite set statistics, the single-object Bayes filter may be rigorously extended to multi-object filtering. Because the random variable in the multi-object

Bayes filter is a random finite set, the estimated number of objects naturally includes the state uncertainty of the individual objects. Using the Markov assumption, the entire information about the multi-object state at a time t is captured by the multi-object posterior density $\pi\left(\mathrm{X}_t\right)$. The prediction of the multi-object posterior at the time of the next measurement $t+1$ is obtained using the Chapman-Kolmogorov equation and the multi-object Markov density, while the update at time $t+1$ is done using the multi-target likelihood function $p(\mathrm{Z}_t|,\mathrm{X}_t)$ integrating over all possible multi-object X_{t-1}:

$$\begin{aligned} &\text{prediction:} \quad \overline{\pi}(\mathrm{X}_t) = \textstyle\int p(\mathrm{X}_t|\mathrm{X}_{t-1})\pi(\mathrm{X}_{t-1}|\mathrm{Z}_{1:t-1})d\mathrm{X}_{t-1}, \\ &\text{update:} \qquad\qquad \pi(\mathrm{X}_t|\mathrm{Z}_{1:t}) = \eta p(\mathrm{Z}_t|,\mathrm{X}_t)\overline{\pi}(\mathrm{X}_t). \end{aligned}$$

This equation has the same form as the single-object Bayes filter prediction, but the integral is a set integral and the random variables are random finite sets rather than random vectors. The multi-object Bayes filter, like the single-object Bayes filter, may be implemented utilizing sequential Monte Carlo (SMC) approaches. In practice, most MOT using this RFS approach use the Reversible Jump Markov Chain Monte Carlo (RJ-MCMC) particle filter implementation or some derivation of it. The particle filter in such algorithms uses one set of "colored" particles which cover the joint state space and whose "color" or identity switches from one object to another depending on the observational evidence. However, most algorithms deploy simplification strategies that put bounds on the state transition $p(\mathrm{X}_t|\mathrm{X}_{t-1})$ as well as the sensor model $p(\mathrm{Z}_t|,\mathrm{X}_t)$. This means that even though the particle filter operates over a joint state space, it uses models which only predict and update in local regions of this joint state space (usually limited to small neighborhoods around each element in X_t. Depending on the level of interaction between the individual elements in X, the association between individual elements in $p(\mathrm{Z}_t|\mathrm{X}_t)$ and how much the assumptions break these rules, joint state trackers can potentially lead to highest tracking accuracy. However, to date most such approaches are either computationally intractable or their accuracy breaks down when assumptions in the models are scaled down to allow for real-time execution. These SMC implementation of the multi-object Bayes filter are also not readily applied to real sensor data due to the said computational complexity of the multi-object likelihood function.

To overcome these issues and enable the real-time tracking of multiple objects, our approach relaxes the LMB RFS model by treating each state vector as conditionally independent, where the position and existence of objects are usually estimated separately for each object. This widely used assumption in multi-object tracking applications allows for the conditional probability densities to be more easily modeled. For some road users, such as vehicles this is a sound line of reasoning. However, when tracking smaller objects in a confined space, as seen in [27], we cannot apply this assumption as their position on the ground plane can overlap, while their motion is largely defined by close object-object interactions. Khan et al. [27] propose a RJ-MCMC particle filter using *localized* detection mediated proposal function. In practice this means that each hypothesis is updated using only the sensor evidence

which falls withing a small region around it. At the same time, the motion of each hypothesis is governed only by its own position in the state space and completely independent of other objects.

Due to the hard time restrictions in on-line tracking, the proposed tracker is designed using the independent/local principle. We decouple the estimation of the existence and number of hypotheses N_{obj} from the estimation of the position of the tracked objects. We assume non-interacting state models for all road user categories as well as local measurement models that are confined to a small gating area $\Omega(\mathbf{x}, \mathbf{g})$. The shape of the gating area is specific to each sensor, for example, for a camera detection (due to the loss of distance) it represents a function along a line defined by the azimuth of the detected bounding box. Thus, the proposed tracker performs independent track prediction for each object by applying a non-interacting state transition. Using this formulation it is still possible to compute the belief in the joint state space $\overline{\pi}(X)$ if we simply the sum the predicted beliefs of all individual objects N_{obj}:

$$\overline{\pi}(\mathrm{X}) = \sum_{i=1}^{N_{obj}} p(\mathbf{x}_i). \tag{2.30}$$

At each time step t, the proposed tracker computes the optimal association between individual detections $\mathbf{z}_{j,t}$ to hypothesis $\mathbf{x}_{i,t}$ within the respective gating radius. The optimal association between individual detections and hypotheses is achieved by optimizing an association cost matrix $\left[\mathrm{d}\left(\mathbf{z}_{j,t}, \mathbf{x}_{i,t}\right)\right]_{N_{det} \times N_{obj}}$ using the Hungarian algorithm [28].

Only one assigned detection $\mathbf{z}_{j,t}$ can be used for updating the state of only one track $\mathbf{x}_{i,t}$ using a localized measurement model $p_{Z|H,X}(\mathbf{z}_{j,t}|\mathbf{x}_{i,t})$. Generally, a detection can also be assigned to multiple tracks, which makes sense in situations of occlusion and ambiguity, and we will use this notion to propose improvements in Section 6.7. Detections that do not match to any hypothesis are delegated to the track manager and used to create new hypotheses, consequently update the cardinality number N_{obj}. Notably, for the class person, object interaction can still occur and cause the formation of groups. In this edge case the group size and motion are largely coherent and influenced by the mean behavior. Therefore, the conditional independence assumption does not apply and the proposed MOT would achieve sub-optimal accuracy. To combat this issue, we delegate the object-to-object interaction modeling to the track manager, details in Section 6.8, which merges tracks that become too close to each other.

2.8 Generating hypotheses and associating detections

In principle, a generic approach would be to solve the detection problem for every possible combination of ($\mathbf{x}$, $\mathbf{g}$). To reduce the number of computations, we rather restrict hypothesis checking to specific combinations of ($\mathbf{x}$, $\mathbf{g}$), which we call candidates. Candidates are generated in multiple-steps: first, knowing that road users cannot dis-

appear without trace all locations with a high prior probability of presence at time t are candidates. According to the Markov assumption, these prior probabilities equal the posterior probabilities of presence resulting from the computations at time $t-1$. So in essence, we consider the last known positions of all tracked road users as candidates. In case we distinguish between weakly tracked road users (we are not sure yet that these tracks are real) and strongly tracked ones, we have the option of also including some weakly tracked road users.

Then, if one of our sensors is significantly more accurate than the rest e.g., a lidar, most road users will be detected by this sensor if we set detection thresholds low enough. The locations of confidently detected road users by an accurate sensor are strong indicators of the true position of objects in the scene. We use such detections to generate additional hypothesis combination ($\mathbf{x}$, $\mathbf{g}$). Therefore, such detections are also added as candidates. When using lidar/radar detections as hypothesis seeds the process is straight forward since they produce sensor coordinates $\mathbf{u}$, which are unambiguously (within a rigid transformation due to ego-motion) related to world coordinates: $\mathbf{x} = \mathbf{u}$. However, when we use camera detections to seed hypotheses there is a complication due to the loss of depth in the image formation process. Since we don't know the distance of a camera detection, $\mathbf{u}$ relates to a stripe along a single azimuth of the hypothesis space. Based on the shape of the image bounding box $\mathbf{s}$, we use statistical models of the average object height to spawn more hypotheses at the most likely ranges along the azimuth stripe and less so elsewhere. This issue will be addressed in Chapter 4 where we propose to use additional depth images to accurately range camera detections, which in turn greatly reduces the hypothesis search space.

Finally, we add road users detected by at least two sensors that can jointly estimate the coordinates $\mathbf{x}$. These sensors could be two cameras in different locations, or a camera and a radar. Again the detections must be above some minimal detection threshold, but it is essential to not exclude candidates, so thresholds must be low. Matching between sensors is performed in the domain which has the least positional ambiguity, usually the image plane, and will be explained in detail in Chapter 4.

2.9 Conclusion

This chapter explained the general framework and concepts that will be used to build the proposed environmental perception system. Each of the following chapters expands on these principles, by providing novel approaches that lead to improved accuracy, improved robustness or both. In the next chapter we start with the problem of estimating the ego-motion of the vehicle which is crucial when matching local sensor observations to hypotheses which in section 2.5 we defined in global coordinates. The proposed novel ego-motion estimation method was published in the proceedings of an international conference [9] and its implementation is used throughout this thesis.

Then, in Chapter 4 we will explain how to range camera detections using several novel techniques for reconstructing dense depth maps. These ranged camera detections provide are very useful in reducing the hypothesis search space which reduces

the algorithmic complexity of detection and tracking. The novel depth completion methods resulted in research papers which were published in the proceedings of three international conferences [10, 11, 13].

Furthermore, we propose a novel algorithm for cooperative fusion of detections between camera, lidar and radar. The principle of cooperative fusion applies information feedback loops between sensors, practically modifying the activation function for weak detection evidence of one sensor using confident detections from the other sensors. The cooperative fusion system remains within the bounds of the Bayesian framework, transparently computing the log-likelihood ratio, Eq. (2.12), with or without using the sensor-sensor feedback loop. However, the inclusion of the cooperative feedback allows for improved object detection in border cases of poor viewing conditions. This work was published as research papers in the proceedings of four international conferences [12, 14, 16, 17].

Finally, Chapter 6 provides a detailed analysis on how to model the temporal dimension, i.e. track detections over time in order to improve the belief in their existence as well as minimize the uncertainty in their location. We base our design choices on the fact that the system will be deployed in a real-world traffic environment, faced with various road users, imperfect sensors and changing weather conditions. There is also the distinct possibility that the system may be faced with cyber attacks, vandalism or other unexplained phenomena which can cause a complete sensor failure. Our tracker has the flexibility to adapt its sensor model parameters and switch to using low-level sensor cues to continue updating its hypotheses even in the complete absence of detections. The novelties proposed in this chapter were published as two open-access journal articles [7, 8].

3 Ego-localization

3.1 Introduction

One of the essential elements in a perception system is the ability to perceive static and moving objects while the vehicle itself is also in motion. We are interested in estimating the evolution of the location and orientation (pose) of other road users, relative to a fixed, starting world coordinate, for example, the last known Geo-location obtained when the system had access to satellite-based positioning. Any well performing environmental perception system must also incorporate its own pose information in order to detect, track and predict the relative position of other objects in environment. Errors in estimating the evolution of its own pose (ego-motion) will accumulate and make the prediction about other objects position more uncertain, ultimately resulting in reduced autonomous driving efficiency. Therefore, the odometry needs to be accurate consistently through the time window within object detection, tracking and prediction is performed.

A naive object tracking implementation applies a motion model (in our probabilistic framework: $p_{X_t|H,X_{t-1}}(\mathbf{x}_t|H_1, \mathbf{x}_{t-1})$) to every tracked object from the moment $t-1$ and matches it to the closest detection $\mathbf{z}_{l,t}$, seen in the present. Without taking into consideration the motion of the vehicle itself, such techniques fail to accurately predict the position of detections when the ego-motion is large or the frame-rate is low. By also applying an appropriate motion model to the position of the vehicle, we can achieve accurate object detection and tracking which is crucial for autonomous driving. This chapter deals with the problem of modeling the own motion of the vehicle by estimating its change in position and orientation over time, referred to as its pose.

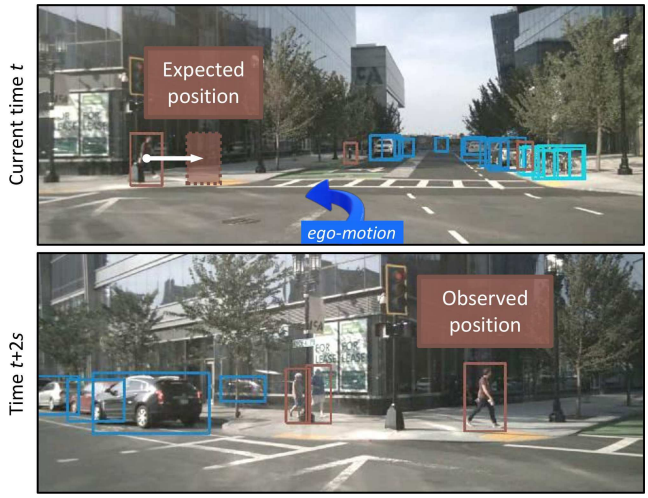


Figure 3.1: Two camera frames captured 2 seconds apart (nuScenes scene-0518). Due to ego-motion, the expected content based on motion prediction in the present differ significantly from what is actually observed.

To illustrate the concept, consider the scene shown in Figure 3.1 captured by a camera on the roof of a moving vehicle. As the vehicle is turning to the left, so too is the camera which creates a shift in the apparent position of other road users in the captured image. This creates a significant challenge for tracking because matching observations over time needs to not only estimate the motion of other road users, but also factor-in the pose change of the camera and its speed. Without knowing the ego-motion, the expected position of the pedestrian does not match with the observed content resulting in lost tracking and the spurious creation of new tracks which refer to the same object in reality. Thus, having an accurate estimate of the ego-motion over short time intervals is a safety critical task that cannot be underestimated.

Satellite-based navigation, a fairly common component in current vehicles, provides basic positional information by estimating the geographic latitude and longitude from satellite signals and motion sensors like wheel and steering encoders. Depending on the satellite reception quality, most current systems provide real-time positional information within a couple of meters of accuracy, which is not accurate enough for the safety critical object detection and tracking in autonomous driving. Matching the position of detected objects over subsequent sensor observations, especially small, fast-moving vulnerable road users, requires a positional accuracy in the range of several centimeters which currently can only be achieved by processing sensor observations captured by the vehicle itself.

Camera-based ego-motion estimation, called visual odometry, computes the relative motion of image features in subsequent camera frames and correlates this motion to the physical motion of the vehicle. Visual odometry combined with offline maps

has been shown [29] to provide localization that is accurate enough for autonomous driving under ideal viewing conditions. However, the loss of visual features in camera images under poor lighting conditions creates large, unpredictable localization errors in visual odometry. Alternatively, ego-localization by means of active depth sensors such as 3-D lidars has also been studied [30] and shown to produce the accuracy needed under most conditions. 3-D lidar is a multi-beam, active-light range measuring device which usually operates in the invisible infra-red part of the spectrum. A typical 3-D lidar, shown as a cross section in Figure 3.2, directly measures the 3-D scene geometry and is unaffected by ambient light levels. Such active light devices can be very effective in perceiving features in the environment which can be used to compute the vehicle ego-motion. Ego-motion can thus be computed through the apparent shift between 3-D scene features between several capture intervals. Such an estimate is independent of GPS signal or ambient light and can be reliably used in perception applications. However, the matching of lidar features can become ambiguous when the input data is sparse or polluted with outliers, requiring the use of voxelization or other sorts of geometric-primitive modeling to retain its robustness.

At first glance, odometry for autonomous vehicles might seem like a mature topic with many scientific contributions advancing the state-of-the-art by only a marginal amount. However, most of the published research is designed and tested on synthetic or small-scale datasets which do not contain the important difficult edge cases. For illustration, a difficult and often referenced problem is the loss of GPS reception in tunnels and city centers. Odometry systems in this situation switch to relying heavily on on-board sensors such as inertial navigation or visual cues and wheel and steering angle readings. Yet, this seemingly simple solution has its own downsides due to the decay of accuracy over time (drift). Wheel speed accuracy decreases as tires wear down or change their pressure due to temperature variations. Moreover, difficult to estimate mechanical effects, such as wheel slip or steering dead-zone cannot guarantee a centimeter-level positional accuracy. Visual cues from the camera, on the other hand, become difficult to match in low light or glare.

This chapter explains a localization technique that is both centimeter-accurate and robust under ideal as well as compromised viewing conditions. Assuming that the vehicle is moving on a locally flat environment, the proposed method uses the sensor features to compute a two-dimensional map of the environment which it then compares to the map from the past using phase-only correlation. Shifts and rotations in two maps correspond to physical translation and change in orientation of the vehicle. The two-dimensional map representation efficiently stores the local occupancy information and can be computed from any available sensor observation such as camera detections, radar targets or lidar point clouds. For simplicity, we will show how to build the proposed odometry system using lidar-only observations which is experimentally shown to provide centimeter-accurate localization while being robust to various light and weather conditions and simple enough to run in real-time.

Finally, the proposed method was designed to be simple enough to be deployed in a real-time system and to be able to operate predictably in the presence of signal degradation. As we will see in the literature study, this combination of requirements

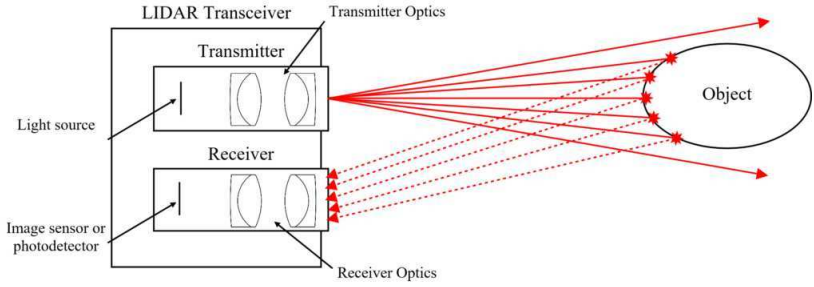


Figure 3.2: Cross section diagram showing the principle of operation of a multi-beam lidar. This sensor uses pulsed infrared light and mechanically rotated transceiver (a light source and a photo detector) to perceive objects on a 360° field of view.

poses a challenge for camera and lidar odometry and has motivated us to seek a solution beyond classical techniques based on GPS, visible light cameras or point cloud matching. This chapter describes an important contribution to lidar-based odometry research field, proposing a simple, yet accurate and robust odometry solution.

3.2 Literature overview

Odometry methods based on perception sensors can be divided into two broad categories: methods which estimate the full 6 degrees of freedom (DOF) pose change in 3-D space and methods which specialize in a domain (such as ground vehicles) where the robot exercises fewer degrees of freedom. Methods in the former category compute the full roll-pitch-yaw pose and x-y-z translation, usually requiring more complex data interpretation. Contrarily, methods which estimate only the ground plane motion are restricted to 3 degrees of freedom: x-y translation and yaw: the rotation along the z-axis.

In the context of autonomous driving, perceiving the immediate surroundings of the vehicle are of most interest. This is because the priority of safety critical tasks such as collision avoidance outweighs optimal ride comfort or fuel economy. Since the car and other road users generally move on the road surface, see example on the left image in Figure 3.3, we are interested in modeling their motion along a surface which, with small exceptions, is locally flat. In this context we deem a surface to be locally flat if it can accurately be approximated as piece-wise planar function where the pieces are comparable in size to the extent of the vehicle sensors. All objects, therefore, move in a 2-D coordinate system and exhibit only 3 degrees of freedom (yaw and x-y translation) instead of the 6 degrees of freedom (roll-pitch-yaw and x-y-z translation) in a 3-D system. Assuming that the world is locally-flat, the matching of camera and lidar features can be greatly simplified because features can more easily be traced using their projection onto the local road surface.

This assumption is acceptable as the extensive road study [31] indicates, where

over 12-million road grade measurements in 1728 U.S. counties were analyzed and found that around 80% of distances on highways exhibit less than 2% absolute road grade. Note that discrepancies between the flat-world and the 3-D world model do not occur on roads with constant grade, but rather when the road grade is changing, for example: when the vehicle is at the foot of a hill such as the one shown on the right image in Figure 3.3. In this scenario, features detected on the hill sit much higher above the ground plane coupled to the vehicle and need to be projected on the ground plane at a distance proportional to the sine of the road grade. It is not difficult to imagine that applying a flat-world motion model will largely underestimate the distance and velocity of road users moving on such slopes. Such situations are only encountered at points of extreme change of road gradient and are rare, border cases when the odometry and perception systems need to be specially tuned or the ground surface be re-modeled using smaller planar pieces. For the remainder of this thesis we will assume that the world is locally flat, modeling a stretch of road as piece-wise linear function. Consequently, odometry on a 2-D plane can be done by only estimating the yaw rate and x-y translation which reduces the solution space and allows for more robust solutions using simpler data representations.

Mainstream of vehicle odometry approaches using cameras and lidars require key feature detection and matching. This can be assessed from the submission details in the KITTI odometry benchmark [20, 32]. The spatial alignment of detected features (referred to as registration or matching) is often done using a variation of the Iterative Closest Point (ICP) approach [33]. When using only camera sensors (visual odometry), features can be accurately detected and matched across the image plane. However, due to the loss of depth information during camera projection, the 2-D motion of features across the image plane is difficult to accurately relate to the physical 3-D motion of the camera. These approaches have the downside that they depend on the existence of distinctive features. For example, matching features between two images taken on the ocean or in the dessert is very difficult due to the repeating content. Additionally, due to the loss of depth during image formation, camera-based odometry can only be accurate up to a scaling factor which is closely related to the intrinsic camera parameters.

When using stereo cameras and/or lidar, many approaches have been proposed to register range images, [34], and then use features extracted from these images to register consecutive scans. Other methods try to extract geometric primitives from within the point clouds, such as planes and edges, and then use those for matching and registration [35]. Although the reported results are promising, these approaches have their problems in situations where the environment does not contain simple planes and edges (open roads, forests, parks). There most important issue with feature based registration approaches lies in the complexity of extracting features and their robust registration which forces the algorithms to use layered system with feedback loops such as local bundle adjustments. Such systems can be highly sensitive to outlier noise which deteriorates their performance whenever a feature cannot be traced through the sequence of data.

Approaches such as [36] combine RGB pixel information with depth measure-

Figure 3.3: Examples of highway engineering in the United States. Left: A completely flat avenue in Manhattan, New York; right: Lombard Street in San Francisco, California, built on a hill with a 27% grade.

ments from a time-of-flight camera to integrate the depth data into a 3D model using the ICP approach. If accurate depth information in the form of a depth image is available, then matching of such "image" features can be performed even purely using the depth images alone. Depth data from a 3-D lidar consists of geometric measurements of surfaces and provides an accurate representation of the 3-D structure. However, directly matching between 3-D points from real 3-D lidar sensors is not trivial as these point clouds tend to be anisotropic and sparse. Many authors ignore the fact that lidar data is intrinsically captured in a fixed scanning pattern and is therefore spatially correlated. By discarding the spatial correlation between neighboring points, valuable information which could have been used to better register the data is being discarded. Instead, authors propose alternative solutions [37] by fusion of data from inertial navigation sensors (INS), global position system (GPS) or wheel rotation sensors. Bayesian filtering techniques such as Kalman or particle filters are often used in order to reinforce the measurements with the past data for more accurate estimation.

To mitigate the aforementioned issues, this chapter proposes an accurate 3-DOF odometry method that uses a sensor agnostic data representation applicable in any camera/lidar/radar sensor configuration. Instead of processing the heterogeneous sensor measurements such as lidar depth, camera features or radar targets, we convert them into the concept of *occupancy*, a binary random variable $\mathrm{o}\left(\mathbf{x}\right)$ describing whether a specific position $\mathbf{x}$ of space is occupied or not. This analysis follows closely the theory in the exposition Section 2.4. However, for the task of odometry we will use a more broad occupancy definition i.e., $\mathrm{o}\left(\mathbf{x}\right)$ returns occupied value if there is *any* object standing within a prespecified to $\mathbf{x}$.

Occupied area, therefore, can contain any object which presents a collision risk to the vehicle such as buildings, greenery, road infrastructure or other road users. Conversely, area that is not occupied represents drivable road surface. An occupancy grid

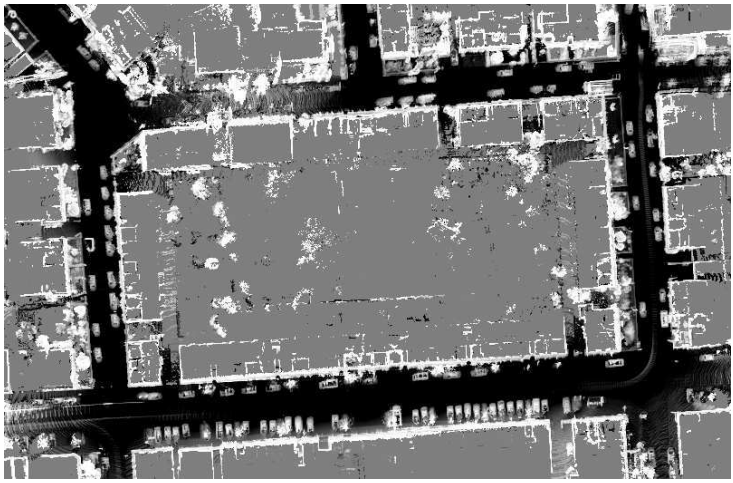


Figure 3.4: Small section of a reconstructed occupancy grid map with cell size $12.5 \times 12.5cm$ (KITTI dataset). Black and white colors indicate free and occupied space respectively, while gray is the prior probability of unobserved area.

map, in this respect, represents the external environment through concepts relevant for autonomous driving. The proposed odometry method uses this representation to aggregate new sensor measurements and simultaneously align them with what was seen in the past, while updating the occupancy of already seen areas with the new observations. The process is commonly refereed to as Simultaneous Localization and Mapping (SLAM).

In the literature, authors make use probabilistic maps, first proposed by [38] in 1985. Besides mapping, the occupancy data can also be used for various other key functions necessary for the mobile vehicle navigation, such as positioning, path planning, collision avoidance object detection and prediction of the future state of the environment. Other authors have also suggested that occupancy grid maps are the most successful environment representation in mobile robotics [39]. Moreover, in the domain of autonomous vehicles, occupancy maps provide a very efficient way of compressing sensor data when recording a background model of vast environments, Figure 3.4. For example, range data from laser or radar sensors can be fused with object detector outputs and 3-D points from stereo cameras within the same probabilistic model. By assuming that the world around the vehicle is locally flat and that it can be precisely modeled as a two dimensional map, the localization can be simplified into an image registration problem which can be solved more robustly than a full 3-D registration. The technique is not without downsides though, representing the environment as an image/map requires us to discretize the space which puts theoretical limits on the registration accuracy. Moreover, maps are poor container for sparse measurements requiring large amounts of memory to be wasted modeling the unobserved space. Thus, the more dense and redundant the sensor data is (think of a

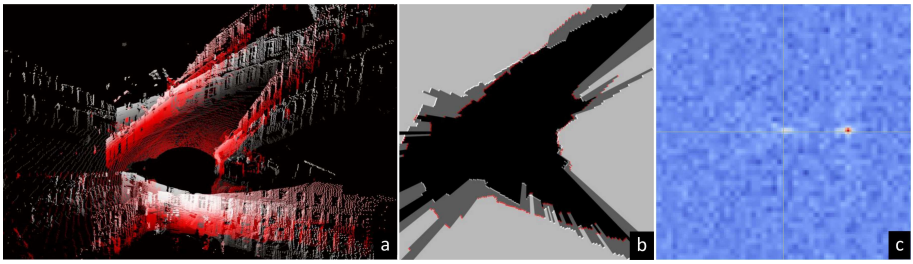


Figure 3.5: Example data from a vehicle taking a right turn. The three images illustrate the data flow in the proposed odometry method: a - overlay of two 3-D lidar point clouds (red-current, white-old); b - overlay of the two occupancy grid representations of the respective point clouds (red-occupied cells in the present, white-occupied cells in the past); c - the computed phase correlation.

wall scanned vertically at regular intervals of 2cm), the more sense it makes to use a 2-D map instead of registering the original sensor data. However, if the sensor data is sparse but unambiguous, other techniques such as ICP offer better performance. In the following analysis we rely on relatively dense data captured by a rotating 3-D lidar sensor which effected the choice to do the registration using a 2-D map representation. An efficient implementation of these maps has been proposed by [40], which will be further explained.

3.3 Overview of the proposed method

The proposed ego-localization method estimates the motion of a land robot which moves on a relatively flat surface. The solution assumes a locally flat world which reduces the 6-DOF solution space to 3-DOF by discarding the pitch and roll as well as vertical translation. In this context the ground surface is considered locally flat if the region seen by the on-board sensors can accurately be approximated by a single plane. The method uses a sensor-agnostic representation, a 2-D occupancy grid map, which transforms observations into occupancy of the environment into small grid cells. In this context of autonomous driving, a grid cell is considered occupied if the sensors observe any obstacle for the vehicle in that area. All detected obstacles are orthographically projected onto the ground plane and converted into occupancy using an appropriate sensor model. When aggregating measurements from multiple sensors such as lidar range data, radar targets or camera detections, the occupancy map integrates these heterogeneous sensor observations into a single image-like container which is then easy to interpret.

We use the notion that sensor measurements from a moving robot map into occupancy content which appears to move opposite to that of the robot's own motion. This is because the physical translation and rotation of the robot causes translation and rotation changes to the sensor coordinate system over time. If the robot turns to the

right, see example a) in Figure 3.5, the 2-D occupancy content appears to be turning to the left. Similarly, if the robot moves forward, the computed occupancy content seems to move to the back. Based on this observation, it's clear that the motion of the robot is proportional to the shift and rotation in computed occupancy maps over time, b) in Figure 3.5. The proposed odometry method estimates these two quantities using the correlation-based image registration technique Phase Only Correlation (POC). POC uses the phase information in the Fourier transform of two consecutive occupancy maps, decoupling the estimation of rotation from the estimation of translation in the following steps:

Step 0: At time t, compute the local *occupancy map* from the current sensor observations by projecting them onto the local ground plane and applying the appropriate sensor model. An optional pre-processing step would include first the estimation of the ground plane orientation (which might change during accelerated motion due to the vehicle suspension).

Step 1: Estimate the vehicle's *yaw rate* by measuring the rotation of consecutive occupancy maps (captured at t and $t-1$). This step uses the notion that an occupancy map that rotates in Cartesian space exhibits linear shift in polar space. Thus, the physical rotation of the vehicle is equal to the change in the theta coordinate which is observed as a horizontal shift of the occupancy in polar coordinates. POC computes this shift by correlating the phase information of two occupancy maps in the Fourier transform domain. The Fourier transform preserves the original rotation information and is invariant to the original translation.

Step 2: Estimate the robots *translation* by measuring the residual shift in the occupancy map. The algorithm works by first correcting the current occupancy map using the angle computed in step 1. After the current occupancy map is "unrotated", the only difference between the two maps can mostly be attributed by to translation. This translation is then estimated using the same procedure as in step 1, with the difference that instead of the polar we now use the original x-y representation of the two occupancy maps.

Step 3: Refine the initial POC solution using *sub-pixel peak estimation*. This step is needed because the proposed method is constructed of several discrete algorithms (finite occupancy map and discrete cells, discrete polar transform and discrete Fourier transform). The position of the initial POC estimate (see example c) in Figure 3.5) takes discrete (whole pixel) values which limits the precision of the estimate to the resolution of the map. Therefore, the final x-y translation and yaw rate is computed using sub-pixel peak estimation of the position of the POC estimate.

3.4 Modeling the environment

The proposed method compares current observations to the ones from the past and estimate the egomotion as the inverse of the apparent disparity in the observations. Assuming that vehicles move in a 2-D coordinate system, we chose to model the environment as a map that describes the 2-D structure relevant to the vehicle's motion.

To illustrate this further, imagine a scene with a single vertical obstacle on the ground. Since our vehicle is limited to moving on the ground, the area on the ground under the obstacle is considered occupied regardless of its height i.e., the vehicle cannot fly over or under an obstacle. Withing these restrictions, the 3-D structure of the environment can safely be represented as a 2-D, birds-eye-view map that models which areas are occupied and which ones are free. We refer to such binary maps as occupancy grid maps.

The system considers a grid cell at location $\mathbf{x}$ to be occupied if the sensors observe any object that sits sufficiently higher than the ground plane, so that it causes a potential collision threat to the vehicle. The likelihood of observing an object above an occupied grid cell increases with height above the ground. Each time a sensor measurement is obtained, a new estimation of the orientation of the local ground plane is performed and the data is projected onto this estimated plane to compute the occupancy for the currently perceived environment. Then, this current occupancy is matched and registered onto a global occupancy map which contains all the occupancy information form the past. Formally, an occupancy map represents a binary random field which summarizes the sensor data into occupancy (position and to a degree shape) of all objects on the ground plane. For practical purposes, we define the occupancy map $\mathbf{o}$ as the matrix containing occupancy values $o(\mathbf{x}); \mathbf{x} \in \mathbb{Z}^2$ for cells at regularly sampled positions $\mathbf{x} : (x, y)$ on a 2-D plane.

Occupancy maps can be estimated either using a single sensor scan or estimated from sensor measurements over time, assuming that the pose (location and orientation) of the vehicle. In the first case, due to the limited range of sensors, we are only estimating the occupancy of the local environment, while in the second case, the occupancy of the environment along the complete trajectory of the vehicle over the given time period.

When multiple sensor measurements $\mathbf{z}_{0:t}$ are used to reconstruct the occupancy over the time period 0 to t, we can use a probabilistic approach and maximize the belief in the probability of occupancy for each grid cell at time t given all previous observations: $\mathrm{bel}(o(\mathbf{x}_t)) = p_{O_t|Z_{0:t}}(o(\mathbf{x}_t)|\mathbf{z}_{0:t})$. In the exposition Section 2.3, we defined a measurement $M(t) \triangleq \{(\mathbf{z}_l)\}$ as the list of detections, however, in this analysis we will use the simpler notation $\mathbf{z}_t$ to indicate the vector that aggregates all observations made by the sensor at time t. These might be the center points of all objects detected by lidar regardless of their class label, or for unprocessed lidar data, observations $\mathbf{z}_t$ represent the points of the scanned 3-D point cloud at time t. Each measurement point $\mathbf{z}_t^{[i]} \in \mathbb{R}^3$ being in Cartesian space with origin at the current position of the lidar.

In the process of estimating the ego-motion, the proposed method applies the concept of simultaneous localization and mapping (SLAM) which aligns the current observation to the ones from the past using occupancy maps. We define the ego-motion as a series of rigid transforms defined as pose matrices P_i. Formally, a 3-D pose matrix $P = (R|\mathbf{t})$, consists of a rotation 3-D matrix $R \in \left\{\mathbb{R}^{3\times3} \,\middle|\, R^T R = I\,, |R| = 1\right\}$ and a 3-D translation vector $\mathbf{t} \in \mathbb{R}^3$. It defines the rigid transformation between two coordinate systems, and in our problem, effectively explaining the change in position

and rotation of the vehicle through time. The SLAM procedure thus constitutes the estimation of the current pose matrix P_t by registering the current to the old measurements, $\mathbf{z}_{0:t-1}$ and $\mathbf{z}_t$, while at the same time updating the probability of occupancy $p_{O_t|Z_{0:t},P_{0:t}}\left(o\left(\mathbf{x}_t\right)|\mathbf{z}_{0:t},P_{0:t}\right)$ using the now registered measurements $\mathbf{z}_t$.

Using a 3-D lidar as an example sensor, the measured set of 3-D points at time t (using homogeneous coordinates) can be transformed to the corresponding set sampled at time $t-1$ using the rigid transform defined by P_t:

$$\begin{bmatrix} \mathbf{z}_t^{[i]} \\ 1 \end{bmatrix} = \begin{bmatrix} R & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{z}_{t-1}^{[i]} \\ 1 \end{bmatrix} = \begin{bmatrix} R\mathbf{z}_{t-1}^{[i]} + \mathbf{t} \\ 1 \end{bmatrix}. \tag{3.1}$$

Note that the origin of the coordinate system of $\mathbf{o}$ is defined as the location of the vehicle at $t = 0$ and the initial orientation R_0 is relative to the orientation of the ground plane at $t = 0$. Even if the world is completely flat, the orientation of the vehicle relative to the ground plane can change due to its suspension. This becomes important for computing $bel\left(o\left(\mathbf{x}_t\right)\right)$ from the current observations because small changes of the orientation of the observed ground plane can cause the map to become inaccurate and distorted. The problem becomes even more pronounced when the vehicle is accelerating, braking or driving over bumps and potholes, see Figure 3.6.

At each time step t, the proposed algorithm first estimates the orientation of the ground plane which will become the plane for mapping the occupancy. In the context of autonomous driving, this plane can be assumed to coincide with the road surface around the vehicle. Since the vehicle is driving on the ground plane, we expect that the overwhelming majority of observed points belong to the road and use this notion to find the plane which fits most of the points in $\mathbf{z}_t$. For the sake of brevity, we chose to use the RANdom SAmple Consesnsus (RANSAC) which iteratively selects a random sub-sample of points to generate a plane equation for which the average distance of all other points is computed and the subset with the lowest average distance (highest number of inliers) is selected. From the list of inliers of this optimal subset, a new plane is fitted in a least squares sense. The orientation of the estimated local ground plane is defined by its normal vector $\boldsymbol{n} = (a, b, c)$, which we can use to compute the Direction Cosine Matrix i.e., the 3x3 rotation matrix of what transforms the captured point cloud from its original reference frame to the reference frame of the estimated ground plane:

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}, \tag{3.2}$$

where the element in the i^{th} row and j^{th} column represents the cosine of the angle between the i-axis of the reference frame $\boldsymbol{n}_{ref} = (0, 0, 1)$, and the j-axis of the ground plane.

As proposed by [22], the process of occupancy mapping can be decomposed into many one-dimensional estimation problems, which are solved independently of each other. In most practical cases, a static-world assumption can be safely made i.e., the world structure does not change over time. Moreover, occupancy maps make an even

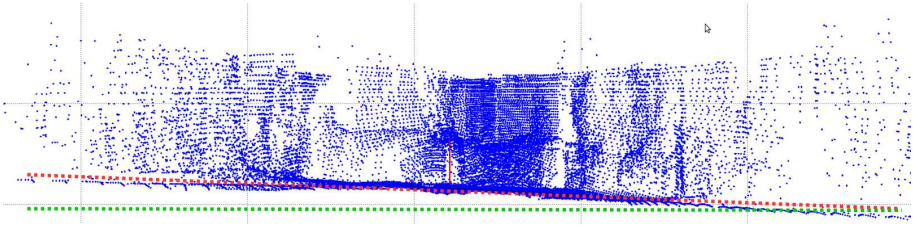


Figure 3.6: A side-view of a tilted lidar point cloud (blue) captured during accelerated motion. The green dashed line coincides with the plane z=0, while the true ground plane lays close to the plane estimated by the proposed method, shown with a red dashed line.

stronger assumption of local conditional independence given knowledge of each individual grid cell regardless of the occupancy of neighboring cells. Depending on the type of sensor (camera/radar/lidar) and the relative size of grid cells, assuming local conditional independence can be incorrect because a single sensor reading will span over multiple grid cells. However, it allows us to decompose the map estimation problem into multiple, independent local estimation problems. These local estimates correspond to the probability of occupancy for the individual cells $o(\mathbf{x}_t)$ at time t: $p_{O_t|Z_{1:t},P_{1:t}}(o(\mathbf{x}_t)|\mathbf{z}_{1:t}, P_{1:t})$. In order to avoid series of multiplication of small numbers (which lead to rounding errors over time), it is common to compute the log-odds, replacing multiplications with addition operations:

$$l(\mathbf{x}_t) = \ln \frac{p_{O_t|Z_{1:t},P_{1:t}}(o(\mathbf{x}_t)|\mathbf{z}_{1:t}, P_{1:t})}{1 - p_{O_t|Z_{1:t},P_{1:t}}(o(\mathbf{x}_t)|\mathbf{z}_{1:t}, P_{1:t})}, \tag{3.3}$$

where the posterior probability of occupancy can be recovered from the log-odds $l_{x,y}$ through:

$$p_{O_t|Z_{1:t},P_{1:t}}(o(\mathbf{x}_t)|\mathbf{z}_{1:t}, P_{1:t}) = \frac{1}{1 + \exp(-l(\mathbf{x}_t))}. \tag{3.4}$$

The log-odds in Eq. (3.3) can be estimated recursively by applying the Bayes rule to the posterior $p_{O_t|Z_{1:t},P_{1:t}}(o(\mathbf{x}_t)|\mathbf{z}_{1:t}, P_{1:t})$. For notational simplicity, the following analysis assumes that the pose matrix P_t is known and is contained in the observation vector $\mathbf{z}_t$. With these assumptions we can decompose the posterior probability of occupancy into:

$$p_{O_t|Z_{1:t}}(o(\mathbf{x}_t)|\mathbf{z}_{1:t}) = \frac{p_{Z_t|Z_{1:t-1},O_t}(\mathbf{z}_t|\mathbf{z}_{1:t-1}, o(\mathbf{x}_t))\, p_{O_t|Z_{1:t-1}}(o(\mathbf{x}_t)|\mathbf{z}_{1:t-1})}{p_{Z_t|Z_{1:t-1}}(\mathbf{z}_t|\mathbf{z}_{1:t-1})}. \tag{3.5}$$

In order to reach a practical solution for Eq. (3.3) we will make the assumption that the world is static, meaning that the map is not changing over time. This allows us to consider all past sensor observations to be independent given knowledge of the map $\mathbf{o}_t$ for any point in time t, and more so, since we assumed the conditional independence

of individual grid cells $o(\mathbf{x})$:

$$p(\mathbf{z}_t \,|\mathbf{z}_{1:t-1}, o(\mathbf{x}_t)) = p(\mathbf{z}_t \,|o(\mathbf{x}_t)). \tag{3.6}$$

Applying this assumption the posterior in Eq. (3.5) becomes:

$$p(o(\mathbf{x}_t) \,|\mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_t \,|o(\mathbf{x}_t)) \, p(o(\mathbf{x}_t) \,|\mathbf{z}_{1:t-1})}{p(\mathbf{z}_t \,|\mathbf{z}_{1:t-1})}, \tag{3.7}$$

which can further be simplified by applying the Bayes rule to the term $p(\mathbf{z}_t \,|o(\mathbf{x}_t))$:

$$p(o(\mathbf{x}_t) \,|\mathbf{z}_{1:t}) = \frac{p(o(\mathbf{x}_t) \,|\mathbf{z}_t) \, p(\mathbf{z}_t) \, p(o(\mathbf{x}_t) \,|\mathbf{z}_{1:t-1})}{p(o(\mathbf{x}_t)) \, p(\mathbf{z}_t \,|\mathbf{z}_{1:t-1})}, \tag{3.8}$$

where $p(o(\mathbf{x}_t))$ is the occupancy prior and $p(\mathbf{z}_t)$ is generally equal for all observations. To complete the log-odds in Eq. (3.3), the probability of the grid cell to be free $p\left(\overline{o(\mathbf{x}_t)}\right)$ is computed by analogy:

$$p\left(\overline{o(\mathbf{x}_t)} \,|\mathbf{z}_{1:t}\right) = \frac{p\left(\overline{o(\mathbf{x}_t)} \,|\mathbf{z}_t\right) p(\mathbf{z}_t) \, p\left(\overline{o(\mathbf{x}_t)} \,|\mathbf{z}_{1:t-1}\right)}{p\left(\overline{o(\mathbf{x}_t)}\right) p(\mathbf{z}_t \,|\mathbf{z}_{1:t-1})}, \tag{3.9}$$

and plugged in Eq. (3.3) to eliminate several hard to compute terms:

$$l(\mathbf{x}_t) = \ln \frac{p(o(\mathbf{x}_t) \,|\mathbf{z}_t)}{1 - p(o(\mathbf{x}_t) \,|\mathbf{z}_t)} + \ln \frac{1 - p(o(\mathbf{x}))}{p(o(\mathbf{x}))} + \ln \frac{p(o(\mathbf{x}_t) \,|\mathbf{z}_{t-1})}{1 - p(o(\mathbf{x}_t) \,|\mathbf{z}_{t-1})}. \tag{3.10}$$

By substituting the previous log-odds $l(\mathbf{x}_{t-1}) = \ln \frac{p(o(\mathbf{x}_t)|\mathbf{z}_{t-1})}{1-p(o(\mathbf{x}_t)|\mathbf{z}_{t-1})}$ into Eq. (3.10), we arrive at a recursive equation:

$$l(\mathbf{x}_t) = \ln \frac{p(o(\mathbf{x}_t) \,|\mathbf{z}_t)}{1 - p(o(\mathbf{x}_t) \,|\mathbf{z}_t)} + \ln \frac{1 - p(o(\mathbf{x}))}{p(o(\mathbf{x}))} + l(\mathbf{x}_{t-1}), \tag{3.11}$$

where the initial values can be constructed from prior probabilities without observing any data:

$$l(\mathbf{x}_0) = \ln \frac{p(o(\mathbf{x}))}{1 - p(o(\mathbf{x}))}. \tag{3.12}$$

In order to implement the occupancy map model in practice, we need a way of evaluating the sensor model $p_{O|Z}(o(\mathbf{x}) \,|\mathbf{z})$. We hereby present a model based on lidar, but the similar analysis can be applied to radar, ultrasonic range sensor and, to degree, a camera object detector. The measured distance by lidar depends on a variety of properties, such as the surface material of the object and the angle between the surface normal and the beam. Using known target positions we analyzed the errors in measured range and azimuth for the Velodyne VLP-16 Puck 3-D lidar and

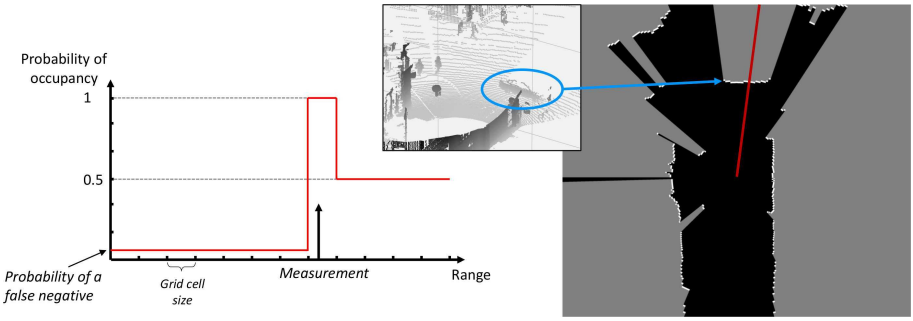


Figure 3.7: Example of a 1-D lidar sensor model (left) and the computed probability of occupancy map (right; brighter is more probable). The red line overlaid in the occupancy map shows the 1-D occupancy model applied to a slice of the lidar scan from the car (blue ellipse).

concluded that both the range and azimuth measurement errors are accurately modeled by a Gaussian distribution with a variance of $3cm$ and $0.35°$ respectively. When we use accurate sensor such as the Velodyne lidar, depending on the grid cell size, $p_{O|Z}\left(o\left(\mathbf{x}\right)|\mathbf{z}\right)$ can be take form as the piece-wise linear function shown in Figure 3.7. Any lidar measurement higher than a collision threshold (for example $15cm$) will cause cells in front of the point to have near-zero probability of occupancy, the cell which contains the measurement to have near-one probability of occupancy, while cells behind the measurement are unobservable and modeled with a constant probability of occupancy, in the literature it is common to use a non-informative value of 0.5. The example in Figure 3.7 demonstrates the missing lidar measurements behind the car (indicated with a blue ellipse) and the resulting uninformative occupancy map region in the shadow of the occupied cells (indicated with blue arrow).

Finally, it is worthy to mention that there exist an alternative approach for computing occupancy maps by using a forward sensor model which computes the likelihood of the sensor measurements in the space of all possible maps. This approach maximizes the probability that the given measurement is observed over the entire state space of all possible map configurations. However, when dealing with point cloud data from lidar the forward model formulation is not suitable for real-time operation since finding the optimal map configuration depends on all sensor measurements from the past. Methods such as [34] propose simplifications using surface patches instead the raw lidar points, yet a single optimization step requires around 5 seconds to compute. Further information about the implementation of such forward models can be found in [22].

3.5 Ego-localization by registration of occupancy maps

As the vehicle is moving through the environment, it experiences rotational and translational changes to its pose and location. This change in pose is best illustrated

through the change in Euler angles i.e., the angular velocities$(\omega_x, \omega_y, \omega_z)$ and the change in location through the offsets$(\triangle x, \triangle y, \triangle z)$ between two time instants. We refer to rotation around the x-axis (pointing to the right of the vehicle) as pitch, rotation around the y-axis (pointing to the front of the vehicle) as roll and rotation around the z-axis (pointing up) as yaw. When the vehicle is turning this motion is causing change in yaw $|\omega_z| > 0$. Similarly, when the vehicle is entering a ramp with change in the incline (grade) the motion results in change in pitch $|\omega_x| > 0$. Lastly, a change of road camber causes change in roll $|\omega_y| > 0$.

Ego-localization constitutes the estimation of the angular and positional changes of the vehicle as it is moving through the environment. The estimation of these changes can sometimes be complicated by the vehicle suspension which resists sudden forces applied to the vehicle's body trying to dampen sudden changes in velocity and keep the vehicle level to the ground. As we discussed in Section 3.2, it can be generally expected that the vehicle is moving in a relatively flat environment which makes the actual change in roll, pitch and elevation between two consecutive laser scans negligible $(\omega_x \approx 0; \omega_y \approx 0; \triangle z \approx 0)$.

Assuming that the vehicle is moving through a mostly static environment, the apparent change in the content of consecutive sensor measurements can be entirely attributed to the ego-motion. Any measured point or feature in the present is thus related to an appropriate point or feature measured in the past through the augmented pose matrix P_t. Estimating the pose then amounts to finding the augmented matrix that minimizes the distance between the two sets of 3-D points after transformation:

$$P_t \approx \underset{R,\mathbf{t}}{\operatorname{argmin}} \sum_i d\left(R\mathbf{z}_{t-1}^{[i]} + \mathbf{t}, \mathbf{z}_t^{[i]}\right), \tag{3.13}$$

using the point to point distance function d. The solution of the ego motion is a typical non-linear least squares problem whose solutions are often highly sensitive to outliers.

A widely used technique to solve point cloud registration for odometry is the Iterative Closest Point algorithm [41] which is iteratively searching for the nearest neighbors for each point. Although effective, ICP is sensitive to errors in the correct matching of points between input point clouds. Faulty matches cause large error in Eq. (3.13) which throws the solution away from the global optimum. For example, the point cloud generated by a rotating 3-D lidar sensor has a non-uniform sample density that can cause (distant) objects to sometimes be scanned, but sometimes missed. A missing data point can be seen as an outlier and causes ambiguity which greatly reduces the accuracy of ICP. Practical implementations of ICP often apply thresholds or probability of distances to discard outliers, but this also creates the need for an appropriate threshold which depends on the structure of point sets and thus is hard to choose. Another strategy for coping with outlier noise is to reject outliers by adopting a coarse to fine process, which can also perform poorly in the case of a large amount of outliers in the point sets. A popular method for robust fitting in the presence of outlier noise is the Random Sample Consensus (RANSAC) which is designed to cope with large percentages of outliers in the data [42] and can be applied to iteratively

estimate the rotation and translation by using a subset of 3-D points which produce the maximum number of inliers. Although robust, RANSAC methods are based on sampling which makes them slow to converge.

The approach taken in the proposed method is guided by the idea that the vehicle motion happens on a 2-D plane and can be explained by matching the features of the environment projected on the ground plane. A large proportion of the 3-D information, which otherwise would increase the computational load in ICP based methods, becomes irrelevant for this task. The proposed odometry algorithm starts by computing the occupancy map of the local environment, as explained in the steps in Section 3.4. Practically, we will use the log-odds representation from Eq. (3.11) to compute the current log-odds map $\dot{l}(\mathbf{x}_t)$ from only the current data $\mathbf{z}_t$ (setting $l(\mathbf{x}_{t-1})$ to zero), and compare it to the previously built log-odds map $l(\mathbf{x}_{t-1})$ which contains all past observations. Computing $\dot{l}(\mathbf{x}_t)$ can be performed independently for each log-odds grid cell in a single step:

$$\dot{l}(\mathbf{x}_t) = \ln \frac{p(o(\mathbf{x}_t)|\mathbf{z}_t)}{1 - p(o(\mathbf{x}_t)|\mathbf{z}_t)} + \ln \frac{1 - p(o(\mathbf{x}))}{p(o(\mathbf{x}))} + l(\mathbf{x}_{t-1}), \tag{3.14}$$

where the term $l(\mathbf{x}_{t-1})$ is zero by definition because we are using only the current observations to compute $\dot{l}(\mathbf{x}_t)$.

The proposed algorithm then tries to register $\dot{l}(\mathbf{x}_t)$ to the map $l(\mathbf{x}_{t-1})$ using image registration and approximate the solution in Eq. (3.13). Since the sensor is rigidly attached to the vehicle, the pose of the sensor corresponds to the orientation and position of the vehicle. At initialization, we set the pose at the origin and all consecutive coordinate transforms are relative to the pose P_0:

$$P_0 = (R|\mathbf{t}) = (I|\mathbf{0}). \tag{3.15}$$

Since our occupancy map consists of equally sized grid cells which are regularly spaced, we can safely treat the two maps $\dot{l}(\mathbf{x}_t)$ and $l(\mathbf{x}_{t-1})$ as gray-scale images where each grid cell is a pixel. From the example shown in Figure 3.5 it is apparent that vehicle rotation ω_z will produce rotation of the image features in $\dot{l}(\mathbf{x}_t)$ relative to $l(\mathbf{x}_{t-1})$, and vehicle translation $\mathbf{t}$ will shift the rows and columns respectively.

Estimating the pose matrix P_t in this image domain becomes as simple as estimating rotation and translation through image registration. We propose to use the Phase-Only Correlation [43] (POC) registration method to solve for the optimal coordinate transform of the two images. POC is a frequency domain technique used to estimate the delay or shift between two copies of the same signal. This technique is based on the shift properties of the Fourier transform. Compared to the classical cross-correlation method the accuracy by which the peak of the correlation function can be detected by POC is much higher [44]. It is known [45] that the phase correlation always contains a single coherent peak at the point of registration corresponding to signal power, and some incoherent peaks which can be assumed to be distributed normally over a mean value of zero. The amplitude of the coherent peak is a di-

rect measure of the degree of similarity between the two images. More precisely, the power in the coherent peak corresponds to the percentage of overlapping areas, while the power in incoherent peaks correspond to the percentage of non-overlapping areas. The steps for estimating the rotation are visualized with example data in Figure 3.8. The two input images $\dot{l}\left(\mathbf{x}_t\right)$ and $l\left(\mathbf{x}_{t-1}\right)$ (a-I and a-II in Figure 3.8) are transformed using the 2-D DFT, with a slight abuse in notation:

$$\dot{L}_t = \mathcal{F}\left(\dot{l}\left(\mathbf{x}_t\right)\right);\ L_{t-1} = \mathcal{F}\left(l\left(\mathbf{x}_{t-1}\right)\right), \tag{3.16}$$

where we use $\mathcal{F}\left(\mathbf{x}\right)$ instead of $\mathcal{F}\left(u, v\right)$ to denote the two-dimensional Fourier transform of image $\mathbf{x}$ at coordinates (u, v), while the respective amplitude spectra are $\left|\dot{L}_t\right|$ and $\left|L_{t-1}\right|$ (b-I and b-II in Figure 3.8). Using the shift-invariance property of the Fourier transform, we can use the amplitude spectra for estimating the rotation difference as follows. We first transform them into polar ρ, θ coordinates (c-I and c-II in Figure 3.8). Recall that translation along the azimuth axis in polar space equals to rotation in 2-D Cartesian space, thus the vehicle rotation ω_z can be estimated from the 2-D convolution using the cross-power spectrum R, written out entry-wise for element index (j, k):

$$R_{t,jk} = \frac{\dot{L}_{t,jk}\overline{L_{t-1,jk}}}{\left|\dot{L}_{t,jk}\overline{L_{t-1,jk}}\right|}, \tag{3.17}$$

where $\overline{L}$ is the complex conjugate of the polar spectrum of the log-odds map $l\left(\mathbf{x}\right)$. The phase-only correlation is defined as the Inverse Discrete Fourier Transform of Eq. (3.17):

$$\boldsymbol{r}_t = \mathcal{F}^{-1}\left(R_t\right), \tag{3.18}$$

where $\boldsymbol{r}_t$ in our case is a 2-D array of non-negative values (d in Figure 3.8). Recall that $\boldsymbol{r}_t$ is the phase-only correlation of the Fourier spectra of the input maps which we converted into polar coordinates. The orientation of the vehicle along the z-axis (and hence its ego rotation) is proportional to the horizontal (azimuth) coordinate the horizontal coordinate in $\boldsymbol{r}_t$. The correlation peak in $\boldsymbol{r}_t\left(x, y\right)$ therefore estimates the physical rotation along the z-axis (yaw) up to a scaling factor M:

$$\omega_z \approx \frac{\Delta x}{M} = \left[\Delta x\right] + \left\lfloor \Delta x \right\rfloor = \underset{x}{\operatorname{argmax}} \sum_i \boldsymbol{r}\left(x, i\right) + \left\lfloor \Delta x \right\rfloor, \tag{3.19}$$

where $\left[\Delta x\right]$ is the integer part of the horizontal coordinate of the peak in $\boldsymbol{r}_t$ estimated as the position of the maximum of the sum of the POC along the vertical direction (e in Figure 3.8) and the scaling factor M converts from the POC pixel values into physical rotation in radians. The accuracy of the integer part of the solution is bounded by the resolution of the grid cells, the resolution of the FFT as well as the polar transform. The remaining fractional part $\left\lfloor \Delta x \right\rfloor$ can estimated by means of sub-pixel fitting using the *Foroosh* method [44] which fits a $\mathrm{sinc}()$ function through the peak and its adjacent

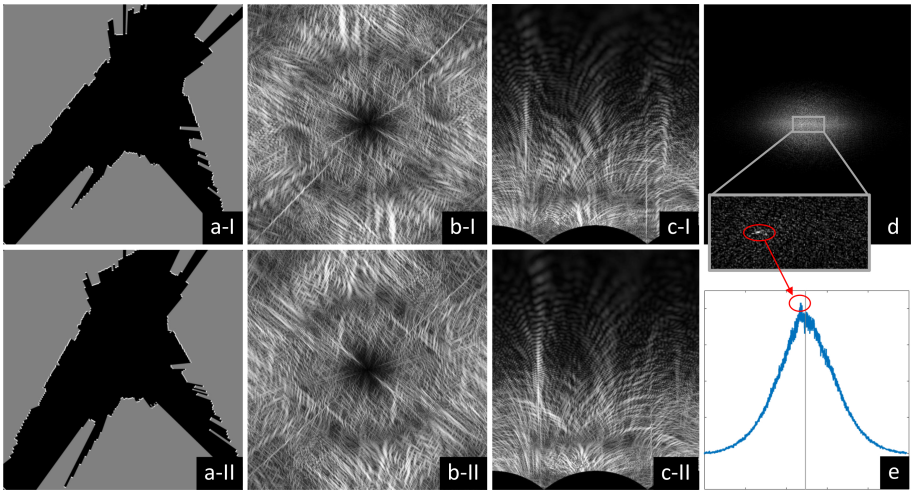


Figure 3.8: Visualization of the steps in the POC algorithm registering two occupancy maps a-I and a-II taken during a right vehicle turn. The images b-I and b-II represent the amplitude spectra (suppressed low frequencies for visualization) computed from Eq. (3.16). The images c-I and c-II represent the polar transform of the spectra in b-I and b-II respectively. The image in d) represents the POC as computed from Eq. (3.18) and the plot in e) is the sum of the POC along the range dimension. The red ellipse indicates the position of the peak in the POC.

cells.

The estimated ego-rotation ω_z is proportional to the integer position of the peak in $\boldsymbol{r}$ plus the sub-pixel displacement (see the derivation [44], equations 18-22):

$$\omega_z \approx \frac{1}{M}\left[[\Delta x] + \frac{\sum_i \boldsymbol{r}\left(1+[\Delta x], i\right)}{\sum_i \boldsymbol{r}\left(1+[\Delta x], i\right) \pm \boldsymbol{r}\left([\Delta x], i\right)}\right]. \tag{3.20}$$

Once ω_z has been estimated, this transform is applied to $\dot{l}\left(\mathbf{x}_t\right)$ to match $l\left(\mathbf{x}_{t-1}\right)$. The only remaining difference of the two maps can be assumed to be due to translation of the vehicle. The longitudinal translation $\mathbf{t}_\rho = \triangle\rho$ can thus be estimated by applying the same technique from Eq. (3.16) through Eq. (3.19), but without the polar transform. Finally, for each time step t the 6-DOF pose change matrix $P_t = \left(R_t \left| \mathbf{t}_t \right.\right)$, approximated by the planar 3-DOF pose change (assuming $\omega_x = 0$; $\omega_y = 0$; $\triangle z_t = 0$) is computed as:

$$P_t = \left(R_t \left| \mathbf{t}_t \right.\right) \approx \left[\begin{array}{ccc|c} \cos\omega_z & -\sin\omega_z & 0 & \triangle\rho_t \cos\left(\omega_z\right) \\ \sin\omega_z & \cos\omega_z & 0 & \triangle\rho_t \sin\left(\omega_z\right) \\ 0 & 0 & 1 & 0 \end{array}\right]. \tag{3.21}$$

Recall that the algorithm presented in this section can also be used for the purpose of mapping the environment through applying Eq. (3.11) with the current pose esti-

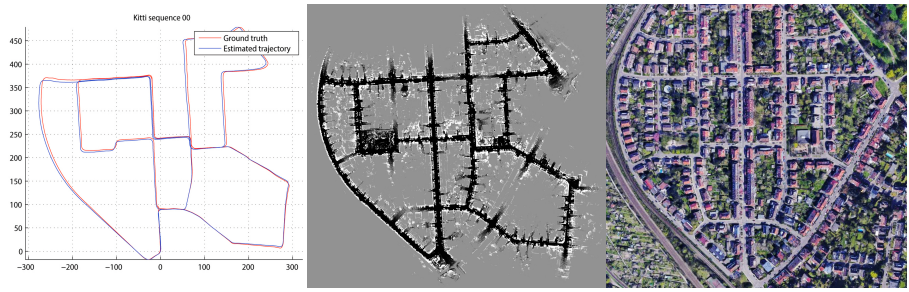


Figure 3.9: Left: estimated trajectory for KITTI sequence 00, middle: computed map with probability of occupancy, right; satellite image of the area (Weiherfeld, Karlsruhe in Germany).

mate P_t. Since the purpose of the method is odometry, the fidelity of the computed maps is of secondary interest. However, since the current map is registered to the aggregated map from the past, the quality of the odometry and mapping are interconnected. The accuracy of the registration is affected by the quality of the past map and how well it matches with the one reconstructed from the current observations. The dependency is relative to the vehicle speed and sensor range since we need to register the current map $\dot{l}\left(\mathbf{x}_t\right)$ only to the the section of the past map $l\left(\mathbf{x}_{t-1}\right)$ that is currently observable. For example, using a lidar sensor the current map $\dot{l}\left(\mathbf{x}_t\right)$ is computed in a grid with a size of the maximum range of the lidar (usually ˜100m) and for its registration with $l\left(\mathbf{x}_{t-1}\right)$ we do not need the historic parts beyond this maximum range because they do not influence the registration in any way. In this analysis we did not give attention to this effect which remains to be addressed in the future.

As a closing remark, we present an example of a vehicle trajectory estimated by this method (left plot in Figure 3.9) as well as the respective probability of occupancy map (middle image in Figure 3.9) and a satellite image of the environment (right image in Figure 3.9). Note that this map was reconstructed without the use of loop closing algorithms even though several trajectory loops can be identified. The accuracy of the odometry and the fidelity of the computed map can be appreciated qualitatively by comparing the shape to the satellite image. A more thorough quantitative evaluation of the proposed odometry method follows in the experiments section.

3.6 Experiments and results

The presented algorithm was implemented and evaluated on data captured in a real-world environment. Accuracy was measured in terms of pose estimation quality for various trajectory lengths and traffic environments. Additionally, simulated tests were carried out in order to find the limits of robustness in the presence of signal degradation. Due to the hard real-time requirements for vehicle odometry, an efficient GPU program was implemented in the programming language Quasar [46, 47]. Poses

estimated by the algorithm were compared to ground truth poses generated by a vastly more accurate sensor. Raw data streams provided by the lidar recordings from the KITTI dataset [20, 32] were used to perform mapping and registration analysis. This particular dataset was chosen as it was the most comprehensive data gathering study about autonomous vehicles driving through publicly accessible roads at the time of development. The relevant data in this experiment was captured by the Velodyne HDL-64E 3-D lidar as input, while the automotive grade Inertial Navigation System OXTS RT3003 was used for ground truth. The lidar provides 10 point clouds per second, using 64 laser beams to measure scene geometry up to $120m$ with accuracy of $3cm$. The ground truth INS provides 250 poses per second with pitch/roll accuracy of 0.03°, yaw accuracy of 0.15° and $1cm$ positional accuracy.

Evaluation of trajectory accuracy

The experimental dataset contains 21 recordings from driving the vehicle through urban, rural and highway roads in Germany. Since the proposed method assumes a flat world for the mapping, the estimated odometry poses contain information for 3-DOF changes of the vehicle, namely yaw rate and x, y position change relative to the starting pose. Although during the ground plane estimation the pitch and roll angle of the sensor relative to the road surface was estimated, these angles were not used in the further analysis. Also, the absolute elevation is assumed to be the same throughout the entire trajectory. Taking these choices we are able to evaluate odometry from a bird's eye view, i.e., in 3-DOF which is most relevant for the application of autonomous driving. Other applications such as drones might be very sensitive to the estimation of vertical displacements and changes in pitch/roll and the odometry must be analyzed in full 3-D.

In order to compare to other works in the literature, we adopted the evaluation methodology and metrics used in the KITTI dataset. The odometry benchmark in the KITTI dataset compares short sub-sections of the traveled trajectory to the true trajectory as recorded by an INS sensor. Practically, the evaluation measures how much the estimated pose at the end of each sub-section differs from the true pose in terms of average orientation (rotation) and translation error. Average errors for all sub-segments of length $\{100m, 200m, 400m, 800m\}_N$ are computed for every KITTI sequence where an average across the test set is used to rank different methods. The error in orientation is defined through the difference between the ground truth and estimated orientation matrices R_{gt}, R_{est} while the error in translation through the difference between the translation vectors $\mathbf{t}_{gt}, \mathbf{t}_{est}$ after traversing a segment of length δ. These errors are computed from the relative pose error $\triangle P_{i,\delta}$:

$$\triangle P_{i,\delta} = \left(P_{est,i}^{-1} P_{est,i+\delta}\right)^{-1} P_{gt,i}^{-1} P_{gt,i+\delta}, \tag{3.22}$$

where i is the starting time of the segment and δ is its duration in time. The time interval defined by δ is variable and depends on speed of the vehicle i.e., the time it takes for the car to cover any of the N preset distances. P_i^{-1} is the pose matrix at the

Error	3-DOF		6-DOF	
Sequence	$\epsilon_{\mathbf{t}}$ [%]	$\epsilon_{\mathbf{R}}$[deg/m]	$\epsilon_{\mathbf{t}}$ [%]	$\epsilon_{\mathbf{R}}$ [deg/m]
KITTI 00	0.661	0.0075	1.434	0.0151
KITTI 01	0.547	0.0053	2.004	0.0118
KITTI 02	0.720	0.0078	3.126	0.0246
KITTI 03	0.895	0.0192	3.068	0.0222
KITTI 04	0.506	0.0022	2.305	0.0074
KITTI 05	0.598	0.0054	1.323	0.0131
KITTI 06	0.562	0.0058	1.350	0.0146
KITTI 07	0.724	0.0076	1.165	0.0182
KITTI 08	0.838	0.0078	1.678	0.0149
KITTI 09	0.779	0.0082	3.890	0.0296
KITTI 10	1.136	0.0098	3.977	0.0254
KITTI 00-10	**0.741**	**0.0079**	**2.301**	**0.0179**
KITTI 11-21	/	/	**1.89**	**0.0083**
KITTI$_{TOP}$ 11-21	/	/	0.54	0.0013
KITTI$_{WORST}$ 11-21	/	/	21.47	0.0425
KITTI$_{MEAN}$ 11-21	/	/	2.97	0.0069

Table 3.1: Translation and rotation errors of the proposed method, evaluated on 21 sequences of the KITTI odometry dataset.

start of the segment and $P_{i+\delta}^{-1}$ is the pose matrix at the end of the segment. Defining the odometry error in such a way is highly relevant for object tracking which will be addressed in the next chapters. Knowing the error in orientation and translation with respect to a starting point 100m, 200m, etc., behind puts an upper limit in the precision of tracked objects which the vehicle encounters as it is moving. For the rotation error we use the rotation submatrix $\triangle R = \left(R_{est,i}^{-1} R_{est,i+\delta}\right)^{-1} R_{gt,i}^{-1} R_{gt,i+\delta}$ (recall $P = \begin{bmatrix} R & \mathbf{t} \\ 0 & 1 \end{bmatrix}$), and compute the rotation difference:

$$\epsilon_{R,i,\delta} = d_R \left(\left(R_{est,i}^{-1} R_{est,i+\delta}\right)^{-1} R_{gt,i}^{-1} R_{gt,i+\delta} \right), \tag{3.23}$$

where the angle of rotation along the axis of the starting pose is computed from the trace of the matrix:

$$d_R\left(R\right) = \cos^{-1}\left(\frac{\mathrm{tr}\left(R\right) - 1}{2}\right). \tag{3.24}$$

For the translation error $\epsilon_{\mathbf{t},i,\delta}$ we use the translation vector $\triangle \mathbf{t} = \left(R_{est,i}^{-1} R_{est,i+\delta}\right)^{-1} R_{gt,i}^{-1} R_{gt,i+}$

$$\epsilon_{\mathbf{t},i} = d_{\mathbf{t}} \left(\left(\mathbf{t}_{est,i}^{-1} \mathbf{t}_{est,i+\delta}\right)^{-1} \mathbf{t}_{gt,i}^{-1} \mathbf{t}_{gt,i+\delta} \right), \tag{3.25}$$

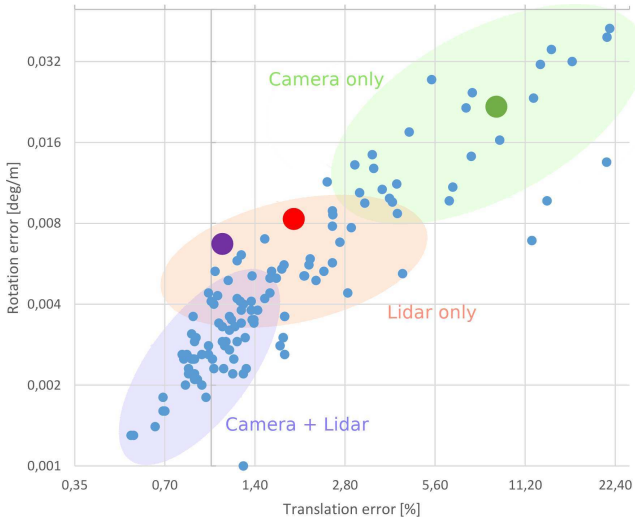


Figure 3.10: Log-log plot of the accuracy of all odometry algorithms evaluated on the KITTI odometry dataset (6-DOF) at the time of writing. The large markers indicate methods developed within the IPI research group: purple-Vlaminck et al., red-proposed and green-Van Hamme et al. The proposed method provides a very fast and simple implementation with accuracy competitive with most methods in the literature.

such that:

$$d_{\mathbf{t}}\left(\mathbf{t}\right) = \left\|\mathbf{t}\right\|_2 . \tag{3.26}$$

Finally, dataset average errors are averaged over all possible combinations of sub-trajectories of length $100m$ to $800m$ for all sequences. The dataset average rotation error is expressed in degrees per meter traveled, while the dataset average translation error is expressed in percent. For example, if the orientation of the vehicle after driving for 1Km is one degree and its position is off by 1m, then the reported rotation error will be $0.001°m^{-1}$ while the reported translation error will be 0.1%.

The results shown in Table 3.1 and compared to other methods on Figure 3.10, are in the form of average degrees per meter rotation error and average percentage (meters per segment length) translation error for each sub-segment of length 100m to 800m. On the 6-DOF KITTI test set benchmark (sequences 11 to 21) the proposed odometry method measures average translation error of 1.89% and average rotation error of $0.0083°m^{-1}$. If we then evaluate the accuracy only on the 3-DOF that the proposed method estimates, the average translation error is 0.29% while the average rotation error is $0.0038°m^{-1}$. Additionally, in Table 3.2 we compare the accuracy of the proposed method to two odometry algorithms developed within the IPI research group.

The bar plot shown in Figure 3.11 ranks all of the submitted methods on KITTI according to the average run-time for a single frame. Note that this is a self-reported

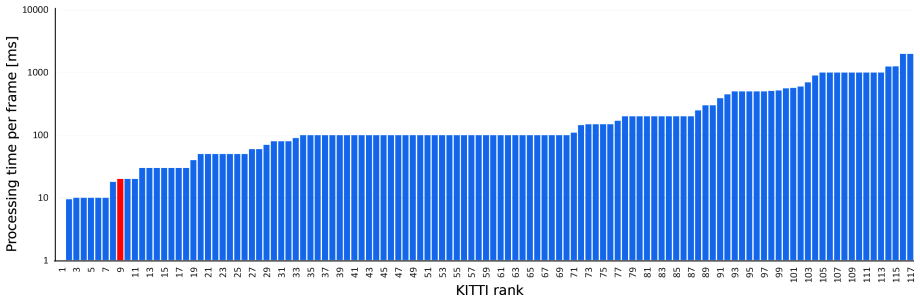


Figure 3.11: Self-reported run-time (in log-scale) of the odometry algorithms evaluated on the KITTI benchmark.

Method	Sensor	Setting	Loop closure	$\epsilon_{\mathbf{t}}$ [%]	ϵ_R [deg/m $\times 10^2$]
Proposed [9]	Lidar	3-DOF	No	2.3	1.8
Van Hamme et al. [48]	Visual	3-DOF	No	8.9	2.2
Vlaminck et al. [49]	Lidar	6-DOF	Yes	1.1	0.7

Table 3.2: Accuracy of the odometry algorithms developed within the IPI research group (6-DOF test). Results from evaluation on the first 11 sequences of the KITTI odometry dataset (KITTI 00-10).

metric that authors need to provide when submitting their results on the on-line evaluation server. The values should be interpreted with caution since some of the methods report CPU, while others GPU time. Regardless, our GPU implementation of the proposed method was ranked among the top 10 fastest with an average time of 20ms per frame.

At the time of the submission in 2016, there were multiple methods which scored higher on both rotation and translation metrics; see the top two plots on Figure 3.10. However, most of the top performing algorithms rely on a combination of camera and lidar information, loop closure, scan to map matching, forward-backward optimization, etc. Additionally, the evaluation server computes the full 6-DOF pose estimation error which creates an unfair disadvantage because the proposed method only computes 3-DOF pose changes.

When evaluated on the full 6-DOF ground truth, the proposed method achieves accuracy which is 10% worse than the state of the art by means of translation error and 20% worse by means of rotation error. This discrepancy is to be expected under 6-DOF evaluation because even though the dataset contains mostly flat roads, there are slight undulations which create changes in the roll-pitch aspects of the pose and the z-axis in the translation which our method assume to be zero. However, when qualitatively evaluating the shape of trajectories from other methods, we found that even

much worse methods than ours (in terms of the rotation and translation metrics) produce reasonably accurate odometry as judged by the fact that the estimated trajectories form closed loops. It is in the opinion of the authors that the pursuit of significantly more accurate odometry than the one estimated by the proposed algorithm has little impact on the performance of perception systems such as object tracking, as will be presented in the following analysis.

Evaluation of robustness

The KITTI odometry dataset was captured in 21 different locations of the *same* city and all data captures were made under *good* weather conditions. This data set only allows for the evaluation of odometry under a sub-set of conditions which does not represent the complete set of working conditions in autonomous driving. For example, a significant missing scenario is nighttime driving and driving in rain/snow/fog. Such conditions will most certainly degrade the camera and lidar measurements which can stress some odometry method that is not designed to cope with noise or outlier data. The quantitative effect of such degradation on the computed vehicle trajectory remains largely unknown.

In order to measure the robustness of the proposed method, in this block of experiments we will simulate scenarios with bad weather conditions using the original KITTI sequences. In order to keep the experiment sensible, we will apply two types of errors to the input lidar data using realistic degradation models. These experiments evaluate the robustness of the method to measurement noise as well as missing data and outliers.

First, the effect of lesser quality 3-D lidar data produced under bad weather conditions was evaluated by simulating a wet environment. During rain, snow or fog when the environment becomes wet, a thin water film can cause the surface of objects to become darker [50]. The main cause for this darkening is the possibility of total internal reflection at the water-air boundary. Some of the light reflected from the Lambertian surface will be reflected back to the surface by the water-air interface. This light is then subject to another round of absorption by the surface before it is reflected again. This can lead to a sequence of multiple absorption, resulting in a darkening of the surface. Such surfaces are more difficult to measure by lidar because of the low signal-to-noise-ratio and cause the measured distances to be inarticulate. Practically, we added white Gaussian noise to the position of each point of the input lidar data, sampling from a zero-mean distribution $\mathcal{N}(0, \sigma)$ with variance in the range $\sigma \in [3cm, 100cm]$ (the Velodyne HDL-64E used to capture the data has a typical error of 3cm). Then, the occupancy mapping and proposed odometry was run using the same settings.

The effect of the parameter σ on the translation and rotation accuracy was evaluated by comparing the output to the ground truth as described in the previous experiment. The left plot on Figure 3.12 shows the relationship between measured translation error (red line), rotation error (blue line) and various noise levels σ. A trend of increasing error with the increase of noise variance can be observed, however, both

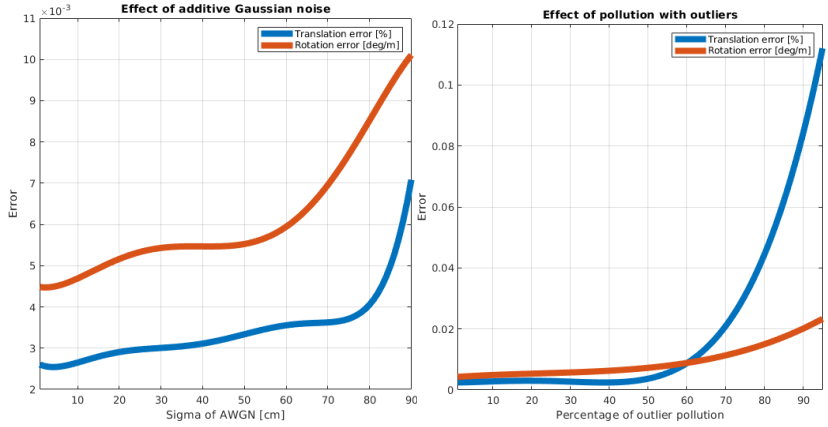


Figure 3.12: Destructive testing of the robustness of the proposed odometry method, left: testing against measurement noise (Additive White Gaussian Noise), and right: testing against presence of outliers (random measurements uncorrelated with the scene content).

accuracies remain relatively low until unrealistically large errors ($\sigma > 50cm$) are simulated. Most contemporary 3-D lidar sensors have a reported measurement variance well within this range.

Lastly, we test the robustness of the method to outliers by simulating environments where the air is saturated with particles (heavy rain, snow, smog, dust, etc.). Compared to the previous experiment, here we expect that the lidar beams will randomly reflect off of particles in the atmosphere, spontaneously producing false range readings. For this test, the input point clouds containing n points were polluted by the introducing m random outliers.

The outliers were placed at random positions irrespective of the content of the measured data. The limitation of this experiment is such that it is possible for an outlier point to be placed behind an object which is not entirely realistic. A more accurate outlier model would be to perform ray-tracing on each point and only pollute the free space with outliers. However, for computational reasons we were unable to perform this and the results should be interpreted within the limitations of the experiment design. Regardless of the outlier process, the odometry will be challenged in a way that the optimal solution will minimize the error between all points, both real and outliers. Since the phase only correlation estimation technique is extremely robust to such outliers, it can be expected that a good translation and rotation estimate can be made as long as most of the occupancy map consists of static content $(n > m)$. Indeed, as seen on the right plot on Figure 3.12, odometry accuracy remains largely unaffected up to the point where half of the input data is static content. When the percent of outliers outgrows the percent of static content the estimation quickly becomes unreliable. This, however, is an extreme example of outlier pollution rarely encountered in the real world.

3.7 Conclusion and practical implications

The proposed ego-localization method based on lidar odometry was published as an article in the proceedings of the VISIGRAPP 2016 conference [9]. It demonstrated competitive trajectory estimation even when compared to methods designed to estimate ego-motion with 6-DOF. By assuming a locally flat world and the 2-D occupancy grid model, registration of consecutive lidar point clouds becomes fast, accurate and robust at the same time. The combination of these three qualities is rarely achieved in the literature which makes the proposed method applicable for real-world tasks. Thus far, the method has been implemented in various programming languages (Quasar [46], MATLAB and Python) and has enabled the further research of environmental perception throughout this thesis.

The practical applications of the proposed method are numerous, and throughout the time of the research in this thesis, it has been re-used for multiple research projects. To illustrate just how accurate the computed odometry is, consider that a state of the art lidar-based object detector, such as Centerpoint [51], detects road users with mean average translation error of 0.262m. Similarly, the state of the art camera object detector FCOS3D [52] achieves a mean translation error of 0.69m using well calibrated data. At typical driving speeds of 15m/s and typical sensor sampling rate of 10Hz, the vehicle covers 1.5m per second during which the sensors make 10 observations. The proposed odometry method is expected to make 1.1cm error in the positional estimate between two samples (recall the mean the errors from Table 3.1: 0.74% translation error and 0.007[deg/m] rotation error). When compared to the positional accuracy of object detection, the proposed odometry is one to two orders of magnitudes more accurate. The apparent lack of accuracy compared to other methods on KITTI can mainly be attributed to the fact that the benchmark is designed to evaluate the full 6-DOF odometry which has limited value in real world applications.

For short-time object detection and tracking on the ground plane, the vehicle position, as estimated by the proposed lidar odometry, can be considered as perfectly accurate. This simplifies the matching of object detections over time because the uncertainty due to ego-motion can safely be ignored. The uncertainty of the detections is thus governed only by the sensor model which greatly simplifies the tracking algorithm presented later in this thesis. It can be concluded that, within the realm of automotive environmental perception, any further improvements in odometry accuracy will have an insignificant impact on the perception accuracy and the effort should be spent in identifying border cases of performance in order to increase the robustness. The proposed odometry algorithm, being relatively simple and robust, is an excellent candidate for the application. As an ultimate testament to the performance of this method, a collage of estimated trajectories for several difficult KITTI sequences is presented on Figure 3.13.

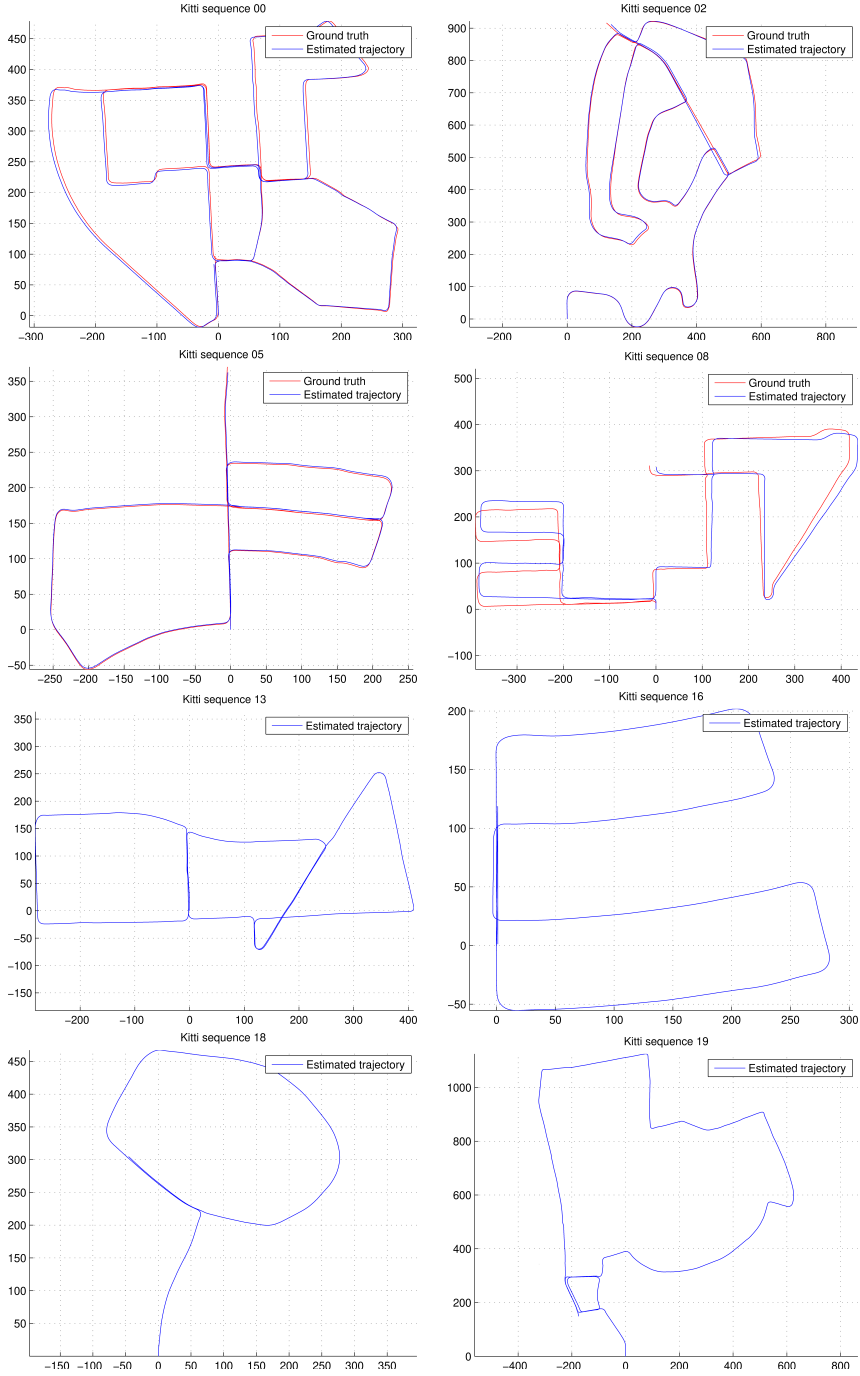


Figure 3.13: Estimated trajectories (blue) for difficult KITTI sequences. Ground truth trajectories (red) are provided for some of the sequences.

4

Single and multi-sensor depth reconstruction

4.1 Introduction

The human brain's biological perception system interprets the various objects in the 3-D environment using inputs from stereo vision, contextual information and prior experiences. Using its visual short term memory our brains are able to store visual information for a few seconds so that it can be used in the service of ongoing cognitive tasks. Such interpretation of the environment can therefore survive eye movements, eye blinks, and other visual interruptions, maintaining continuity across these interruptions. Although this process is highly subjective and difficult to accurately replicate, scientists have exploited various ideas to come up with ways of reconstructing the 3-D world from various sensor inputs.

Most perception systems comprise of a camera and use indirect principles for ranging. The camera images are used to interpret the scene from the visual content while ranging, needed to localize the interpreted objects, is inferred from the images using statistical models. Such an artificial perception system could theoretically outperform human vision by using more sensitive cameras, more precise ranging, smarter reasoning, faster processing, etc. Moreover, an artificial perception system does not suffer from strain which sets in after prolonged cognitive tasks in most biological systems. An artificial perception system can thus remain in optimal operation indefinitely.

Depth estimation is the process of computing the probable range for all points within the field of view i.e., the camera image. In terms of detected objects, depth

estimation refers to the ranging process i.e., computing the range of individual objects. If we have the depth value for each image pixel, then ranging of detected objects becomes as simple as looking up in the depth image. However, the estimation of a depth image is not a trivial task. One method for computing a depth image is to find the vanishing lines in the image and use them to infer object ranges from their appearance along the vanishing lines. Closer objects appear lager and distant objects are smaller. This task is called monocular or blind depth estimation and will be briefly discussed in Section 4.5.3, however, as it leads to ambiguous depth estimates it is given limited interest in this thesis.

More accurate depth perception can be achieved by means of sensors which directly (lidar, radar or ultrasound) or indirectly (stereo, structured light or structure from motion) perceive the scene geometry. Stereo and structure from motion methods operate on a similar principle where the distance to objects is inferred from their apparent shift (disparity) in the image content when the camera is moved. Stereo reconstruction uses two, slightly horizontally offset, cameras that look in the same direction in the scene and measure the image disparity for each pixel. Knowing the distance between the two cameras (their baseline) and the camera focal lengths, we can apply a simple trigonometric function to transform disparity values into ranges. Structure from motion, on the other hand, uses a single camera which moves through time and captures the scene from slightly different angels. Assuming that the scene is static and knowing how far the camera moved, structure from motion methods can accurately transform the perceived disparity of image content into depth values. Both methods, however, suffer from estimation errors in image regions which are devoid of textured content e.g., clear sky, road surfaces, walls, etc. The problem is especially noticeable for distant objects which have very small disparity in the camera images.

Depth perception using 3-D lidars and imaging radars provides the most accurate distance information, but due to the limited resolution of these sensors the depth images contain only sparse measurements, see the example shown on the top row in Figure 4.1. Estimating dense depth from accurate sparse measurements (i.e., depth completion) is valuable as even smaller detections in the image can be accurately ranged from the depth pixel values. This chapter of the thesis explores methods for estimating depth images from camera and lidar which are both dense and accurate. The remainder of the chapter is structured as follows, we start with an overview of the state-of-the-art in Section 4.2. Then, in Section Section 4.3.2 we deal with the problem of monocular depth estimation proposing a convolutional neural network which maps images directly into depth values. In Section 4.3.1 we use a 3-D lidar sensor to generate sparse depth images which we then complete using a novel semantically-aware filter. Finally, the two methods presented in Section 4.4 perform depth completion by fusing camera and sparse lidar depth images using an early fusion convolutional neural network. All of these methods have been evaluated for accuracy of their reconstructions and the results of these experiments can be found in Section 4.5. We conclude this chapter with an analysis on how to integrate the proposed methods into a larger detection and tracking system and discuss the practical implications and future directions of work in Section 4.6.

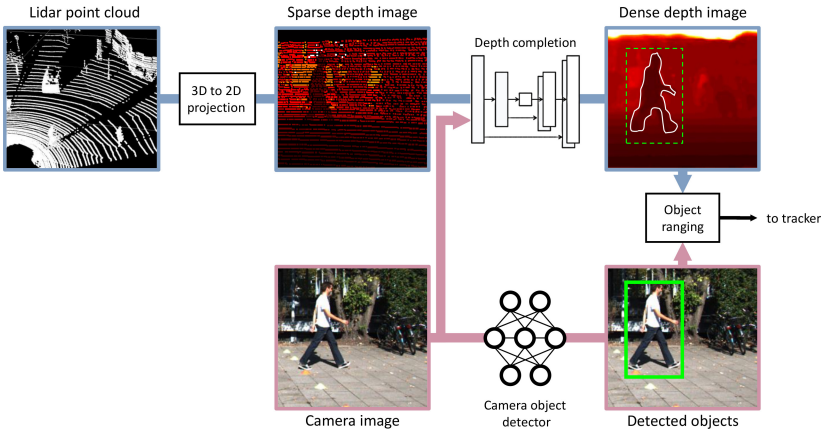


Figure 4.1: Visualization of camera, lidar and radar measurements in a typical autonomous driving use-case. 2-D detection of road users can be performed using the camera image (top) 3-D detection is done by lidar (middle) and radar (bottom). For reference, the images show also the scene depth measured by lidar (middle) and reflected radar signal (bottom).

4.2 Literature overview

Depth reconstruction using a single camera

In the case of a single camera, where range information is not directly available, depth estimation can be performed in one of two ways: depth estimated from camera motion or depth estimated using statistical modeling of image cues. Assuming a single moving camera, one can estimate the motion using matching of subsequent key-points [53] and by knowing specifications such as camera focal length and optical axis, tracked pixels can be back-projected into 3D space. This technique, most often cited as structure from motion [54], can be effectively applied in realistic problems such as Simultaneous Localization and Mapping as we saw in the previous chapter. When the camera is static, depth can be modeled as a function of the focus [55] or illumination changes [56] using prior knowledge of the lighting and the environment. Without such assumptions, the inverse problem of projecting image pixels in the 3D world has no unique solution. However, it is a fact that the human brain can grasp the depth structure of the scene even with only one eye. This is because the brain has capacity to learn high level concepts from past experiences and exploit monocular cues such as perspective or color contrast.

One of the pioneering and most notable approaches in blind/monocular depth reconstruction is the work by Saxena et al. [57]. Using a 3-D laser scanner they collected a small scale ground truth dataset that was used to train a Markov Random Field (MRF) which predicts depth as a posterior distribution given a set of image features. This basic model uses L-2 term in the MRF interaction potential computations which captures depths and interactions of depths between several spatial resolutions.

Although these results advanced the state of the art results at the time, these techniques [57, 58]have limited applicability in a sense that they rely on a fixed set of absolute and relative depth features. These fixed features must be adjusted individually for each specific setup where sensor size and lens type can vary greatly.

With the recent advancements in computer hardware and Deep Learning, convolutional neural networks are becoming a more attractive solution for the monocular depth reconstruction problem. Eigen et al. [59] propose a two scale CNN to learn coarse and fine depth details directly from RGB images. They argue that much of the reconstruction error using standard element wise metrics may be explained simply by how well the mean depth is predicted, so they formulate a Scale-Invariant Error metric to measure the relationships between points in the scene. This technique produces fairly accurate depth images at the cost of a high computational load, to mitigate this problem they only process the image at 1/4 of the input resolution. Authors in [60] offer a different solution to the problem by using a super-pixel segmentation as the domain of processing. The resulting system yields a 10x speedup over previous approaches while maintaining state of the art accuracy on both indoor and outdoor datasets. An obvious limitation to this approach is that it is severely affected by the accuracy of the super-pixel segmentation which is a complex operation in itself. Another notable recent work by Laina et al. [61] proposes a multi-scale CNN approach for monocular depth reconstruction that tackles the computational issues of the previous authors. They introduce a fully convolutional network with novel up-sampling blocks that outputs higher resolution depth images and at the same time requires fewer parameters and trains on one order of magnitude fewer data than the state of the art. The novelty of the architecture comes from the use of Residual or Skip Layers, first introduced in [62] which in part inspire our proposed approach. One of the proposed methods in this thesis, Section 4.4, is built upon these findings extending them by applying the multi-scale auto-encoder architecture of the U-Net [63].

Depth reconstruction using lidar

The topic of obtaining a dense depth map and interpolation from automotive lidar point clouds has been researched by researchers such as [64] and recently [65]. However, even though these resulting depth images look appealing to the eye, the actual values around object boundaries are far from their correct values. One of the pioneering depth completion methods, [66] considers estimating each missing pixel location in the sparse depth image by means of local interpolation within a square window. The authors analyzed various classical reconstruction techniques which rely on depth information alone such as inverse distance weighting, Shepard's Method, ordinary Kriging, Delaunay triangulation and bi-lateral filtering. Furthermore they introduced a modified bilateral filter which also considers depth dispersion within the interpolation window. This method can crudely model the appearance of an object edge or boundary into two categories: foreground and background. A local segmentation is performed on the depth pixels which produces two clusters from which only the points that belong to the dominant cluster contribute to the bi-lateral filter. These au-

thors found out that even simple techniques such as the minimum and median filter can complete missing depth with comparable accuracy to the more complex bi-lateral filter. A major drawback in this work is the overly-simplified model of the environment which doesn't take into account the geometrical and contextual structure of objects.

Ku et al. [67] propose a surprisingly simple yet efficient depth completion method using a sequence of morphological operations on the sparse depth image. In their experiments they show that a small set of fine tuned dilations and erosions is enough to reconstruct a high quality depth image. By experimenting with various kernel sizes and shapes they come to the conclusion that 5×5 diamond shaped morphological operators are able to outperform even some neural network based methods. However, higher level information about object types and shapes is completely ignored, which can potentially lead to even better reconstructions.

Recently, [68] proposed a method for semantically guided depth completion by means of local plane fitting. They assume that the environment is locally smooth and can be piece-wise modeled by 3D planes. With the intention of preserving depth discontinuities and tiny structures, they introduce an novel edge and semantics aware geodesic distance metric. Additionally, they propose an outlier rejection scheme by utilizing labels from the state-of-the-art semantical segmentation algorithm, FCN [69]. Their reported qualitative results are promising, however, the method is reliant on external segmentation technologies and has since been outperformed by special purpose neural networks.

Uhrig et al. [70] propose a depth completion method by processing the raw RGB-D data cube using a novel neural network. They propose a sparsity invariant convolutional layer which is built using an additional sampling mask. The mask holds binary information about which pixel is scanned by the LIDAR and is used to normalize the convolutional operations. Therefore, the network can easily handle varying degrees of input data sparsity without any adjustments or tweaking of the parameters. One downside of this method is that the network is based on the Fully Convolutional architecture which has a high computational load. Each inference produces a single depth pixel value and thus can not be employed in real time applications.

Depth reconstruction using camera-lidar fusion

The simplest depth completion methods using camera and lidar use the camera image to extract semantic information which is then used to guide a classical image reconstruction algorithm operating on sparse lidar depth images. The method in [71] is one such typical example where the authors propose a guided depth reconstruction filter where guidance is provided by an image of surface normals constrained by a so-called local brightness normal (LBN) derived from the Lambertian model. As most of the object's surfaces are rough in outdoor scene, LBN constraint is derived approximately from the Lambertian model. It provides physical constraints for normal estimation. With the guidance of dense normal, smooth and dense depth is obtained from the guided filter. In the KITTI depth completion data sets, the proposed method outperforms the current non-learning methods. However, it has some limitations that

it over-smooths the results and does not run in real-time.

Similarly, the authors of [72] propose a novel non-learning depth completion method based on camera and lidar exploiting the local surface geometry. The proposed surface geometry model is inspired by the observation that most pixels with unknown depth have a nearby lidar point. Therefore, it is assumed those pixels share the same surface with the nearest lidar point, and their respective depth can be estimated as the nearest lidar depth value plus a residual error. The residual error is calculated by using a derived equation with several physical parameters as input, including the known camera intrinsic parameters, estimated normal vector, and offset distance on the image plane. The proposed method is further enhanced by an outlier removal algorithm that is designed to remove incorrectly mapped lidar points from occluded regions. This method achieves competitive reconstruction accuracy on the KITTI benchmark and is computationally efficient making it useful in any environment.

The method presented in [73] describes another image-guided lidar depth completion algorithm which works completely without deep learning. This method assumes that separate objects in the depth maps mostly consist of the same color but typically differ from the neighboring regions. Under this assumption, the method then decomposes the camera image into Superpixels corresponding to the regions with similar depth value and then merges Superpixels corresponding to same objects by gathering them using a cost map. At the end, the method applies morphological dilation on the sparse lidar depth image and uses the Superpixel object boundaries to confine the dilation. The authors claim state of the art reconstruction accuracy among non-learning depth completion techniques.

Besides these notable non-learning camera-lidar algorithms, there is a plethora of end-to-end learning depth completion methods. We hereby provide a short overview several such methods. The authors of [74] propose a convolutional neural network that is designed to upsample a series of sparse range measurements based on the contextual cues gleaned from a high resolution intensity image. Their approach draws inspiration from related work on super-resolution and in-painting. The proposed dual-backbone architecture seeks to pull contextual cues separately from the intensity image and the depth features and then fuse them later in the network. They argue that this approach effectively exploits the relationship between the two modalities and produces accurate results while respecting salient image structures. These authors have encountered a common problem when applying CNNs on sparse inputs. Namely the depth completion CNNs have poor performance when there is input sparsity. To mitigate this problem they experimented with replacing all convolutions in the depth branch with sparse convolutions but noticed a significant drop in performance. They are more inclined to believe that desirable performance can be achieved with the use of regular convolutions and operations for multi-modal input with simple pre-processing hole filling operations such as morphological filters, fill maps and nearest neighbor interpolation. This notion motivates the pre-processing steps we take in our proposed methods in Section 4.4.

Authors in [75] noticed that a significant challenge in designing CNNs for depth completion is that their output tends to suffer from depth smearing between objects.

They propose a new representation for depth called Depth Coefficients (DC) to address this problem. DC use a one-hot encoding of depth using fixed number of discrete depth values to represent the sparse input. DCs trade memory for precision while enabling convolutions to more easily avoid inter-object depth mixing. Furthermore, they show that the standard Mean Squared Error (MSE) loss function can promote depth mixing, and thus propose instead to use cross-entropy loss for DC. The downside of this approach is that due to GPU memory constraints, it can only process image patches and has a slower execution speed. The authors show that their dense depth estimates can improve object detection compared to sparse depth. Finally, they argue that MSE is a flawed metric to evaluate depth completion, proposing to use thresholded tRMSE and tMAE (where the threshold distance distinguishing within-surface variation from inter-object separation) as they reward high-probable depth estimates and give equal penalty to large errors, which are mostly mixed-depth pixels.

A deep neural network architecture to infer dense depth from an image and a sparse point cloud is proposed in [76]. This approach uses a representation based on inverse mapping of image pixels into 3-D space using the camera calibration matrix. At inference time, the calibration of the camera is fed as an input to the network along with the sparse point cloud and a single image. A Calibrated Back-projection Layer back-projects each pixel in the image to three dimensional space using the calibration matrix and a depth feature descriptor. The resulting 3D positional encoding is concatenated with the image descriptor and the previous layer output to yield the input to the next layer of the encoder. A decoder, exploiting skip-connections, produces a dense depth map. The resulting Calibrated Back-projection Network, or KBNet, is trained without supervision by minimizing the photometric re-projection error. The main benefit of this method is that it can easily use a model trained with a certain sensor platform with a different one at inference time because the network also reads the camera calibration parameters.

In [77] the authors propose a CNN for solving both depth completion as well as semantic segmentation. They use an encoder-decoder network architecture and a sparse training strategy and show that it can efficiently handle sparse inputs of various densities without the need of retraining or any additional mask input. Furthermore, the authors found that varying synthetic densities within range of [0, 1] naturally helps networks to be invariant to different densities. They use a concatenation of RGB and sparse depth at input and optimize the network parameters using the inverse mean average error (iMAE) as a loss function. The experimental results on depth completion outperform all published methods on the KITTI benchmark and are qualitatively remarkable with only 8 layers lidar. Changing only the last layer, the network can also perform semantic segmentation on synthetic and real datasets showing that there is an intrinsic link between the two tasks.

Finally, authors in [78] propose a lidar-only depth completion method that is trained to reconstruct both a dense depth image as well as a corresponding camera image. Specifically, they formulate image reconstruction from sparse depth as an auxiliary task during training that is supervised by the camera images. During testing, the system accepts sparse depth as the only input, i.e., the image is not required. Such a

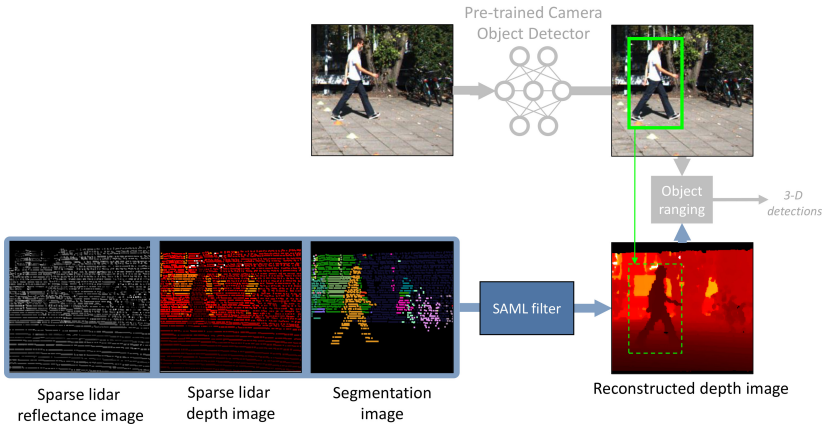


Figure 4.2: Block diagram of the proposed lidar-only depth completion algorithm as a component in an early camera-lidar fusion object detector. The input to the detector consists of an RGB image and a depth image reconstructed using a semantically aware multi-lateral filter. Data from the KITTI dataset.

design allows the depth completion network to learn complementary image features that help to better understand object structures in the sparse lidar data. The extra supervision incurred by image reconstruction is minimal, because no annotations other than the image are needed. The authors claim that this unique design offers significantly improved depth completion via the auxiliary supervision of image reconstruction. The contributions that follow do not explicitly extend any of the aforementioned methods, but were in part inspired by the novel ideas of the state of the art as well as the established signal reconstruction theory.

4.3 Singe sensor depth reconstruction

The two techniques explained in this section outline novelties in single-sensor (camera-only and lidar-only) depth map estimation. The main challenge for camera-only depth perception is learning the inherently ill-posed transformation from 2-D to 3-D which can be approximated using statistical models and a large training dataset. On the other hand, depth map estimation using only lidar data alone is also challenging because of the significant measurement sparsity. The result of the proposed methods has been re-used numerous times within the IPI research group and has resulted in the publication of several novel object detectors: [79–81].

4.3.1 Lidar-only depth estimation

In this section we will analyze the problem of reconstructing accurate depth images from data captured by lidar. As with the previous approach, these reconstructed

depth maps can then be used to range detections from a stand alone detector, or alternatively, fused with camera images to train an object detector, see Figure 4.2. The main focus of the following analysis is reconstruct the depth image from lidar most accurately, while in the experimental section we provide an example of how depth can be further used as additional channel to train an object detector operating on RGB-D images.

We use a common representation for the lidar point cloud and the camera image: a depth map matching the camera field of view and pixel resolution. A point-cloud projected onto this view produces a sparse depth map which we then complete using a signal de-noising and completion theory. At the time of writing, the proposed method represented the state-of-the-art in classical depth completion of 3-D lidar data. The method completes a depth image using high-level semantical information, extracted by segmenting the lidar point cloud itself. This semantical information is used to guide a semantically aware multi-lateral filter (SAML) which preserves not only edges, but complete shapes of objects.

In this single-sensor method the semantical information is extracted also from the lidar point cloud, while the methods in the following sections propose a sensor-fusion approach where the much richer camera image information can be used for even better discrimination between objects and the background. Much of the analysis here will be focused on data captured by the Velodyne HDL-64E automotive lidar, however the developed algorithm can be adjusted to operate over point clouds from other 3-D sensors. During the later stages of the research covered in this thesis the multi-lateral filter was successfully deployed on data captured by the Velodyne VLP-16 and the Ouster OS1-128 lidars.

The method starts by computing a sparse depth image $D\left(.\right)$ and a sparse infra-red reflectance image $I^{[IR]}\left(.\right)$ from the lidar point cloud using the pinhole camera model, see formal definition in Appendix A. An example of how such sparse images look like is shown on the left image in Figure 4.3, where the depth pixels are overlaid on the camera image for reference. The goal of the proposed method is to estimate the missing depth values for each pixel in the depth image, as shown on the right image in Figure 4.3.

Due to the low sampling density of the projected depth image, object boundaries are not well represented and simple interpolation techniques produce unsatisfactory results. For example, reconstructing the missing depth values of pixels near object edges using linear interpolation will produce ramp values which are unnatural. However, it can be expected that the empty depth pixels contain the same or similar structure to nearby, sampled pixels. Object edges can be seen as a discontinuity in the depth function and flat regions have smoothly varying values. Thus it is natural that we do the reconstruction of the depth image using a form of an edge preserving filter. This task has been performed with great success in the image domain using the bi-lateral filter [82]. It follows the paradigm of locally varying filter coefficients that process the image intensity in two directions simultaneously. Two functions, measuring geometric closeness and photometric similarity, are adapting the filter coefficients to the local image patches. The resulting filter is optimized to suppress noise while

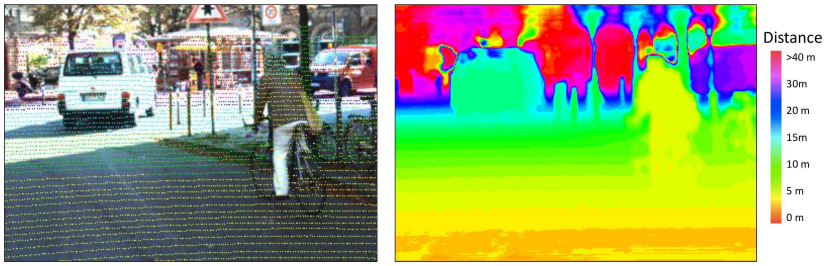


Figure 4.3: Section of a scene from the KITTI tracking dataset. Left: camera image overlayed with projected range values from lidar, right: fully reconstructed depth image from the data on the left.

preserving details around sharp edges. Formally the discrete bi-lateral filter output at pixel coordinates $\mathbf{u} = (u, v)$ is defined as:

$$I'(\mathbf{u}) = \frac{1}{w} \sum_{\mathbf{u}_i \in S} I(\mathbf{u})\, \mathrm{f}(\|\mathbf{u}_i - \mathbf{u}\|)\, \mathrm{g}(\|I(\mathbf{u}_i) - (I(\mathbf{u})\|), \tag{4.1}$$

where the output $I'(\mathbf{u})$ becomes the weighted average of the input value and the product of the kernels $\mathrm{f}(.)$ and $\mathrm{g}(.)$ over the local neighborhood S and the weight factor w is a normalization constant. The first function measures the inverse Euclidean distance of pixel positions within S, and the second function measures the distance in luminance values, usually following a radial basis function. The proposed depth upsampling method performs a similar filtering task on the sparse depth image D by extending the bilateral filter Eq. (4.1) so that it not only depends on the spatial similarity:

$$\mathrm{f}_1(\|\mathbf{u}_i - \mathbf{u}\|) = \exp\left(-\alpha \|\mathbf{u}_i - \mathbf{u}\|_2^2\right), \tag{4.2}$$

but also the infra-red reflectance similarity, a property measured by most lidars:

$$\mathrm{f}_2\left(I^{[IR]}(\mathbf{u}) - I^{[IR]}(\mathbf{u}_i)\right) = \exp\left(-\beta \left(I^{[IR]}(\mathbf{u}) - I^{[IR]}(\mathbf{u}_i)\right)^2\right), \tag{4.3}$$

depth similarity:

$$\mathrm{f}_3(D(\mathbf{u}) - D(\mathbf{u}_i)) = \exp\left(-\gamma (D(\mathbf{u}) - D(\mathbf{u}_i))^2\right), \tag{4.4}$$

and semantical similarity computed from the semantic segmentation image O over the local window S_i:

$$\mathrm{f}_4(O(\mathbf{u}_i), \mathrm{mode}(O(S_i))) = \begin{cases} \delta & \text{if } O(\mathbf{u}_i) = \mathrm{mode}(O(S_i)), \\ 1 - \delta & \text{otherwise,} \end{cases} \tag{4.5}$$

where S_i is an image patch, formally a set of pixel positions $\mathbf{u}_j$ centered around $\mathbf{u}_i$, $O(\mathbf{u}_i)$ is the categorical label of the pixel at $\mathbf{u}_i$ (semantical class, object id or similar) and mode($O(S_i)$) computes the most frequent label of the image patch S_i not counting any missing values. The depth and reflectance similarities are defined as the convex metrics, while the semantical similarity is the discrete metric over the discrete metric space of segmented object labels.

The parameters α, β, γ and δ control the relative influence of each factor on the reconstructed value. Formally the proposed depth upsampling is a multi-lateral extension of Eq. (4.1) which computes a new depth value $D'(\mathbf{u})$ for each pixel position $\mathbf{u}$ as:

$$D'(\mathbf{u}) = \frac{1}{w} \sum_{\mathbf{u}_i \in S} \begin{array}{l} D(\mathbf{u}) \mathrm{f}_1 \left(\|\mathbf{u}_i - \mathbf{u}\| \right) \mathrm{f}_2 \left(I^{[IR]}(\mathbf{u}) - I^{[IR]}(\mathbf{u}_i) \right) \\ \mathrm{f}_3 \left(D(\mathbf{u}) - D(\mathbf{u}_i) \right) \mathrm{f}_4 \left(O(\mathbf{u}_i), \mathrm{mode}(O(S_i)) \right), \end{array} \tag{4.6}$$

where the empty inputs are assigned with object labels based on a local k-NN clustering, with the value of k varying based on the lidar model, usually $k = 3$.

In order to be able to combine the depth and infra-red reflectance filter terms we rely on two important assumptions about the nature of the input signal. Firstly, depth is a smoothly varying property except at object boundaries where the derivative is infinite and secondly, properties of infra-red reflectance image can be approximated with properties of natural light images i.e., smooth local variations and sharp object edges. In practice, it is very difficult to model the infra-red reflectance image since it is product of angle of incidence of the lidar light i.e., the scene geometry and the surface material properties of the object. Based on this assumption, the first factor of Eq. (4.6) allows more influence based on image pixel distance, the second factor similarly looking infra-red measurements to contribute more, the third factor allows similar depth measurements to contribute more while the last factor allows depth to be interpolated from samples within the same object. The dominant object $O(S)$ is recomputed for each position $\mathbf{u}$ and thus every depth pixel $D(\mathbf{u})$ will be reconstructed from the data of the dominant object in its own neighborhood.

The novelty of this method is the proposed algorithm for computing the semantical image O from lidar point cloud itself and not the camera image. This way the semantical image is perfectly aligned with the sparse depth image D. In an automotive context, the objects of interest are usually not physically connected to each other and can be segmented in a birds-eye view based on the region growing algorithm. Object boundaries are therefore very important and can be defined as the limits of free space that spans around the vehicle. To that end, the occupancy grid maps defined in Section 3.4 can be re-used for segmenting the lidar point cloud into non-overlapping objects. In this representation objects of interest can be segmented from the ground plane by applying a threshold on the probability of occupancy.

The proposed method uses a RANSAC based ground estimation technique to fit a plane to the point cloud around the vehicle. Then, an occupancy map is computed by discarding points above the ground plane. In most this can be done using a simple occupancy threshold. Following this step, we are left with a binary grid consisting of

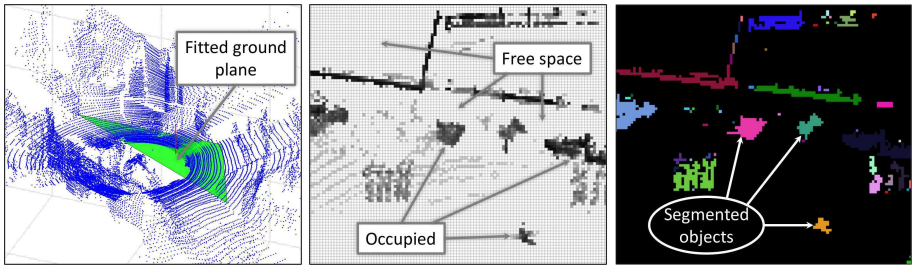


Figure 4.4: Example demonstrating the proposed Lidar point cloud segmentation. Left: the input 3-D point cloud and the estimated ground plane with normal vector, middle: the computed occupancy map for the same scene, and right: a color-coded segmentation map.

free and occupied space. Assuming that each unique object is completely surrounded by free space, we treat every cluster of connected occupied space as a unique object. The actual computation of clusters can be performed by any fast connected components labeling algorithm. The object label of each cluster can then be traced back to individual points in the point cloud and be projected onto the camera image to form $O(\mathbf{u})$. By varying the occupancy grid resolution and threshold parameter, we can tune our segmentation to separate specific objects such as pedestrians, cyclists, cars, buses, etc. A visualization of the intermediate steps of the proposed algorithm are presented on Figure 4.4. On the left plot, the 3-D point cloud is overlaid with the fitted ground plane model; the middle plot shows the computed occupancy map for the scene; and the right plot shows the labeled occupancy map where the colors of the blobs correspond to the segmented objects in Figure 4.5.

In the experimental section we will first measure the absolute accuracy of depth maps reconstructed by the proposed algorithm. Then, we show how by to fuse the reconstructed depth map with an RGB camera frame and train an early fusion camera-lidar pedestrian detector based on the aggregated channel features (ACF) [83]. More details are provided in the experimental evaluation Section 4.5.2.

4.3.2 Camera-only depth estimation

Estimating the depth or distance relative to the camera using the camera view is called monocular depth estimation. This task is slightly different from the estimation of missing pixels in sparse lidar data in a sense that the camera does not offer (not even sparse) direct measurements of distance. Thus, depth needs to be inferred from the content of the image. This process usually relies on assumptions of the scene geometry and statistical and contextual modeling. From the seminal work done by Saxena et al. [57] all the way to the current state-of-the-art monocular depth estimator by Kim et al. [84] the task of mapping image pixel values into depth has been mainly solved by means of supervised learning. Methods most commonly apply a function that decomposes the input image into multiple resolutions and feature repre-

sentations, and then uses regression to transform the feature space into depth values. Best accuracy is achieved when the statistical model is trained using supervision in the form of manually labeled depth maps. However, labeling for depth information is a non-intuitive and expensive process.

Alternatively, training the model parameters can be done in a semi-supervised manner, where ground truth information is provided by another, non-perfect depth sensing device. For example, a stereo camera running a disparity estimation algorithm can provide depth information which can be used to train a monocular depth estimator. Similarly, depth estimated by structure-from-motion can also be used as a semi-supervised ground truth for monocular depth estimation. Unfortunately, these two techniques in themselves provide poor depth information, especially for distant objects. As discussed in the introduction, sensor arrays installed in autonomous vehicle prototypes often include a 3-D lidar or multiple radar sensors. Since these sensors make direct distance measurements using active sensing, the depth information they provide is much more accurate and robust than stereo and structure from motion. Therefore, in our analysis we assume the availability of depth data captured by lidar which we can then use to train our monocular depth estimator.

The proposed method is a convolutional neural network which takes camera images as input and maps them into depth values using supervised training from dense lidar depth images. We use an encoder-decoder network architecture with skip connections based on the U-Net [63] convolutional neural network. It consists of a feature extractor, a fully connected layer and a generator part where each block is linked with the corresponding feature extractor block by skip connections, Figure 4.5. The feature extractor applies 5 consecutive blocks of convolutions and max-pool operations which can compute useful features at different spatial resolutions. The fully connected layer has the capacity to infer the global scale of the environment from a representation in high-dimensional feature space. Finally, the generator is built as the inverse operation of the feature extractor: 5 consecutive blocks of transposed convolutions which upscale the output from the fully connected layer together with the original features from the respective skip connections. The CNN model provides a one channel output with the same resolution as the input image. We use the L-2 loss function to regress the CNN output to a depth map provided by a 3-D lidar. Training is performed in a standard deep learning fashion, using the back-propagation algorithm and the stochastic gradient descent optimizer. More implementation details as well as experimental evaluation of the method is given in Section 4.5.3.

This method computes depth images with the same spatial resolution as the camera image. This makes the process of ranging image detections i.e., their bounding boxes as easy as looking up the depth values in the bounded image area, see the example in Figure 4.5. This example shows how to range pedestrians detected in the image using the depth values within the area of their torso: the center-most 25% region in the upper part of the bounding box. We found that our depth completion and range estimation method provides very reliable ranging when paired with pre-trained image object detectors. We have therefore re-used this technique to compute the range of objects in the image several times throughout this thesis.

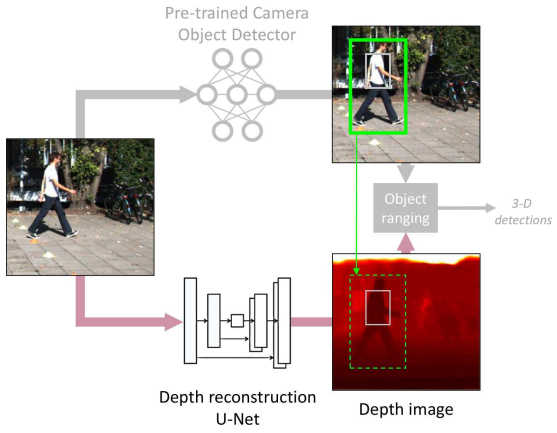


Figure 4.5: Block diagram of the proposed monocular depth estimator as a component of a late fusion object detector. The proposed model learns the mapping between RGB pixel values to depth using offline supervision from lidar depth images.

However, even though CNNs are universal function approximators, monocular depth estimation is an ill-posed and inherently ambiguous problem. This practically means that there does not exist any function which performs the back-projection of 2-D content into 3-D with 100% accuracy. As we will see in the evaluation section, the proposed solution nevertheless outperforms classical reconstruction methods when given sufficiently large supervision data for training. Nevertheless, the achieved accuracy, especially at distance, is still behind what is needed for deployment in safety critical applications such as autonomous driving.

4.4 Depth reconstruction by early camera-lidar fusion

So far we've seen that camera-only depth estimation has rich contextual information but suffers from large range inaccuracy in the distance, while lidar-only depth estimation provides excellent range accuracy but struggles to preserve fine contextual information in sparsely scanned areas. Intuitively, it would be possible to achieve both high density and high accuracy if we process the camera pixel data and lidar range data simultaneously. The technique proposed in this section applies low-level, or early, fusion of camera and lidar, combining the strengths of the two sensors while mitigating their individual weaknesses. The main challenge in this task is finding a common representation for the two modalities which will allow for efficient training of the fusion model. We propose two solutions to the sparse depth input problem, the first using a linear interpolation filter, Section 4.4.1, while the second is a trainable approach which learns the common representation from the data, Section 4.4.2.

The main idea behind both methods is that the camera image contains fine information about object shapes and boundaries which can be used to guide the completion of a sparse lidar depth image. A depth image reconstructed in this manner has more accurate object boundaries and can therefore be used to easily match objects detected in the camera image in order to estimate their range. The main drawback in such an approach is the difficulty in extracting semantical information from images which usually requires learning based, analysis techniques. Both of the methods proposed in this section perform the image-guided depth completion using early-fusion convolutional neural networks that take an aligned RGB-D tensor as input and produce a single-channel, dense depth image as an output. The RGB channels come from a high resolution camera while the D channel is an input depth image from a projected lidar point cloud. The CNN models are trained using deep learning over large annotated datasets. Both networks consist of multi-resolution processing blocks which systematically reduce and encode the image structure. The encoded structure goes through a fully connected coding layers and, at the end, a series of up-sampling blocks are used to reconstruct the wanted information in its original resolution.

As other authors have previously reported [70], training CNNs on arrays containing empty values such as lidar depth images is extremely difficult. The convolution operators are not able to natively handle unobserved samples and this sparsity needs to be taken care of in pre-processing. There are several ways to deal with unobserved inputs: first, invalid inputs can be encoded using a default value e.g., zero depth. The problem with this approach is that the network must learn to distinguish between observed inputs and those being invalid. This is a difficult task as the number of possible binary patterns grows exponentially with the kernel size. Alternatively, the network can take an additional input in the form of a pixel sampling mask in the hope that it learns the correspondence between the observation mask and the inputs. Unfortunately, both variants struggle to learn robust representations from sparse inputs [70]. Finally, we can introduce domain knowledge which reduces the input sparsity by pre-processing the data with algorithms known to be effective. For example, simply filling-in the missing pixels with depth data from the nearest sample fixes the sparse input problem at very little computational cost.

4.4.1 Method 1: pre-processing using bi-linear interpolation

This method uses a similar CNN architecture as the monocular depth estimation method from the previous section, which is loosely inspired by the semantical segmentation U-Net [63] and the ResNet [62]. The input, however, consists of the rgb camera image and a dense (interpolated) depth image projected from a lidar point cloud. At the output end we expect a fully reconstructed dense depth image that is much more accurate than the interpolated one at input. This network also uses an encoder-decoder architecture which can be broken down into three distinct parts, namely a down-sampling, reasoning and up-sampling part.

Down-sampling is performed by series of convolution filters and max pooling layers. Each down-sampling block halves the spatial resolution and doubles the number

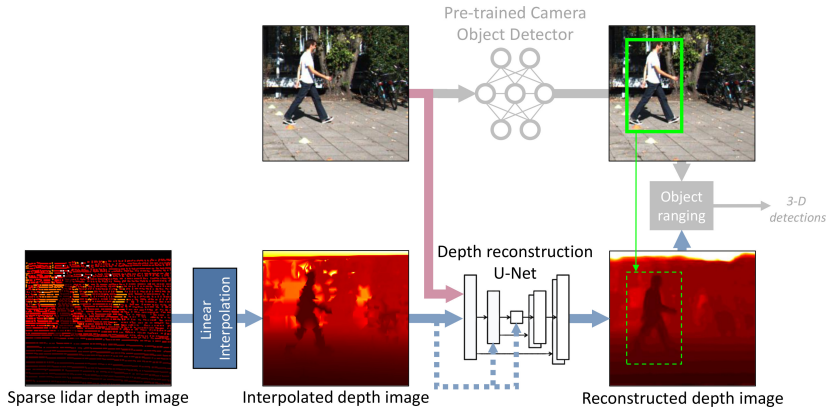


Figure 4.6: Block diagram of the first proposed early-fusion depth completion CNN as part of a camera-lidar object detector. The sparse depth input is first interpolated using a bi-linear filter.

of channels of the data structure. This ensures that the input image is processed at several spatial resolutions where increasingly more complex features are being extracted down the pipeline. In other words, at the start where the spatial resolution is high, the network computes a small set of features (16) and at the bottom block N, where spatial resolution is very low, the amount of features equals to $16 \cdot 2^N$.

The motivation behind this down-sampling structure is that as the image gets smaller in spatial resolution, we can extract an increasing amount of higher level concepts. Modeling low level features like edges, local color distributions or texture should take place at an early stage of the network while the image resolution is high, but in order to model high level visual cues such as perspective and horizon lines the network needs to operate on the entire image size, but not necessarily at the highest resolution. Max pooling is preferred over other pooling techniques as we experimentally found out that this operation requires less parameters, produces superior results and allows the network to converge much faster. Authors in [85] made extensive tests on various pooling operators and came to the same conclusion that max pooling is better suited at modeling translation-invariant features.

Reasoning about the structure of the scene is performed at the lower spatial resolution level of the network where the feature vector has a high feature depth. This last convolutional filter has a higher support of 5×5 and the number of feature channels is kept at a reasonable number of 256. We suspect that the activations of this 5×5 filter bank will correspond to higher concepts linking RGB to depth information as they are able to link image features to object class concepts in object detectors such as [24]. It is at this stage that the global scale or distance of the scene is being estimated.

The following up-sampling step needs to filter out the massive amount of channels into a single depth image, at the same time increasing the spatial resolution. For this task we use a series of up-sampling blocks, each doubling the spatial resolution of

the previous block. Up-sampling blocks concatenate the output of the previous, lower scale, block together with the output from the respective down-sampling block. We perform this concatenation in order to re-introduce part of the original feature channels that were computed during down-sampling.

After the channel maps are concatenated, we perform two additional convolutions. The first convolution mixes the information coming up-stream from the reasoning layer together with the leaked channels from the down-sampling block, and the second performs the actual up-sampling of the channels. These convolution blocks are designed to re-introduce the high-resolution features back into the output. We have experimented different modes of information leak between down-sampling and up-sampling blocks, where we also considered Residual and Linear Combination Layers. We found out that that simple concatenations followed by a mixing layer works best because this way the network can adapt its own optimal mixing protocol based on the data itself. At the end a Rectifying Linear Unit fixes the network outputs to positive depth values.

Ground truth is provided in the form of semi-sparse depth images produced by a high resolution 64 beam Velodyne LiDAR. We used the previously explained multi-lateral upsampling algorithm from [10] to produce accurate fully sampled dense depth images for the semi-sparse ground truth data. The network is trained using the stochastic gradient descent (SGD) by optimizing the L_2 loss between the network output $\widehat{D}$ and the ground truth depth image D:

$$L_2\left(D, \widehat{D}\right) = \sum_x \left(D(\mathbf{u}) - \widehat{D}(\mathbf{u})\right)^2, \tag{4.7}$$

where $D(\mathbf{u})$ is the depth value at pixel coordinate $\mathbf{u} = (u, v)$. The choice for this specific loss function was made in part due to its convenient capability for strongly penalizing large errors in the estimate and partly because most benchmarks in the literature measure the performance of depth completion by means of the root mean squared error (RMSE) which correlates with the L_2 loss function.

Additionally we experimented with the location of entry in the network for the sparse depth data. The sparse or low resolution input can be plugged in at a point in the network where the data structure has similarly low resolution, see dashed lines in Figure 4.6. One can argue that all inputs should enter the first stage of the network and the learning process itself can decide at which layer which information is extracted. However, due to practical reasons (network depth, machine precision, optimizer convergence, number of epochs, etc.) the point of entry at which the low-resolution input depth is fused with the camera image can have practical implications on the training speed. Thus, finding the optimal fusion point will enable the same model accuracy to be reached with less training. Inserting the sparse depth in the beginning creates more degrees of freedom for the network to adapt to the data. On the other hand, inserting the sparse depth in the middle of the network relieves the computational load of an additional high resolution channel at input.

We performed experimental evaluation of our network in an automotive context

where input data comes from sensors mounted on a moving vehicle through urban and rural environments. We followed the protocol in [59] and [60] using the KITTI tracking sequences for training and ground truth data. A total of 20 video sequences (8000 frames) are used for training, and a different set of 28 sequences (11000 frames) serves as the test set. We use the color RGB camera data at half resolution as input and we process the Velodyne HDL-64E point clouds using the method in [10] in order to get fully sampled depth ground truth data. For the first experiment, laid out in detail in Section 4.5.3, we establish a baseline using only RGB data as input, while for the second experiment, detailed in Section 4.5.4, we inject sparse depth images in the network in order to reinforce the RGB data.

4.4.2 Method 2: pre-processing using learnable morphological filters

The novelty of this method comes from the introduction of morphological layers before the contracting part of the U-Net which eliminates the sparsity in the input. The main idea is that the pre-processing step can also, with some constraints, be delegated to the neural network and be learned from training data. The proposed method adds a novel CNN block which replaces the bi-linear interpolation algorithm explained in Section 4.4.1. It has been shown by Ku et al. [67] that a simple yet efficient depth completion of sparse lidar depth images can be achieved by applying a sequence of morphological operations on the sparse input. In their experiments these authors show that a small set of finely tuned dilations and erosions is enough to reconstruct a high quality depth image. Additionally, the authors discovered that better performance is achieved when applying the morphological operations on the sparse disparity image rather than on the sparse depth image. This is done because of the nature of the morphological dilation operation in gray level images, where pixels with larger values are extended by the shape of the structuring element. In cases where an area to be dilated is completely filled with measurements, the resulting dilation will accentuate objects that are closer to the camera (lower depth, greater disparity), rather than the background. This result is more desirable since it is safer to assume that no object with size less than half of the structuring element will be completely lost by applying a dilation.

Since standard 2D convolution operations have difficulties in learning sparse data input problems [70, 86], the proposed method uses a trainable morphological filter block operating on the sparse disparity images, see Figure 4.7. The purpose of this morphological sub-network is to better learn an initial disparity image estimate which we then convert to depth and fuse with the image data. We approximate morphological dilation and erosion operations by utilizing the limit behavior of the Contraharmonic Mean Filter (CHM). These filters can be easily implemented in most contemporary deep learning frameworks through differentiable programming using standard convolutional layers and other arithmetical operators. In the later CNN layers, morphologically processed disparity, converted into depth, and RGB information are fused using standard U-Net architecture.

Morphological operators are the foundation of many image segmentation algorithms. Using so called "structuring elements" they represent fundamental non-linear operations which compute the minimum, maximum or the combination of both within the element support. Morphological operations are also invariant to translation and are strongly related to Minkowski addition. In the context of depth completion, it is of interest for the system to learn the shape and the operation type that fits best the data. However, due to the non-differentiable nature of minimum and maximum filtering, only few approaches have been found to succeed in the literature. Note that CNNs do support max-pooling operators with fixed support, but in this context we would like the network to learn the shape of the maximum operator i.e., structuring element. To this end, we find that the approximation of morphological operators by the contraharmonic mean (CHM) filter in [87] is the best founded technique which can easily be integrated in a deep learning framework. In this method, we also use the CHM to approximate our basic learnable morphological block.

Following the analysis in [87], [88] and [89], we use the same notation $I(\mathbf{u})$ to represent a 2-D real-valued image where the 2-D vector $\mathbf{u} : (u, v)$ represents the pixel coordinate vector in the image domain. A filter kernel w is any positive 2-D matrix $w : W \rightarrow \mathbb{R}_+$where W is the support of the filter. We approximate the contraharmonic mean filter function $\psi_w^k(I(\mathbf{u}), w)$ as the 2-D convolution of the image I^k, whose pixel values are raised to the power k, and a filter w representing the structuring element:

$$\psi_w^k(I(\mathbf{u}), w) = \frac{\left(I^{k+1} * w\right)(\mathbf{u})}{\left(I^k * w\right)(\mathbf{u})} = \frac{\int_{\mathbf{u}_i \in W} I^{k+1}(\mathbf{u}_i)\, w(\mathbf{u} - \mathbf{u}_i)\, d\mathbf{u}_i}{\int_{\mathbf{u}_i \in W} I^k(\mathbf{u}_i)\, w(\mathbf{u} - \mathbf{u}_i)\, d\mathbf{u}_i}. \tag{4.8}$$

The CHM filter can also be interpreted as the k-deformed convolution $\psi_w^k(I)(\mathbf{u}) \equiv (I * kw)(\mathbf{u})$ where the order k of the filter defines the desirable properties such as morphological erosion when $k \ll 0$, or morphological dilation when $k \gg 0$:

$$(I * kw)(\mathbf{u}) = \begin{cases} \frac{\left(I^{k+1} * w\right)(\mathbf{u})}{\left(I^k * w\right)(\mathbf{u})} & if\ k \in \mathbb{R} \\ \inf_{\mathbf{u}_i \in W} \left\{ I(\mathbf{u}_i) - \frac{1}{k} \log\left(w(\mathbf{u} - \mathbf{u}_i)\right) \right\} & \text{if } k \ll 0. \\ \sup_{\mathbf{u}_i \in W} \left\{ I(\mathbf{u}_i) + \frac{1}{k} \log\left(w(\mathbf{u} - \mathbf{u}_i)\right) \right\} & \text{if } k \gg 0 \end{cases} \tag{4.9}$$

Due to the larger exponent in the divisor in Eq. (4.9), when k is large the filter output depends mostly on the pixels with the largest values within the support region W, which in the limit case $k \rightarrow \infty$ equates to the supremum i.e., morphological dilation:

$$\lim_{k \rightarrow \infty} \psi^k(I)(\mathbf{u}) = \max_{\mathbf{u}_i \in W} \left(I(\mathbf{u} - \mathbf{u}_i)\right) \equiv \psi^\infty(I). \tag{4.10}$$

Otherwise, when k is sufficiently small, the CHM filter will tend to select the smallest valued pixels which in the limit case $k \rightarrow -\infty$ equates to the infimum i.e., the morphological erosion:

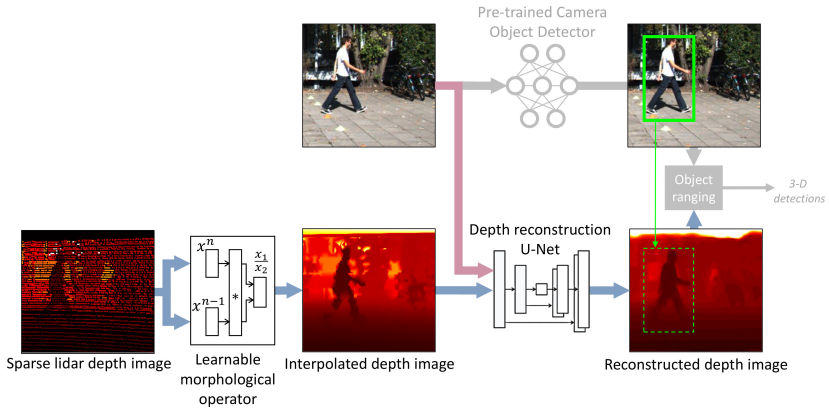


Figure 4.7: Block diagram of the second proposed early-fusion depth completion CNN as part of a camera-lidar object detector. The sparse depth input is first interpolated using a morphological filter with a learnable structuring element.

$$\lim_{k \to -\infty} \psi_k \left(\mathbf{u} \right) = \min_{\mathbf{u}_i \in W} \left(f \left(\mathbf{u} - \mathbf{u}_i \right) \right) \equiv \psi^{-\infty} \left(f \right). \tag{4.11}$$

In practice, the choice of k, and thus computing the derivative, will be limited by the computer number precision, but we found that a value of $k = 5$ produces the desired morphological filtering effect using single-precision floating point filter and pixel values in the range of $[0, 1]$. For a more detailed analysis of the filter properties and their proofs we refer the reader to Appendix B as well as the works of van Vliet [88] and Angulo [89].

We note again that $I \left(\mathbf{u} \right)$ indicates the pixel value at image position $\mathbf{u}$, I^k is a pixel-wise power operator of order k and $*$ indicates the 2-D valid convolution. In practice we used the MatConvNet [90] framework with the AutoNN implementation of automatic differentiation API which successfully computes the inference and back-propagation of the gradient values. Formally, the CHM filter is implemented using two convolution layers representing the denominator and numerator in Eq. (4.9), shown in the learnable morphological block on the diagram in Figure 4.7. The convolution layers share the same filters and biases and have the same learning rates. The learned structuring element can thus be visualized by taking the logarithm $\mathfrak{m} = \log \left(w \right)$.

Finally, the model is trained using supervised learning from dense depth images generated by registering multiple lidar point clouds. The gradient of the errors of the estimated depth is used to adjust the network model for more accurate reconstruction. Deviations from the ground truth can be quantified by a multitude of different metrics, such as absolute error, squared error, inverse absolute error, inverse squared error, absolute and squared relative error, percentage of outliers, etc. but for this method we decided to use a standard L-2 loss function. For training the entire network we employ the stochastic gradient descent by adaptive moment estimation (ADAM) tech-

nique, [91]. This method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. We set an initial learning rate $\left(\alpha = 10^{-5}\right)$ and two hyper-parameters: decay for the first moment vector $(\beta_1 = 0.9)$ and decay for the second moment vector $(\beta_2 = 0.999)$. More details on how this method was trained and evaluated using real-world data are given in the experiments and results Section 4.5.5.

4.5 Experiments and results

This section provides details about the experimental evaluation of the proposed depth reconstruction methods. Each proposed method will be evaluated in a separate experiment, tailored to compare it to relevant techniques from the literature which solve the reconstruction problem using the same modality. We will use the KITTI stereo and depth completion and depth prediction benchmarks which offer independent evaluation of the accuracy of predicted depth images in urban driving. In all experiments, the performance of the proposed methods are compared either to the state-of-the-art or to a comparable control method. This way we are able to analyze the experimental results and come to unbiased conclusions about the potential gains in accuracy and efficiency.

4.5.1 Single-sensor depth reconstruction

In order to evaluate performance of the methods proposed in Section 4.3.1 and Section 4.3.2, we followed the consensus protocol in the literature. The lidar-only depth completion algorithm was developed in the early phases of this research and due to lack of specialized depth estimation datasets for autonomous driving, at the time, it could only be tested using the KITTI Stereo 2012 and Stereo 2015 [32] benchmarks. These benchmarks rank algorithms according to the accuracy of their stereo disparity image estimates which is a related task to depth estimation. We were able to easily convert our reconstructed monocular depth images into disparity and test on the KITTI Stereo benchmark using the standard protocol. Using an exhaustive search through the provided raw data, Premebida et al. [66] have found a practical sub-set of the Stereo 2015 ground truth images that we will also use to quantitatively compare our dense depth map reconstructions against. The dataset consists of 100 original point clouds and 100 corresponding dense point clouds considered as ground truth. For fair comparison, the experiments in the following section will be performed using the same dataset as [66].

The monocular depth estimator was developed later on in the research and we were able tested its performance on the more recent KITTI tracking dataset which provided substantially more depth data which could be used for training and testing. Training and testing is performed using a content-independent split of 40 capture sequences, each at least 10 seconds long. Ground truth information is represented as a dense depth map computed from aggregating consecutive lidar point cloud scans (5 before and 5

Method	Outlier pixels
Proposed	**2.75%**
Premebida et al. [66]	3.35%
Bilateral filter	4.77%
IDW [66]	7.14%
KRI [66]	7.25%
Mean filter	7.56%

Table 4.1: Quantitative evaluation of reconstructed depth images using sparse LiDAR depth as input. KITTI Stereo 2015 dataset.

after the frame of interest) using the iterative closest point algorithm. Aggregated point clouds are projected onto the camera view and then all ambiguous image regions such as windows and fences are manually removed.

4.5.2 Lidar-only depth reconstruction

The lidar-only depth completion proposed in Section 4.3.1 serves as a first baseline for interpreting the accuracy of the proposed fusion methods. It is implemented as a filter with content-varying coefficients that processes sparse inputs. The filter input consists of a sparse depth image, a reflectance image and a segmentation image. Depth and reflectance are obtained directly from the raw lidar data while the segmentation image is computed by projecting and clustering the lidar height data onto the ground plane. The filter operates only on the missing pixel locations treating the missing values as unknowns which have no contribution in the output. In order to compute the segmentation image we first project the sparse (input) lidar point clouds on the ground plane and compute a local occupancy grid with cell size of 0.125x0.125m. Then, non-overlapping objects are segmentation by applying the connected components algorithm on the occupancy grid. The output of this step is an additional object label that is attached to each lidar point which is then projected onto the camera view to form the segmentation image, see the example in Figure 4.2.

The filtering and up-sampling works on local image patches with size 17x30 pixels optimized for the Velodyne HDL-64E lidar used to capture ground truth information in the KITTI stereo 2015 dataset. We used a grid-search technique to estimate the optimal values for the parameters $\{\alpha, \beta, \gamma, \delta\}$ from Eq. (4.6) on a content-independent training set (30-70 split) and found the optimal values to be $\{0.129, 0.011, 0.999, 56.23\}$ respectively. All of the steps in the proposed method were implemented in the Quasar programming language [46] and optimized for real-time execution on a CUDA device.

At the time of writing, we followed the evaluation protocol of the KITTI stereo 2015 benchmark which ranks algorithms according to their error in the number of disparity pixel outliers. Accuracy is measured by means of the metric $D - all[\%]$ which represents the percent of outlier image pixels averaged over all ground truth pixels.

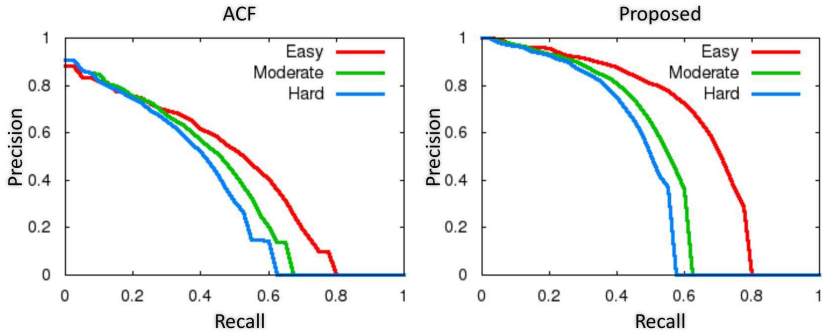


Figure 4.8: Precision-recall plots for detected pedestrians in KITTI object detection test set. Left: baseline ACF detector, right: re-trained ACF detector using additional depth channel.

An outlier pixel is one that has a disparity error of more than 3 pixels from the ground truth. To that end, we transformed all depth images to disparity images using a camera baseline of 0.537m and compared our results to the given ground truth. We conducted testing using the provided evaluation code in KITTI Stereo 2015 and obtained the accuracies which we compare to the work of [66]. The proposed semantically aware lidar depth upsampling method outperform all classical signal processing methods as well as the algorithm proposed in [66] by a significant margin, see Table 4.1. Several examples of the reconstructed depth images, reflectance images and segmentation images can be seen on the project page. [1]

Contrary to classical edge-preserving filters, this method relies on semantic information about object instances i.e., their shapes, to accurately predict the depth around borders of objects. The developed prototype reconstructs depth images using segmentation information extracted from the point cloud itself. The point cloud is segmented into disjoint, free standing objects whose projection on the image plane guides the proposed depth completion filter. However, since the lidar point cloud is sparse, its segmentation also results in a sparse semantic image. This semantical information can be easily substituted or reinforced by image-based segmentation which can offer better spatial resolution of object boundaries and will be evaluated in more detail in Section 4.5.4.

The proposed segmentation technique can cope well with nearly flat roads, however our projection on a 2D occupancy grid is sensitive to changing road gradient. In traffic situations where the vehicle is approaching a ramp or a steep incline parts of the road which are higher than the current surface will be segmented as separate objects. In such cases, a more robust ground plane segmentation algorithm than the one presented here is warranted. For example, ground plane fitting can applied over smaller spatial patches in order to compute a local linear estimates.

To show the potential effectiveness of our reconstructed depth images in an automotive application we trained an early-fusion camera-lidar pedestrian detection model

[1] `http://telin.ugent.be/~mdimitri/depth.html`

using the reconstructed depth as an additional channel. We expect that the addition of dense depth information to the available RGB data can boost both accuracy and robustness. At the time of writing this thesis, the KITTI object detection benchmark is one of the most relevant dataset for evaluation and ranking of object detection algorithms in the domain of autonomous and intelligent vehicles. It covers different urban scenarios, from university campus to downtown and residential areas. At the time when the experiment was conducted the ACF object detector [83] was one of the best performing algorithms that has a publicly available real-time implementation. We expanded the original ACF algorithm by adding our up-sampled depth maps to the processing pipeline and re-trained a pedestrian detection model. The pedestrian classifier we chose is a multi-stage cascade of weak decision trees. Once the models are trained, the detection of pedestrians is performed on the combined RGB and our reconstructed depth images.

In order to measure and compare our results to the literature, we uploaded the detected bounding boxes to the KITTI evaluation server. The proposed method achieved an average precision of 0.509, outperforming the RGB-D method "Fusion DPM" [65] which achieves average precision of 0.467, and significantly outperforming the baseline ACF method which achieves average precision of 0.398. This is a significant result since it shows that, in a controlled experiment, the reconstructed depth images by our method combined with the camera RGB information lead to better pedestrian detection than the competing Fusion DPM method which uses a Bilateral filter to upsample the sparse depth data. In Figure 4.8 we show the complete precision-recall curve for individual class sub-categories. On the left plot, the precision-recall curves for easy, moderate and hard to detect pedestrians of the baseline ACF detector [83] are shown, while the right plot shows the precision-recall curves for the re-trained detector.

By directly extending ACF to an additional depth channel we observe more than 10% improvement over the original camera-only ACF algorithm, and more than 4% improvement over the comparable RGB+D method in [65], see Table 4.2. These gains are most noticeable in image regions of poor light conditions such as shadows and generally poor visibility due to occlusion, appearance ambiguity, etc. In Figure 4.9 we present two such examples. In the first scene, two people (enlarged in the crop) are walking in a shaded area and their appearance closely matches that off the background, similarly, in the second scene the two people (enlarged in the crop) walking in a shaded area suffer from poor contrast. In both cases the proposed ACF operating on RGB+D data is able to detect all difficult objects without producing additional false positives. Furthermore, our proposed detector runs real-time (depth upsampling on the GPU and classification on the CPU) and is among the most accurate non-neural network based algorithm at the time of writing of the analysis.[2] Finally, the results of the proposed depth completion method has inspired the further development of occlusion-robust object detectors using early-fusion techniques and has led to the publication of several papers [79–81].

[2] https://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=2d

Figure 4.9: Examples of increased detection performance of the proposed early fusion ACF detector working on RGB and Depth images.

Method	Average precision
Proposed	**0.509**
Fusion DPM	0.467
ACF (baseline)	0.398

Table 4.2: Pedestrian detection results using reconstructed depth as additional input channel. KITTI dataset

4.5.3 Camera-only depth reconstruction

In this sub-section we will evaluate the accuracy of the depth images reconstructed by the monocular depth estimation algorithm described in Section 4.3.2. Our depth prediction CNN model was trained to predict depth images from RGB channels by minimizing the reconstruction error using annotated training data from the KITTI dataset. The ground truth comes from sparse depth images captured by the Velodyne HDL-64E lidar where each depth pixel is accurate within 0.03m. Note that during both training and evaluation only the sampled depth pixels are being evaluated by the loss function and the evaluation code while the CNN provides a fully sampled depth output.

We trained our CNN model parameters using stochastic gradient descent which was stopped after 20 epochs, a point after which no significant improvement in the loss function could be observed. Error is measured in the form of RMSE to the ground truth on a content-independent test set. Only pixels for which ground truth data is available are considered as per the standard KITTI evaluation protocol. We report an overall RMSE and also per sequence RMSE for every considered sequence. Our practical implementation using MatConvNet [90] is built on the Directed Acyclical Graph (DAG) model. Depth accuracy is maximized by minimizing the Mean Squared Error (MSE) between the output and the respective ground truth image. We used

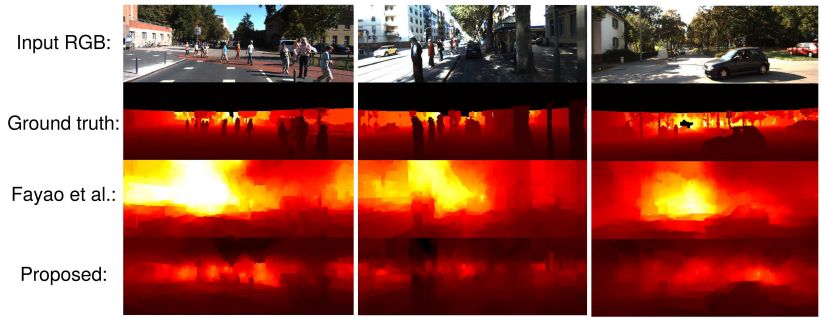


Figure 4.10: Examples of monocular depth estimation. Scene from the KITTI dataset.

a momentum value of 0.95 and weight decay of $2 \cdot 10^{-8}$ for all our experiments. Data augmentation was performed in the form of flipping the input images along the vertical axis. There are approximately 3.05 million free parameters in the network model. A single inference operation needs 0.52 TFLOPS which is greatly sped up using single precision floating point operations. In our experiments we measured an average of 7.4ms (135FPS) for a single inference of image with resolution 1280×384 on a GTX1070 GPU.

For fair comparison to the method of [60], we followed their super-pixel based approach combining the provided super-pixel segments and our reconstructed depth image. Using the super-pixel segmentation as a guide, this approach produces high quality depth images which seem to follow the natural contours of objects, however the computation time for the initial segmentation is prohibitively high for real-time applications. On Figure 4.10, we present three typical results of the proposed approach compared to the ground truth and the output from [60]. At the time of writing, the KITTI depth completion benchmark became available and submitted methods could no longer be compared using the obsolete stereo disparity error as in Section 4.5.2, but rather the absolute depth error measured as the root mean squared error in meters. For the remaining depth completion experiments in this thesis only the RMSE values will be reported.

In Table 4.3, we summarize the accuracies for the raw depth image reconstruction compared to several state of the art methods. We outperform all evaluated methods by a small margin, at the same time our network does not rely on any pre-processing such as the super-pixel segmentation applied in [60]. Finally, since we do not have ground truth data for all pixels in the image, not all of the differences visible in Figure 4.10 lead to the numerical results shown in Table 4.3. This is especially true for the top part of the image which is not covered by the lidar. Finally, due to the relatively high and ambiguous depth reconstruction errors the practical impact of this method on object detection in 3-D was not examined. However, monocular depth reconstruction is a topic of interest in systems that only employ cameras for environmental perception.

Method	RMSE [m]
Proposed CNN	**6.965**
Fayao et al. [60]	7.046
Eigen et al. [59]	7.156
Saxena et al. [92]	8.734

Table 4.3: Performance evaluation results of monocular depth estimation. KITTI dataset

4.5.4 Depth reconstruction by early camera-lidar fusion (pre-processing using bi-linear interpolation)

The following set of experiments experiments fuse an RGB camera image with a sparse depth image from lidar in order to estimate a more accurate dense depth image. For this experiment, we use the model based on the architecture described in Section 4.5.3 which we expect to outperform other methods due to the capacity to extract semantic information about object shapes from the RGB image. The CNN model is trained to perform early fusion where low resolution depth is introduced together with the input RGB image. The low resolution depth input is computed by projecting a lidar point cloud onto the camera view. As we saw earlier in this thesis, projecting a lidar point cloud onto a camera image generates a very sparse depth map. This information in addition to the RGB image is expected to yield much better depth reconstruction than the monocular methods in the previous experiment.

In addition to the best possible depth reconstruction accuracy using all available data, we want to investigate how much influence the input depth sparsity has on the reconstruction quality. The lidar used for the KITTI dataset experiments is a fairly modern and expensive sensor whose price limits the application in autonomous vehicles, so we are also interested how accurate our CNN model is if we feed a much more sparser depth input. To illustrate this, consider that when projecting a point cloud from the Velodyne HDL-64E lidar onto an image with resolution $(1244px \times 378px)$, around 3% of the image pixels contain depth values and the remaining 97% image area is sparse. Even with this sparse input, out lidar-only depth upsampling method was able to reconstruct accurate depth images which when fused with RGB images offered significant improvements in object detection.

In order to measure the influence of the level of sparsity of the input depth, we project only a small number of the already sparse lidar points, simulating an even sparser input from a low-end lidar with 3,5,9 and 17 laser beams. In each experiment, the beams are equally spaced along the elevation axis and the data density along the azimuth is kept at the original 2000 samples per revolution. Before feeding the sparse depth maps into the CNN we perform a bi-linear interpolation to fill in zero depth values. Table 4.4 gives an overview of the input density at which depth was introduced into the network.

After re-training each model, we evaluated the outputs of every sub-sampled set to a fully sampled ground truth and report the errors in terms of RMSE, Table 4.4. Our

first finding is that fusing the RGB image with even an extremely sparse (3 beam) lidar information results in reconstruction error of $5.415m$ over the test set which is significantly better than any monocular depth estimation methods, see Table 4.3. This shows that even minimal depth cues from lidar are beneficial in removing the depth ambiguity of monocular estimation. Second, by increasing the number of lidar beams in the input the proposed method produces depth images with drastically higher quality as seen in the RMSE of the reconstruction in Table 4.4. Lastly, we find that the improvement that the fusion CNN brings over a simple linear interpolation diminishes as the input depth density increases. This effect is because less contextual information is needed to reconstruct an already dense input depth image. Without the need of contextual information, simple techniques such as linear interpolation reach visually accurate depth images. For example: the density of the sparse input of a 17 beam lidar at an image resolution $1244px \times 378px$ is 0.76%. This input seems to contains enough information for reconstruction of smaller objects even using simple linear interpolation (RMSE=$2.877m$) as compared to the CNN output (RMSE=$2.676m$). Some typical results that were obtained in this experiment are shown on Figure 4.11. From visual inspection of these results a very important observation can be made: the proposed depth completion method is reaching a high level of quality with the data from current hardware.

When choosing a depth sensor for a specific application like object detection and ranging several factors should be taken into account. Obtaining a high quality depth information can be achieved either by using a range sensor with high sampling density, or by applying a reconstruction algorithm on data from a lower quality sensor. The former usually results in higher cost and power usage by the physical sensor while the later requires more computing capability, which also comes at an increased power usage and cost. With these experiments we ultimately want to show that the proposed depth completion method has diminishing benefits over using a higher quality sensor.

By minimizing the per-pixel MSE loss function we are able to quickly optimize the model parameters and compute depth images with both high accuracy and low number of outlier pixels. When sparse depth input data is not available, these experimental results show that our lightweight architecture can outperform the state of the art in monocular depth reconstruction in outdoor traffic scenarios. The method shows promising results when the input depth is fairly sparse, however once the input depth becomes denser (e.g., more than 1%) the accuracy gains (in terms of reconstruction RMSE) introduced by the network become marginal. The main drawback of this method is that it relies on hand-tuned pre-processing for handling the missing input values before the CNN is applied. This is a potential flaw in the depth completion process where we suspect that applying an end-to-end method will lead to bigger improvements in accuracy. This has in turn motivated the design of the following method, which removes the input depth sparsity using a a learnable pre-processing CNN block. It should be noted again that dealing with sparse input is not a trivial task for contemporary convolutional NN layers and great deal of care was taken to find a suitable sparsity suppressing operator.

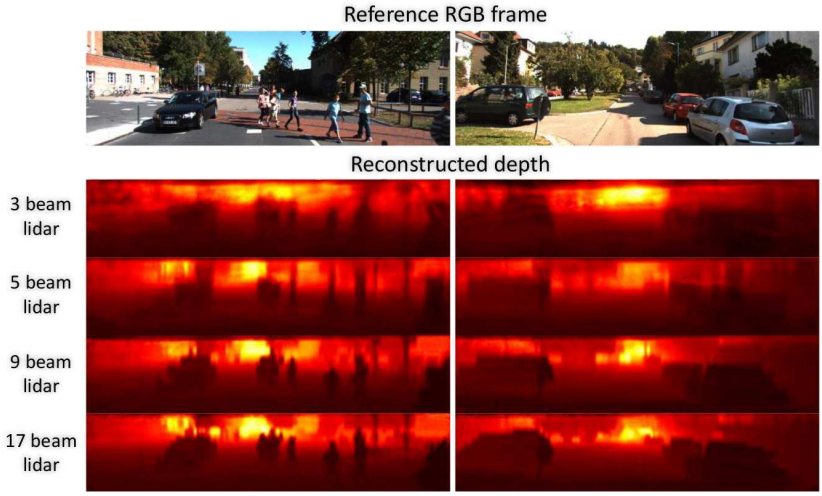


Figure 4.11: Examples of depth completion using RGB images and linearly interpolated depth as input. Scene from the KITTI dataset.

Input data	Input sparsity	Input RMSE [m]	Reconstruction RMSE [m]	Improvement
RGB	/	/	6.965	/
RGB + LiDAR (3 beams)	0.09%	9.246	5.415	41.4%
RGB + LiDAR (5 beams)	0.19%	6.061	4.209	30.6%
RGB + LiDAR (9 beams)	0.38%	3.827	3.349	12.5%
RGB + LiDAR (17 beams)	0.76%	2.877	2.676	7.0%

Table 4.4: Performance evaluation results for sparse depth completion using linear interpolation of the input. KITTI dataset.

4.5.5 Depth reconstruction by early camera-lidar fusion (pre-processing using learnable morphological filters)

The development and testing of the method proposed in Section 4.4.2 was enabled by the availability of the recently published KITTI depth completion[3] dataset [20, 32, 70]. It provided us with an excellent basis in the context of autonomous driving in terms of both rich annotated data as well as standardized evaluation protocol. This dataset consists of sequences captured by a stereo RGB camera pair as well as point clouds from the Velodyne HDL-64E LiDAR. Each point cloud is projected on a virtual camera image creating a sparse depth image. There is a total of 151 sequences with

[3] http://www.cvlibs.net/datasets/kitti/eval_depth.php?benchmark=depth_completion

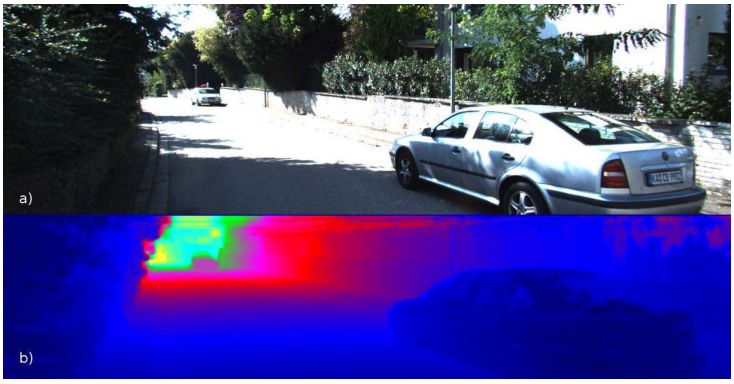


Figure 4.12: Typical use case scenario in an urban environment. Top: RGB camera frame, bottom: depth image completed by the proposed method.

93505 frames split into $\sim 92\%$ training and the remaining ones for model validation. Independent method evaluation is also provided by means of an on-line server which compares uploaded results to frames with ground truth data that is stored on the server.

Due to limitations in computing power we decided to use only a sub-sample of the training set for training. We noticed that, due to the relatively high sampling rate, most of the 93K frames in the original data contain temporally correlated information. Additionally, many of the sequences are recorded from a static vehicle and thus repeat the same content. Thus, in all our experiments we removed most of the static sequences and used every sixth frame from the remaining data. The training set sampled in this manner consists of ~ 4.3K frames of highly variable content. Input images are padded to a fixed resolution of 1280×384 pixels from which we randomly sample rectangular patches of size 96×96. Since our network uses 3 stage contraction, the lowest resolution of the input image inside the network is 12×12 with a channel depth of 256.

Learning of the optimal network parameters is done by presenting the network with batches of the labeled training set. After each inference, batch-average MSE is calculated from ground truth and the gradient is used to adjust the convolution filter parameters and biases. Each successive layer is updated by backpropagation using the chain rule. We employ the ADAM optimization method and, since we train using small patches of images, we train until convergence for ~ 200 epochs. During training we keep the hyper-parameters α, β_1 and β_2 fixed, but adaptively change the batch size, starting from 4 increasing to 64. We found that model convergence was fastest when training with collections of small random image patches as batches. Following every epoch, we perform validation using a small *sub-set* of the validation dataset.

We deployed our trained neural network on the 1000 test samples from the KITTI depth completion benchmark and submitted the results to the on-line evaluation server. The accuracy of our method in terms of iRMSE, iMAE, RMSE and MAE is independently measured by the KITTI on-line server and ranked along other methods. Results

Method	RMSE [m]	MAE [m]	Runtime	Publication
HMS_Net_v2	0.911	0.310	0.02 s / GPU	Anonymous
Sparse-to-Dense-2	0.954	0.288	0.04 s / GPU	Anonymous
HMS-Net	0.976	0.283	0.02 s / GPU	Anonymous
Morph-Net	**1.045**	**0.310**	**0.17 s / GPU**	**Proposed**
IP-Basic	1.288	0.302	0.01s / 1 core	Ku et al. [67]
ADNN	1.325	0.439	0.04 s / GPU	Anonymous
NN+CNN	1.419	0.416	0.02 s	Uhrig et al. [70]
SparseConvs	1.601	0.481	0.01 s	Uhrig et al. [70]
NadarayaW	1.852	0.416	0.05s / 1 core	Uhrig et al. [70]
SGDU	2.312	0.605	0.2s / 4 cores	Schneider et al. [68]
NiN CNN	2.378	0.685	0.01 s	Anonymous
NiN+Mask CNN	2.534	0.848	0.01 s / GPU	Anonymous

Table 4.5: Comparison of depth completion results on the KITTI depth completion benchmark.

are summarized on Table 4.5 and publicly available on the benchmark website. At the time of submission, we outperform classical methods such as [68] and [67], as well as the only published CNN method [70] in terms of RMSE error. Qualitatively, our method also better preserves object boundaries which is visible from the results shown on Figure 4.12. Using the RGB information in the contracting and expanding network architecture, we are able to effectively fill in missing object parts with the relevant depth information. This is especially noticeable in transparent objects such as house and car windows and glass displays. The inclusion of morphological layers makes the network flexible enough so that sparse data is handled in the initial layers, while the rest of the network is dedicated to better extracting contextual information. In terms of reconstruction accuracy, our method outperforms the state-of-the-art classical and neural network based approaches while operating on image batches which are concatenated to form the final depth image. The run-time for completing a single image of size 1280x384 pixels is on average 0.175s including the time needed for the fixed morphological pre-processing. We suspect that this increased processing time compared to the other methods is due to the size of our network and the use of non-conventional morphological blocks. However, the results as well as the runtime in Table 4.5 are self-reported and cannot be independently confirmed.

4.6 Conclusion and practical implications

In this chapter we proposed and evaluated several methods for reconstructing depth images from camera and lidar sensors. Depth completion from sparse inputs

such as lidar point clouds has traditionally been solved by local image processing that handles sparsity using fine-tuned filters. However, in cases where the level of sparsity varies spatially or parts of objects are completely missing, local processing is unable to accurately reconstruct depth information. Contextual information from the entire scene or parts of objects that are seen should be considered to better fill-in missing depth. The proposed methods were published as articles in the proceedings of the IEEE Intelligent Vehicles 2017 conference [10], the SPIE Optics and Photonics 2018 conference [11] and the IEEE Advanced Concepts for Intelligent Vision Systems 2018 conference [13].

The first contribution to depth completion was made based on classical signal processing using semantic information from the lidar point cloud itself. This method delivered dense depth images with a fidelity that outperformed the state of the art by almost 20%. In a secondary contribution, a monocular depth completion method was proposed to blindly reconstruct depth images from RGB pixel data. This method, based on the encoder-decoder convolutional neural network, slightly outperformed the state of the art on the KITTI depth completion dataset. However, the accuracy of camera-only depth prediction is relatively low compared to lidar-based methods and highly dependent on the scene content. Therefore, we deem that monocular depth prediction be considered with caution especially for safety critical applications where accurate ranging of objects is very important. The third and fourth contributions were depth reconstruction methods based on early fusion of camera and lidar. By combining the high pixel density of camera images and the high accuracy of sparse lidar point clouds, the two proposed methods were able to estimate depth images with very high accuracy. The main challenge in this research was the application of convolutional neural networks for the fusion of RGB images and sparse depth data from lidar. The literature has widely acknowledged that convolutional operators have difficulties when dealing with sparse inputs. Our two methods apply data pre-processing on the sparse depth inputs in order to expedite the CNN model training. The first proposed method applies bi-linear interpolation, while the second proposed method learns the pre-processing operator from the data.

The most accurate depth reconstruction results in terms of RMSE were obtained when using a camera-lidar fusion CNN with learnable morphological filters which outperformed the baseline depth completion method by more than 20%. From our qualitative analysis of reconstructed depth images we deem that the output from our most accurate method can be used to range even distant objects detected in the camera image. By using this approach the system does not necessarily need to perform object detection in 3-D, rather use state of the art camera object detection to find objects on the image plane and rely on accurate depth images for their ranging. However, we also showed that the reconstructed depth images offer very useful additional information source for training an object detector from scratch. We trained a proof of concept RGB-D object detector that showed significantly better robustness to illumination changes and occlusion, resulting in an increase of average precision by almost 30% over detection using RGB data alone.

Finally it is important to note that depth reconstruction has some practical limita-

tions that limit its deployment in real-world systems. Firstly, even the simplest of the proposed methods require additional CPU processing power to accurately reconstruct dense depth images and the methods using CNNs require significant GPU computing power in order to run real-time. In perception systems with limited power/compute capabilities it is prudent to use a lidar sensor and complete its sparse data using nearest neighbor or bi-linear interpolation. However, if a lidar sensor is not available, then monocular depth estimation should be performed using the camera images.

The main benefit of performing depth reconstruction independently of object detection is the opportunity to use off the shelf camera object detectors. At the time of writing, image-based object detection is a much more mature topic providing detection models that surpass the precision of detectors in any other sensor modality. This is because of the abundance of annotated image datasets, many of which do not have depth data. Therefore, given the more training data, it is obvious that image objects detectors can be trained to detect in 2-D much more robustly than in 3-D. Ranging 2-D detections is made trivial when we have an additional depth image, since the distance to a detection can be easily looked up from the depth values under its image bounding box.

5

Cooperative sensor fusion for object detection

5.1 Introduction

Object detection is a fundamental computer vision topic which has its applications in many modern camera based systems including autonomous vehicles. Detection, as opposed to *tracking* which uses temporal information, is the process of detecting the presence of objects and estimating their location using data captured over a short time period. Considering that most perception sensors operate by sequentially recording data into frames, detection is a discrete process done over short snapshots in time. In the literature this is also referred to as instantaneous, single-frame or frame-by-frame detection stemming from the short time integration of a single camera frame. As of the time of writing, object detection is most accurate in camera images because their high pixel density allows for better content interpretation over other sensors. However, there exist situations where cameras tend to under-perform (low light, glare, inclement weather, etc.) and detection is better done in other sensor modalities such as lidar or radar. A detection system that remains effective in all weather circumstances must therefore apply some sort of data fusion.

This chapter explores the details of multi-modal object detection using a combination of cameras, radar and lidar sensors, see Figure 5.1 for an example. The main goal is the detection (classification and ranging) of road users which comprises the computation of the belief in the existence as well as the belief in the location of objects in the scene given the multi-sensor observations. We will start by analyzing different

sensor fusion strategies designed to detect and range road users. By identifying key challenges in this topic and analyzing the state-of-the-art, this chapter will then propose several novel sensor fusion methods which can lead to better object detection and ranging. The main principle of operation of the proposed fusion methods is to use the strengths of one sensors in order to mitigate the weaknesses of the other, especially in border cases where single-sensor detection might be compromised. Therefore, the proposed cooperative fusion methods provide increased robustness to sensor failures while at the same time yielding higher detection accuracy than standard fusion techniques.

Within the confines of a probabilistic framework, we propose several cooperative techniques of feeding detection information from one sensor to another sensor and using this information as a prior to further improve object detection. The feedback system transmits only a small amount of information using low bandwidth sensor-sensor transmission channels. By applying these feedback loops, individual detectors detect more objects by focusing into regions of the scene indicated by the other sensors. At the same time, this sensor-sensor cooperation reduces the number of false alarms. The proposed fusion methods continue to effectively operate even in the case of a complete single-sensor failure or the corruption of a data stream.

The proposed cooperative fusion can be thought of as a system which uses awareness for the quality of its individual sensors in order to optimize the overall detection performance. It has the ability to adjust the operating points of individual sensors on the fly and moreover, vary the the operating point locally over the field of view. For example, reduce the detection threshold of the camera object detector in a region with low contrast in order to recall an object that is barely visible but detected by another senor. The change of the detection threshold is applied only locally, and only in the current frame. In Chapter 6 we will further extend this concept over time where confident tracking information is fed back to the detectors in order to improve the detection thresholds in regions where objects have been tracked in the past. The common motivation for the proposed methods is that measurements from multiple sensors are fused cooperatively, discarding as little data as possible while still retaining the robustness to sensor failures. Cooperative fusion offers the maximum robustness of late fusion, at a minimal loss of accuracy as compared to early fusion.

The remainder of the chapter is structured as follows, we start with an overview of the state-of-the-art in Section 5.2. Then, we explain the details of the proposed cooperative camera-lidar, camera-radar and camera-lidar-radar fusion architecture in Section 5.4, and we demonstrate how object detection can be improved in both camera as well as radar. In Section 5.6 we give practical tips for transforming the heterogeneous sensor data into a common representation. Finally, in Section 5.7 we present a detailed analysis of the experimental methodology, datasets and the results that were obtained in this study.

Figure 5.1: Visualization of camera, lidar and radar measurements in a typical autonomous driving use-case. 2-D detection of road users can be performed using the camera image (top) 3-D detection is done by lidar (middle) and radar (bottom). For reference, the images show also the scene depth measured by lidar (middle) and reflected radar signal (bottom).

5.2 Literature overview

5.2.1 Camera object detection

Object detection in images is one of the most studied topics in computer vision. The complete literature on object detection is too extensive to be fully covered. As an illustration, at the time of writing even the KITTI [32] object detection benchmark, which is highly specific to autonomous driving has more than 500 submitted methods for evaluation. Similarly, the COCO [93] dataset which organizes yearly object detection competitions has around 250 methods submitted only in 2020. In this section we will summarize the camera-based object detection literature through the most significant papers which made strides in accuracy, efficiency or both. Object detection papers can generally be split into two eras, the first being an era of hand crafted features and specialized classifiers, while the second is the era of artificial neural networks and deep learning.

Era of detectors based on hand-crafted features and classifiers

One of the pioneering methods in the field of real-time image object detection is the face detector described in the paper [94] by Viola and Jones. This paper describes a machine learning approach for visual object detection which is capable of processing images extremely rapidly and achieving high detection rates. The method uses the integral image representation to compute Haar-like features in constant time and applies a classifier tree trained using the Adaptive Boosting method which selects a small number of critical visual features from a larger set and yields extremely efficient classifiers. Individual classifier trees are connected in a cascade which can be viewed as an object specific focus-of-attention mechanism which unlike previous approaches provides statistical guarantees that discarded regions are unlikely to contain the object of interest. In the domain of face detection the system yields detection rates comparable to the best previous systems. Used in real-time applications, the detector runs at 15 frames per second without resorting to image differencing or skin color detection.

The seminal paper [95], show experimentally that grids of Histograms of Oriented Gradients (HOG) descriptors significantly outperform existing feature sets for the task of people detection. The authors study the influence of each stage of the computation on performance, concluding that fine-scale gradients, fine orientation binning, relatively coarse spatial binning, and high-quality local contrast normalization in overlapping descriptor blocks are all important for good results. The new approach gives near-perfect separation on the original MIT pedestrian database. The authors also developed their own, more challenging dataset containing over 1800 annotated images with a large range of pose variations and backgrounds. They argue that although the fixed-template-style detector has proven difficult to beat for fully visible pedestrians, humans are highly articulated and we believe that including a parts based model with a greater degree of local spatial invariance would help to improve the detection results in more general situations.

The last significant method of this era is the Deformable Part Models (DPM) detector [96]. This method is an object detection system based on mixtures of multi-scale deformable part models, able to represent highly variable object classes, achieving state-of-the-art results in the PASCAL object detection challenges. The authors propose to combine a margin-sensitive approach for data-mining hard negative examples with a formalism, which they call latent Support Vector Machines (SVM). A latent SVM is a reformulation of Multiple Instance SVM in terms of latent variables. A latent SVM is semi-convex, and the training problem becomes convex once latent information is specified for the positive examples. This leads to an iterative training algorithm that alternates between fixing latent values for positive examples and optimizing the latent SVM objective function.

Deep learning era: two stage detectors

Region-based convolutional neural networks or regions with CNN features (R-CNNs) are pioneering approaches that apply deep models to object detection. R-

CNN models first select several proposed regions from an image (for example, anchor boxes are one type of selection method) and then label their categories and predict the shape of bounding boxes by regression of their offsets from the anchors. In R-CNN [97], the input image is first divided into nearly two thousand region sections, and then a convolutional neural network is applied for each region, respectively. The size of the regions is calculated, and the correct region is inserted into the neural network. Training time is significantly greater compared to later methods such as YOLO because it classifies and creates bounding boxes individually, and a neural network is applied to one region at a time.

SPP-Net [98] is a convolutional neural architecture that employs Spatial Pyramid Pooling (SPP) to remove the fixed-size constraint of the network. Specifically, the authors of this method add an SPP layer on top of the last convolutional layer. The SPP layer pools the features and generates fixed-length outputs, which are then fed into the fully-connected layers (or other classifiers). Doing so the CNN performs some information aggregation at a deeper stage of its hierarchy (between convolutional layers and fully-connected layers) to avoid the need for cropping or warping at the beginning. This method avoids repeatedly computing the convolutional features. In processing test images, it is 20-100× faster than the R-CNN method, while achieving better or comparable accuracy on Pascal VOC 2007.

Fast R-CNN [99] was developed with the intention to cut down significantly on train time. While the original R-CNN independently computed the neural network features on each of as many as two thousand regions of interest, Fast R-CNN runs the neural network once on the whole image. This is very comparable to later approaches such as YOLO. At the end of the network is a novel method known as Region of Interest (ROI) Pooling, which slices out each Region of Interest from the network's output tensor, reshapes, and classifies it. This makes Fast R-CNN more accurate than the original R-CNN. Fast R-CNN trains the very deep VGG16 network 10× faster than R-CNN, is 200× faster at test-time, and achieves a higher mAP on PASCAL VOC 2012. Compared to SPPnet, Fast R-CNN trains VGG16 3× faster, tests 10× faster, and is more accurate.

Faster RCNN [100] builds upon the notion that all of the previous CNN methods suffer from bottleneck in the region proposal. The authors introduce a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. Their RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection. Faster RCNN further merges RPN and Fast R-CNN into a single network by sharing their convolutional features using the recently popular terminology of neural networks with "attention" mechanisms, the RPN component tells the unified network where to look. For the very deep VGG-16 model, the Faster RCNN detector has a frame rate of 5fps on a GPU, while achieving state-of-the-art object detection accuracy on PASCAL VOC 2007, 2012, and MS COCO datasets with only 300 proposals per image.

Mask R-CNN [101] is a significant improvement over the previous methods by

focusing on the task of object instance segmentation. This approach efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance. Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps. Moreover, Mask R-CNN is easy to generalize to other tasks, e.g., al lowing us to estimate human poses in the same framework. This method show top results in all three tracks of the COCO suite of challenges, including instance segmentation, bounding box object detection, and person key-point detection.

Feature Pyramid Networks (FPN) [102] uses the inherent multi-scale, pyramidal hierarchy of deep convolutional networks to construct feature pyramids with marginal extra cost. A top-down architecture with lateral connections is developed for building high-level semantic feature maps at all scales. This architecture shows significant improvement as a generic feature extractor in several applications. Using FPN in a basic Faster R-CNN system, this method achieves state-of-the-art single-model results on the COCO detection benchmark, surpassing all existing single-model entries including those from the COCO 2016 challenge winners. In addition, the method can run at 6 FPS on a GPU and thus is a practical and accurate solution to multi-scale object detection.

Lastly, G-RCNN [103] uses the recurrent convolution neural network (RCNN) architecture, inspired by abundant recurrent connections in the visual systems of animals. The critical element of RCNN is the recurrent convolutional layer (RCL), which incorporates recurrent connections between neurons in the standard convolutional layer. With increasing number of recurrent computations, the receptive fields of neurons in RCL expand unboundedly, which is inconsistent with biological facts. The paper limits the receptive fields of neurons by introducing gates to the recurrent connections. The gates control the amount of context information inputting to the neurons and the neurons' receptive fields therefore become adaptive. The resulting layer is called gated recurrent convolution layer (GRCL). Multiple GRCLs constitute a deep model called gated RCNN (GRCNN). The GRCNN was evaluated on several computer vision tasks including object recognition, scene text recognition and object detection, and obtained much better results than the RCNN. In addition, when combined with other adaptive receptive field techniques, the GRCNN demonstrated competitive performance to the state-of-the-art models on benchmark datasets for these tasks.

Deep learning era: single stage detectors

The popular You Only Look Once (YOLO) [23] object detector is a one-stage CNN able to detect objects of multiple categories. The convolutional neural network re-purposes classifiers and localizers to perform detection and applies the detection model to an image at multiple locations and scales. At the output, high scoring regions of the image are considered detections. As a single-stage detector, YOLO performs classification and bounding box regression in one step, making it much faster than

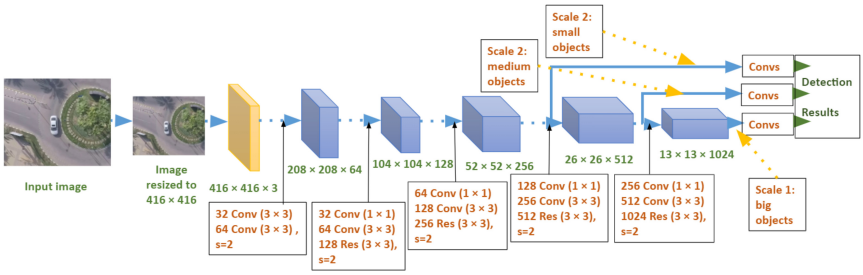


Figure 5.2: Visualization of the data structures and network architecture of a typical camera object detector.

most convolutional neural networks. For example, YOLO object detection is more than 1000x faster than R-CNN and 100x faster than Fast R-CNN.

Single-shot detector (SSD) [104] is a one-stage detector that can detect multiple classes of objects in images using a single deep neural network by discretizing the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. The object detector generates scores for the presence of each object category in each default box and adjusts the box to better fit the object shape. Also, the network combines predictions from multiple feature maps with different resolutions to handle objects of different sizes. The SSD detector is easy to train and integrate into software systems that require an object detection component. In comparison to other single-stage methods, SSD has much better accuracy, even with smaller input image sizes.

The authors of RetinaNet [105] discovered that the extreme foreground-background class imbalance encountered during training of dense detectors is the central cause for the limited accuracy of one-stage object detectors. They propose to address this class imbalance by reshaping the standard cross entropy loss such that it down-weights the loss assigned to well-classified examples. The novel Focal Loss focuses training on a sparse set of hard examples and prevents the vast number of easy negatives from overwhelming the detector during training. The CNN uses a feature pyramid architecture to efficiently detect objects at multiple scales. The results show that when trained with the focal loss, RetinaNet is able to match the speed of previous one-stage detectors while surpassing the accuracy of all existing state-of-the-art two-stage detectors.

YOLOv3 [24] outperforms existing state-of-the-art detectors such as DSSD513 [106] and RetinaNet [105] in multi-label classification by employing overlapping patterns for training, see Figure 5.2 for an overview of its architecture. Hence it can be used in more complex scenarios for object detection. Because of its multi-class prediction capabilities, YOLOv3 can be used for small object classification while it shows worse performance for detecting large or medium-sized objects. Owing to its good balance between accuracy and algorithmic complexity, YOLOv3 has been implemented in many deep learning frameworks. It is both easy to train as well as to deploy in the real world. *Throughout this thesis YOLOv3 serves as the object detec-*

tor of choice and as such will be re-used several times to demonstrate the proposed cooperative fusion concept.

The Scale-Aware Trident Network [107] performs Object Detection paying special attention to the challenge of scale variation. By investigating the effect of the size of receptive fields for scale variation in object detection the authors propose a novel CNN aiming to generate scale-specific feature maps with a uniform representational power. The method is based on a parallel multi-branch CNN architecture in which each branch shares the same transformation parameters but with different receptive fields. The model is trained in a scale-aware training scheme to specialize each branch by sampling object instances of proper scales for training. On the MS-COCO dataset it outperforms methods such as R-CNN and SNIPER [108] by a significant margin without any extra parameters and computations.

YOLOv4 [109] represents a significant improvement over the previous YOLO detectors in a way that it employs a set of universal features which are optimized for multiple computer vision tasks over multiple datasets. Specifically, the authors propose to use weighted-residual-connection blocks, cross-stage-partial-connection blocks, cross mini-batch normalization, self-adversarial-training and self-regularized MISH activations. The paper provides an extensive analysis and is an excellent resource on training modern object detectors. At the time of publication it offers a state-of-the-art detection performance which is faster and more accurate (in terms of AP50...95 and AP50 on MS-COCO) than all available alternative detectors. Moreover, the authors of YOLOv4 make sure that the proposed method can be trained and deployed using a GPU with 8-16GB memory which makes it usable in broad range of applications.

YOLOR [110] is a recently proposed, novel object detector which applies implicit and explicit knowledge to the model training. It can learn a general representation and complete multiple tasks through this general representation. Implicit knowledge is integrated into explicit knowledge through kernel space alignment, prediction refinement, and multi-task learning. Most significantly, YOLOR achieves greatly improved object detection performance results. Compared to other object detection methods on the MS-COCO dataset benchmark, YOLOR improves upon the state of the art in terms of average precision at the same inference speed. Compared with the Scaled-YOLOv4, the inference speed has been increased by 88%, making it the fastest real-time object detector available in 2021.

Finally, the recently proposed Swin Transformer V2 [111] uses principles from the natural language processing and applies them to the task of object detection. This method proposed three main improvements over the state of the art: first, a residual-post-norm method combined with cosine attention which improves training stability; second, a log-spaced continuous position bias method which effectively transfers models pre-trained using low-resolution images to downstream tasks with high-resolution inputs; and third, a self-supervised pre-training method, SimMIM [112], which reduces the needs of vast amounts of labeled images. Through these techniques, the authors of this paper successfully trained a 3 billion-parameter Swin Transformer V2 model, which is the largest dense vision model to date, and makes it capable of training with images of up to 1,536×1,536px resolution. It set new performance

records on 4 representative vision tasks, including ImageNet-V2 image classification, COCO object detection, ADE20K semantic segmentation, and Kinetics-400 video action classification. This method gives a glimpse into the future of camera-based object detection where the model backbones will be trained on variety of visual tasks employing self-supervised strategies that will enable the training of ever larger models using unlabeled data.

5.2.2 Lidar object detection

In contrast to images where pixels are regularly distributed on an image plane, point clouds are sparse and require additional transformations into representations that are more easily processed by existing feature extraction and classification tools. As we saw in the previous chapter, depth images offer a dense and compact representation, but range pixels contain 3-D information instead of RGB values and directly applying conventional convolutional networks on range images may not be an optimal solution. On the other hand, detection in autonomous driving scenarios generally has a requirement on real-time inference. Therefore, how to develop a model that could effectively handle point cloud or range image data while maintaining a high efficiency remains an open challenge to the research community.

Depending on how the lidar point cloud data is represented, object detection methods can be categorized into one of the following categories. Point-based 3-D object detectors propose diverse architectures to detect 3D objects directly from raw points. Grid-based 3-D object detectors rasterize the point cloud into discrete grid representations, i.e. voxels, pillars, and bird's-eye view (BEV) feature maps. Point-voxel based approaches use to a hybrid architecture that leverages both points and voxels for 3-D object detection. Finally, methods based on depth images rely on the fact that the lidar point cloud is scanned from a single point and can be projected onto a 2-D surface (a depth image) without loss of information.

Point cloud-based 3-D object detectors

Methods in this category firstly pass point clouds through a point-based backbone network, in which the points are gradually sampled and features are learned by point cloud operators. 3-D bounding boxes are then predicted based on the down-sampled points and features. There are two basic components of a point-based 3-D object detector: point cloud sampling and feature learning. The PointNet and more recently PointNet++ methods [113, 114] which introduces a hierarchical neural network and applies PointNet recursively on a nested partitioning of the input point set, exploit metric space distances, and are able to learn local features with increasing contextual scales. With further observation that point sets are usually sampled with varying densities, which results in greatly decreased performance for networks trained on uniform densities, the authors propose novel set learning layers to adaptively combine features from multiple scales. Experiments show that the PointNet++ network is able to learn deep point set features efficiently and robustly. In particular, results significantly bet-

ter than state-of-the-art have been obtained on challenging benchmarks of 3-D point clouds. Similarly, [115] uses two stages of furthest point selection strategies to progressively down-sample input point cloud and generate 3-D proposals from the down-sampled points. Extensive experiments on the 3-D detection benchmark of KITTI dataset show that this architecture outperforms state-of-the-art methods with remarkable margins by using only point cloud as input. Feature learning is usually performed using some sort of set abstraction on the point cloud. Specifically, ball query with a pre-defined radius can yield context points which are ten aggregated through multi-layer perceptrons and max-pool operators to obtain new features. Other methods use graph operators [116], attention operators [117] as well as transformers [118].

The representation power of the point- based detectors is mainly restricted by two factors: the number of context points and the context radius adopted in feature learning. Increasing the number of context points will gain more representation power but at the cost of increasing much memory consumption. These two factors have to be determined carefully to balance efficacy and efficiency of detection models. Point cloud sampling is a bottleneck in inference time for most point-based methods. Furthest point sampling and its variants can attain a more uniform sampling result by sequentially selecting the furthest point from the existing point set. Nevertheless, furthest point sampling is intrinsically a sequential algorithm and can not become highly-parallel. Thus furthest point sampling is normally time-consuming and not ready for real-time detection

Grid-based 3-D object detectors

Grid-based methods transform the point cloud data into rasterized format (voxels or pillars). Then they apply conventional 2-D convolutional neural networks or 3-D sparse neural networks to extract features from the grids. Finally, 3-D objects can be detected from the BEV grid cells. There are two basic components in grid-based detectors: grid-based representations and grid-based neural networks. There are 3 major types of grid representations: voxels, pillars, and BEV feature maps which are related to occupancy maps described in Section 3.4. Voxels are 3-D cubes and contain the point density inside voxel cells. Since point clouds are sparsely distributed, most voxel cells in the 3-D space are empty and contain no points. In practical applications, only those non-empty voxels are stored and utilized for feature extraction.

The VoxelNet method described in [119] removes the need of manual feature engineering for 3-D point clouds proposing a generic 3-D detection network that unifies feature extraction and bounding box prediction into a single stage, end-to-end trainable deep network. The proposed method divides a point cloud into equally spaced 3-D voxels and transforms a group of points within each voxel into a unified feature representation through the newly introduced voxel feature encoding (VFE) layer. In this way, the point cloud is encoded as a descriptive volumetric representation, which is then connected to a RPN to generate detections. Experiments on the KITTI car detection benchmark show that VoxelNet outperforms the state-of-the-art lidar based 3-D detection methods by a large margin. Furthermore, this network learns an ef-

fective discriminative representation of objects with various geometries, leading to encouraging results in 3-D detection of pedestrians and cyclists, based on only lidar.

Pillars are special voxels in which the voxel size is unlimited in the vertical direction. Pillar features can be aggregated from points through a [113] and then scattered back to construct a 2-D BEV image for feature extraction. The seminal method PointPilalrs described in [120] introduces the pillar representation and uses a novel encoder which utilizes PointNets to learn a representation of point clouds organized in vertical columns. While the encoded features can be used with any standard 2-D convolutional detection architecture, this method further proposes a lean downstream network. Extensive experimentation shows that PointPillars outperforms previous encoders with respect to both speed and accuracy by a large margin. Despite only using lidar, the full detection pipeline significantly outperforms the state of the art, even among fusion methods, with respect to both the 3-D and bird's eye view KITTI benchmarks.

Bird's-eye view feature map based methods use a dense 2-D representation, where each pixel corresponds to a specific region and encodes the points information in this region. BEV feature maps can be obtained from voxels and pillars by projecting the 3-D features into the bird's-eye view, or they can be directly computed from raw point clouds by summarizing points statistics within the pixel region. The commonly-used statistics include binary occupancy and the height and density of local point cloud, recall the lidar sensor model shown in Figure 3.7. There are 2 major types of grid-based networks: 2-D convolutional neural networks for BEV feature maps and pillars, and 3-D sparse neural networks for voxels. Conventional 2-D convolutional neural networks can be applied upon the BEV feature map to detect 3-D objects from the bird's-eye view. In most works, the 2-D network architectures are generally adapted from those successful designs in 2-D object detection, like the ResNet [62]. 3-D sparse convolutional neural networks are based on two specialized 3-D convolutional operators: sparse convolutions and sub-manifold convolutions, which can efficiently conduct 3-D convolutions only on those non-empty voxels. The method SECOND [121] is a seminal work that implements these two sparse operators with GPU-based hash tables and builds a sparse convolutional network to extract 3-D voxel features. The authors of this method also introduce a new form of angle loss regression to improve the orientation estimation performance and a new data augmentation approach that can enhance the convergence speed and performance. The proposed network produces state-of-the-art results on the KITTI 3-D object detection benchmarks while maintaining a fast inference speed.

In contrast to the 2-D representations like BEV feature maps and pillars, voxels contain additional 3-D information. In addition, deep voxel features can be learned through a 3-D sparse network. However, a 3-D neural network brings additional time and memory cost. It's important to note that this network architecture has been applied in numerous other works, and is the foundation of the CenterPoint [51] lidar detector which we use extensively throughout this thesis.

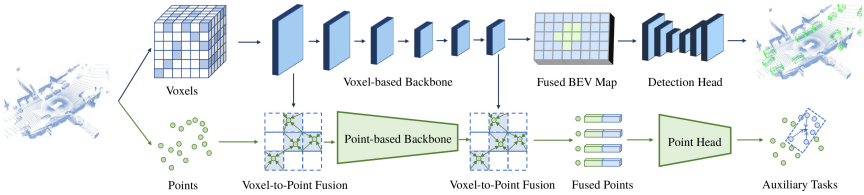


Figure 5.3: Visualization of the data structures and network architecture of a typical point-voxel 3-D object detector.

Point-voxel-based 3-D object detectors

Point-voxel based approaches use a hybrid architecture that leverages both points and voxels for 3-D object detection. Those methods can be divided into two categories: the single-stage and two-stage detection frameworks. Single-stage point-voxel based 3-D object detectors try to connect the features of points and voxels with the point-to-voxel and voxel-to-point transform in the backbone networks, see the diagram in Figure 5.3 which demonstrates a generic architecture of a method in this category. Points contain the original, fine-grained geometric information and voxels are efficient for computation, and combining them together in the feature extraction stage naturally benefits from both two representations. Methods such as the Point-Voxel CNN detector [122] and [123] are typical example detector that perform point-voxel feature fusion in their backbones. The diagram shown in Figure 5.3 depicts a generic network architecture of one such method.

PVCNN [122] represents the 3-D input data in points to reduce the memory consumption, while performing the convolutions in voxels to reduce the irregular, sparse data access and improve the locality. The PVCNN model is both memory and computation efficient. Evaluated on semantic and part segmentation datasets, it achieves much higher accuracy than the voxel-based baseline with 10x GPU memory reduction. It also outperforms the state-of-the-art point-based models with 7x measured speedup on average. Remarkably, the narrower version of PVCNN achieves 2x speedup over PointNet (an extremely efficient model) on part and scene segmentation benchmarks with much higher accuracy.

In [123] authors propose Sparse Point-Voxel Convolution (SPVConv), a lightweight 3D module that equips the vanilla Sparse Convolution with the high-resolution point-based branch. With negligible overhead, this point-based branch is able to preserve the fine details even from large outdoor scenes. To explore the spectrum of efficient 3-D models, this method uses a flexible architecture design space based on SPVConv. The authors performed a 3-D Neural Architecture Search to search the optimal network architecture over this diverse design space efficiently and effectively. Experimental results validate that the resulting proposed model is fast and accurate: it outperforms the state-of-the-art ranking 1st on the competitive Semantic KITTI leaderboard. It also achieves 8x computation reduction and 3x measured speedup over the competition with higher accuracy. Finally, the model was transferred to perform 3-D object detec-

tion where it achieved consistent improvements over the one-stage detection baseline on KITTI.

Two-stage point-voxel based 3D object detectors resort to different data representations for different detection stages. Specifically, at the first stage, they employ a voxel-based detection framework to generate a set of 3-D object proposals. At the second stage, key-points are firstly sampled from the input point cloud, and then the 3-D proposals are further refined from the key-points through novel point operators. The most representative method of this category is PV-RCNN [124], which adopts SECOND [121] as the first-stage detector, and the ROI-grid pooling operator is proposed for the second-stage refinement. The following works try to improve the second-stage head with novel modules and operators.

Range-based 3D object detection

Range-based methods address the detection problem from two aspects: designing new models and operators that are tailored for range images, and selecting suitable views for detection. Since range images are 2-D representations like RGB images, range-based 3-D object detectors can naturally borrow the models in 2-D object detection to handle range images. Most notably, LaserNet [125] leverages the deep layer aggregation network to obtain multi-scale features and detect 3D objects from range images. This method is a fully convolutional network to predict a multi-modal distribution over 3-D boxes for each point and then it efficiently fuses these distributions to generate a prediction for each object. Experiments show that modeling each detection as a distribution rather than a single deterministic box leads to better overall detection performance. Benchmark results show that this approach has significantly lower runtime than other recent detectors and that it achieves state-of-the-art performance when compared on a large dataset that has enough data to overcome the challenges of training on the range view.

Some works resort to novel operators to effectively extract features from range pixels, including range dilated convolutions, graph operators, and meta-kernel convolutions. Depth images are captured and projected on a spherical projection. It has been a natural solution for many range-based approaches to detect 3-D objects directly from this view. Nevertheless, detection from the range view will inevitably suffer from the occlusion and scale-variation issues brought by the spherical projection. To circumvent these issues, many methods have been working on leveraging other views for predicting 3-D objects, e.g. the cylindrical view leveraged in [126], and a combination of the range-view, bird's-eye view (BEV), and point-view adopted in RangeIoUDet [127]. RangeIoUDet is an efficient and effective 3-D object detection framework that uses the range image as input. Benefiting from the dense representation of the range image, RangeIoUDet is entirely constructed based on 2-D convolutions, making it possible to have a fast inference speed. This model learns pointwise features from the range image, which is then passed to a region proposal network for predicting 3-D bounding boxes. The authors optimized the pointwise feature and the 3-D box via the point-based IoU and box-based IoU supervision, respectively. The

point-based IoU supervision is proposed to make the network better learn the implicit 3-D information encoded in the range image. The 3-D Hybrid GIoU loss is introduced to generate high-quality boxes while providing an accurate quality evaluation. Through the point-based IoU and the box-based IoU, RangeIoUDet outperforms all single-stage models on the KITTI dataset, while running at 45 FPS for inference.

Compared to bird's-eye view detection, detection from the range view is vulnerable to occlusion and scale variation. Hence, the combination of feature extraction from the range view and object detection from the bird's eye view become the most practical solution to range-based 3-D object detection.

5.2.3 Radar object detection

When working with automotive radar as the main ranging sensor, detection needs to take into account the specifics and limitations of the radar. At the time of writing this thesis, most commercially available automotive radars provide very limited angular resolution in the elevation plane. For example, more often than not, radar sensing is performed in only one elevation plane and much of the vertical information of the scene is lost. A typical radar data sample consists of signal strength sampled on a 3-D range, azimuth and Doppler grid, often referred to as a radar cube. Radar, however, has several distinct advantages over other sensors in real-world applications. First, the cost of the sensor is much lower than lidar, second, radar signals are less susceptible to extreme weather conditions, then, radar generally offers larger detection range than lidar, and last, radar provides additional velocity measurements. Most of the classical radar perception literature focuses on radar super-resolution techniques [128] and depth sensing at an object level irrespective from the resolution of the imaging camera. Techniques such as late fusion can then be employed to match ranged objects by the radar to detected regions of interest in the camera image. The following paragraph lists some notable radar analysis methods for detecting moving objects, mainly road users in automotive context.

Classical methods such as the Constant False Alarm Rate (CFAR) [129] can perform accurate moving target detection in radar signal by determining whether a target exists in the clutter or noise background. Existing CFAR detection procedures are commonly performed using sliding windows in the radar signal, from which the parameters of the hypothesized model are estimated, and the data available in the reference window are employed to compute the decision threshold. CFAR offers reliable detection of moving targets, however, further processing on these detections is needed for classification. In [130], the authors present one typical example of detecting the motion of people using hand crafted range and Doppler features. Their method is built on analyzing the radar waveform design i.e., the expected characteristics of the return signal given the known radiation pattern and the most likely object motion features. A comparative study [131] analyses the performance between random forests and LSTM, see [132], network for classification of cars, pedestrians, groups, bikes and trucks. Backed by large scale experiments, their conclusion is that the difference between LSTM and random forest is surprisingly small (0.884 vs. 0.871 F1 score).

They also found that the performance of the LSTM network, in particular, is highly sensitive to the amount of training samples, a motivation which drives us towards training with large dataset and weak supervision.

A semantical radar grid building algorithm is presented in [133]. The authors rely on 4 radars, whose observations are first registered, to classify regions containing cars and other objects. In this paper a shallow fully convolutional neural network was used to classify input occupancy grid cells into classes of objects.a In contrast to this approach, we're interested in instantaneous road user detection and operate on a practical, short time window. In [134], the authors present a semi-supervised deep radar detector operating on 3-D dense radar data. The method splits the input dimensions by applying two independent CNNs that process in the range-Doppler and elevation-azimuth dimensions respectively. A weakness of this method is that radar dimensions are only combined in late feature space, thus the potential of inter-dimensional dependencies is lost from the start. Since we are only interested in detecting VRUs, we can discard any unnecessary Doppler data, for example velocities too high for a pedestrian, by pre-processing steps, and retain a complete 3-D radar space as input. The authors of the paper [135] propose a CNN object detection and 3-D estimation based on the U-Net architecture. They use a coupled Radar and Camera sensor to prepare a set of training samples for a radar CNN which determines the presence or absence of a car in the radar signal. This method uses a 3D network architecture, where the input tensor consists of radar range, velocity and receiver channel information, and the output consists of 3 layers: a binary probability of occupancy and two image plane coordinates. The main drawback of this method is that its training protocol is limited to cars in the image plane and the authors do not provide extensive evaluation for cluttered environments.

In [136] the authors propose a hybrid radar detection system consisting of initial target detection by classical processing followed by radar target classification network. The network operates on cropped range-azimuth-Doppler radar tensors extracted around the initial radar targets and outputs a class label and score for the categories car, person and cyclist. Finally, they apply clustering in order to group similarly classified targets into complete objects. This method was evaluated on a real-world dataset using automatically annotated ground truth from matched camera and stereo-depth sensors. Even though the authors report promising results, this method relies on single time integration radar cubes, therefore overlooking important micro-Doppler cues needed for classifying VRUs.

Following a comprehensive analysis of applying deep learning to radar signals presented in [137] the authors propose a deep learning method for vehicle detection in bird's eye view using Range-Azimuth-Doppler tensors. Interestingly, the method doesn't truly work with the full 3-D radar data, rather it computes three image-like inputs by collapsing each radar dimension respectively. This paper also proposes a semi-automated annotation framework based on a 64-beam lidar sensor, however manual human correction was needed to obtain ground truth. By controlling for various factors in their deep learning architecture the authors came to the following conclusions: best performance is achieved by operating in the native polar coordinates and applying

a Cartesian transformation on the latent features, second: incorporating Doppler information using their proposed model has marginal benefits and third: exploiting the temporal dependency with a LSTM cell has marginal benefits . A potential weakness of this method is the recurring loss of micro-Doppler information due to the collapse of dimensions in the pre-processing.

Very recently, 4-D imaging radar based on the phased-array principle has promised to bring cheap depth sensing to the automotive market. Using a limited number of receive and transmit antennas, this type of radar can produce a sparse range image spanning over range, azimuth and elevation. Reconstructing the pixel-level information that remains missing between the radar beams can then be performed by any other sparse depth completion techniques. This technology, however, remained unavailable at the time of writing and will not be the subject of further analysis.

5.2.4 Fusion object detectors

Within the application domain of advanced driver assistance systems, a vast number of late sensor fusion techniques exist. The vast majority of autonomous driving prototypes employ a combination of cameras and lidar/radar sensors. Since lidar-based detection methods perform much better than camera-based methods for 3-D detection, the state-of-the-art approaches are mainly based on lidar-based 3-D object detectors and try to incorporate image information into different stages of a lidar detection pipeline. Combining the two modalities together inevitably brings additional computational overhead and inference time latency. Therefore, how to efficiently fuse the multi-modal information remains an open challenge.

Early fusion 3-D object detectors

Early-fusion based methods aim to incorporate the knowledge from images into point cloud before they are fed into a lidar-based detection pipeline. Hence the early-fusion frameworks are generally built in a sequential manner: 2-D detection or segmentation networks are firstly employed to extract knowledge from images, and then the image knowledge is passed to point cloud, and finally the enhanced point cloud is fed to a lidar-based 3-D object detector. Based on the fusion types, the early-fusion methods can be divided into two categories: region-level knowledge fusion and point-level knowledge fusion.

Region-level fusion methods aim to leverage knowledge from images to narrow down the object candidate regions in 3-D point cloud. Specifically, an image is first passed through a 2-D object detector to generate 2-D bounding boxes, and then the 2-D boxes are extruded into 3-D viewing frustums. The 3-D viewing frustums are applied on lidar point cloud to reduce the searching space. Finally, only the selected point cloud regions are fed into a LiDAR detector for 3-D object detection.

F-PointNet [138] first proposes this fusion mechanism for 3-D object detection from RGB-D data in both indoor and outdoor scenes. A key challenge of this approach

is how to efficiently localize objects in point clouds of large-scale scenes (region proposal). Instead of solely relying on 3-D proposals, this method leverages both mature 2-D object detectors and advanced 3-D deep learning for object localization, achieving efficiency as well as high recall for even small objects. Benefited from learning directly in raw point clouds, the method is also able to precisely estimate 3-D bounding boxes even under strong occlusion or with very sparse points. Evaluated on KITTI and SUN RGB-D 3D detection benchmarks, F-PointNet outperforms the state of the art by remarkable margins while having real-time capability.

Frustum ConvNet [139] uses 2-D region proposals in an RGB image and generates a sequence of frustums for each region proposal. Then it uses the obtained frustums to group local points. F-ConvNet aggregates point-wise features as frustum-level feature vectors, and arrays these feature vectors as a feature map for use of its subsequent component of fully convolutional network (FCN), which spatially fuses frustum-level features and supports an end-to-end and continuous estimation of oriented boxes in the 3-D space. The authors also propose component variants of F-ConvNet, including an FCN variant that extracts multi-resolution frustum features, and a refined use of F-ConvNet over a reduced 3-D space. Ablation studies verify the efficacy of these component variants. F-ConvNet assumes no prior knowledge of the working 3-D environment and is thus dataset-agnostic. F-ConvNet outperforms all existing methods on SUN-RGBD, and claims to outperforms all published works on the KITTI benchmark.

A maritime object detection and fusion method is proposed in [140]. The method is based on proposal fusion of multiple sensors such as infrared camera, RGB cameras, radar and lidar. Their framework first applies the selective search method on RGB image data to extract possible candidate proposals that likely contain the objects of interest. Then it uses the information from other sensors in order to reduce the number of generated proposals by selective search and find more dense proposals. The final set of proposals is organized by considering the overlap between each two data modalities. Each initial proposal by selective search is assumed as a final proposal if it is overlapped (Intersection over Union $IOU > \alpha$) by at least one of the neighboring sensor proposals. Finally, the objects within the final proposals are classified by a Convolutional Neural Network (CNN) into multiple types.

Point-level fusion methods aim to augment input point cloud with image features. The augmented point cloud is then fed into a lidar detector to attain a better detection result. The most prominent example for method in this category is PointPainting [141]. This method leverages image-based semantic segmentation to augment point clouds. Specifically, an image is passed through a segmentation network to obtain pixel-wise semantic labels, and then the semantic labels are attached to the 3-D points by point-to-pixel projection. Finally, the points with semantic labels are fed into a lidar-based 3-D object detector. This approach has later been taken by many other authors to propose very effective early-fusion object detection. The work in [142] proposes an approach to seamlessly fuse RGB sensors into lidar-based 3-D recognition. This Multi-modal Virtual Point (MVP) approach takes a set of 2-D detections to generate dense 3-D virtual points to augment an otherwise sparse 3-D point

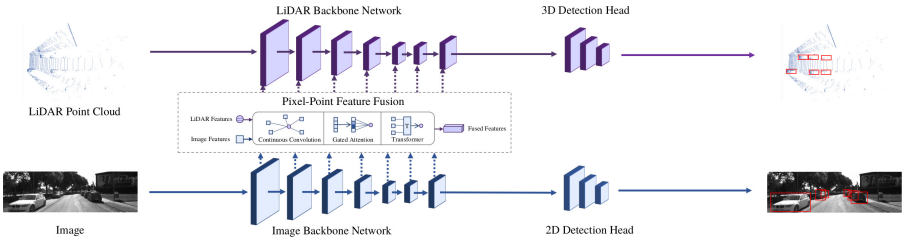


Figure 5.4: Visualization of the data structures and network architecture of a typical intermediate fusion camera-lidar detector.

cloud. These virtual points naturally integrate into any standard lidar-based 3-D detector along with regular lidar measurements. The resulting multi-modal detector is simple and effective. Experimental results on the large-scale nuScenes dataset show that our framework improves a strong CenterPoint baseline by a significant 6.6 mAP, and outperforms competing fusion approaches. We use MVP as a baseline to compare against the performance of the proposed method in this chapter, details in Section 5.7.2.

Early-fusion methods generally perform multi-modal fusion and 3-D object detection in a sequential manner, which brings additional inference latency. Given the fact that the fusion step generally requires a complicated 2-D object detection or semantic segmentation network, the time cost brought by multi-modal fusion is normally non-negligible. Hence, how to perform multi-modal fusion efficiently at the early stage has become a critical challenge. Another important aspect of early fusion methods is their inability to cope with sensor failures. As most neural networks are trained on the assumption of a steady flow of camera and lidar data, even soft failures where the camera information is degraded can cause domain shift and drastically reduce the performance of the fusion model.

Intermediate fusion 3-D object detectors

Intermediate fusion based methods try to fuse image and range features at the intermediate stages of a lidar-based 3-D object detector, e.g. in backbone networks, at the proposal generation stage, or at the ROI refinement stage. These methods can also be classified according to the fusion stages. Many contributions have been made to progressively fuse image and lidar features in the backbone networks. In those methods, pixel-to-point correspondences are firstly established by camera-to-lidar transform, and then with the pixel-to-point correspondences, features from a lidar backbone can be fused with features from an image backbone through diverse fusion operators. The multi-modal fusion can be conducted in the intermediate layers of a grid-based detection backbone, with novel fusion operators such as continuous convolutions [143], hybrid voxel feature encoding [144], and Transformer [145], see the diagram in Figure 5.4 which depicts a generic architecture of a method in this category. The multi-

modal fusion can also be conducted only at the output feature maps of backbone networks, with fusion modules and operators including gated attention [146], unified object queries [147], BEV pooling [148], learnable alignments [149], point-to-ray fusion [150], and Transformer [151]. In addition to the fusion in grid-based backbones, there also exist some papers incorporating image information into the point-based detection backbones such as [152].

Intermediate fusion in proposal generation or the ROI head conducts multi-modal feature fusion at the proposal generation and ROI refinement stage. In these methods, 3-D object proposals are first generated from a lidar detector, and then the 3-D proposals are projected into multiple views, i.e. the image view and bird's-eye view, to crop features from the image and ldiar backbone respectively. Finally, the cropped image and lidar features are fused in an ROI head to predict parameters for each 3D object.

The Multi-View 3-D (MCV3D) detector proposed in [86] is one such intermediate sensor fusion framework that takes both lidar point cloud and RGB images as input and predicts oriented 3-D bounding boxes. This method encodes the sparse 3-D point cloud with a compact multi-view representation. The network is composed of two sub-networks: one for 3-D object proposal generation and another for multi-view feature fusion. The proposal network generates 3-D candidate boxes efficiently from the bird's eye view representation of 3-D point cloud. The design is a deep fusion scheme which combines region-wise features from multiple views and enables interactions between intermediate layers of different paths. Experiments on the challenging KITTI benchmark show that the proposed approach outperforms the state-of-the-art by around 25% and 30% AP on the tasks of 3-D localization and 3-D detection.

The AVOD detector [153] is another notable intermediate fusion example where the proposed neural network architecture uses lidar point clouds and RGB images to generate features that are shared by two sub-networks: a region proposal network (RPN) and a second stage detector network. The proposed RPN uses a novel architecture capable of performing multi-modal feature fusion on high resolution feature maps to generate reliable 3-D object proposals for multiple object classes in road scenes. Using these proposals, the second stage detection network performs accurate oriented 3-D bounding box regression and category classification to predict the extents, orientation, and classification of objects in 3-D space. This architecture is shown to produce state of the art results on the KITTI 3-D object detection benchmark while running in real time with a low memory footprint, making it a suitable candidate for deployment on autonomous vehicles.

In [154] the authors propose an improvement to the YOLO [23] image object detection algorithm for detection of poorly lit road users using decision level fusion and feedback from lidar depth. They increase the granularity of the YOLO anchor grid in regions where people are detected with low confidence scores. This way, the improved YOLO algorithm can try twice to detect the target at a certain distance according to the characteristic of dim pedestrians and non-motor vehicles. Thus, it can reduce the missing rate of the target and output a more comprehensive scene model and ensure the safe driving of vehicles. This method compensates for a weakness of the original YOLO algorithm where a predefined raster grid is used for region proposal.

The intermediate methods encourage deeper integration of multi-modal representations and yield 3-D boxes of high quality. Nevertheless, camera and lidar features are intrinsically heterogeneous and come from different viewpoints, so there still exist some problems on the fusion mechanisms and view alignments. Moreover, due to the entanglement of the modalities in the deeper layers of the network, intermediate fusion methods too suffer from the effects of domain shift in situations of sensor failures.

Late fusion 3-D object detectors

Late fusion methods perform information fusion at the detection level i.e., the output 2-D and 3-D bounding boxes from camera and lidar detectors. In these methods, object detection with camera and lidar sensor can be conducted in parallel, and the output 2-D and 3-D boxes are fused to yield more accurate 3-D detection results. Late fusion is traditionally performed using statistical models and information theory which provides rules for the optimal combination of detection information from multiple sources. As such, late fusion methods are relatively mature, offering improved detection performance over single-modality detectors while having maximum resilience to sensor failures. Nonetheless, there are challenges in late fusion, especially in the part of aligning the detection data between the modalities. The literature on late fusion is saturated with approaches whose organization along the many categories is outside of the scope of this thesis. We direct the reader to the ProFusion2 project in the paper [155] as well as the survey papers presented in [156, 157] for more details on late fusion.

Notably, a recent method explained in [158], proposes a learning-based late fusion approach using a novel Camera-Lidar Object Candidates (CLOCs) fusion network. CLOCs fusion provides a low-complexity multi-modal fusion framework that significantly improves the performance of single-modality detectors. CLOCs operates on the combined output candidates before Non-Maximum Suppression (NMS) of any 2-D and any 3-D detector, and is trained to leverage their geometric and semantic consistencies to produce more accurate final 3-D and 2-D detection results. The experimental evaluation on the challenging KITTI object detection benchmark, including 3-D and bird's eye view metrics, shows significant improvements, especially at long distance, over the state-of-the-art fusion based methods. At time of submission, CLOCs ranks the highest among all the fusion-based methods in the official KITTI leaderboard.

The late-fusion based approaches focus on the instance-level aggregation and perform multi-modal fusion only on the outputs of different modalities, which avoids complicated interactions on the intermediate features or on the input point cloud. Hence these methods are much more efficient compared to other approaches. However, without resorting to deep features from camera and lidar sensors, these methods fail to integrate rich semantic information of different modalities, which limits the potentials of this category of methods.

Cooperative 3-D object detection

Cooperative object detection in the literature often refers to the process of object detection from a network of spatially diverse sensors distributed around the environment [159]. Methods such as this apply either early or late fusion principles, with the only difference being that the sensors are not collocated. This is in contrast to our definition of cooperative fusion, where we allow sensors with overlapping fields of view to share feedback information with each other, regardless whether they sit closer or further apart.

In [160] authors address the problem of huge volume of raw sensor data required to transfer between autonomous vehicles in a cooperative sensing system. They argue that it is practically infeasible to exchange raw data among vehicles, which would cause severe bottlenecks in existing network infrastructures. To reduce network traffic, a feature map based data sharing mechanism is proposed for 3-D object detection on autonomous vehicles. They propose a mechanism that uses important features received from other vehicles that could significantly improve the current vehicle's object detection performance. The proposed Cooperative Spatial Feature Fusion (CoFF) enables a vehicle (referred to as the receiver) to effectively utilize the supplementary information provided by another vehicle (referred to as the sender), and weighs the sender's feature map in the regions where its own feature map has a hard time detecting objects. The authors use two strategies to selectively fuse features of the sender and receiver in the overlapping areas. The first strategy uses the *maxout* function which takes the maximum of the feature maps of the two sensors, while the second strategy (Information-based Spatial Feature Fusion) takes the maximum along every channel of the features independently.

This cooperative method assumes that the feature maps from the sender and receiver are similar to each other i.e., they originate from the same type of sensor/detector, which limits the potential of the method. We argue that this method has a weakness that it uses an ad hoc metric (L-2 distance) to estimate the information gains from the two feature maps which, in the case of CNNs can be potentially misleading. This is because CNN feature maps can be quite sensitive to perturbations where small differences of the input lead to large differences in the feature values and the effect of the maximum operator is unpredictable. Moreover, by communicating feature maps this method still transmits a significant portion of information between the vehicles. In our cooperative fusion methods we transfer only high-level information between the sensors, and use metrics with well understood behavior to improve the detection rates in ambiguous situations.

A physics-based cooperative sensor fusion between visible and infra-red cameras for moving object detection is proposed in [161]. This algorithm uses collocated sensors and automatically adapts to the environmental changes that affect sensor measurements. The adaptation is done through a cooperative co-evolutionary algorithm that fuses the scene contextual and statistical information through a physics-based method. The method assumes static cameras where observations carry the contextual information used to build statistical (mixture of Gaussian) background model. The authors

show that the proposed fusion model adapted to various illumination conditions and is suitable for detection under variety of environmental conditions.

The idea proposed in this paper pertains to similar principles used in our cooperative fusion method i.e., by exchanging contextual information. In our work, the cooperating sensors create a model of the scene (which we call feedback line) where the presence of objects of interest is strongly indicated. This information serves as context for other sensors and enables them to adapt to the local conditions of the scene. Whereas in [161] use fixed sensor locations and low-level physics modeling of the environment, we apply the concept on moving sensors, building a contextual model of the scene using the same detection information that the sensors are tasked to produce.

In [162] authors propose a cooperative ranging-imaging detector based on lidar and camera for the task of road obstacle detection. The cooperation of the sensors refers to the process of region selection in the camera image which is made more effective by employing semantic information from the lidar. A lidar point cloud is assumed to contain both the ground plane and all objects of interest. After a ground plane removal and density-based segmentation step, 3-D objects are projected on the camera image plane to form ROIs. The authors use heuristics to fit rectangular masks to various objects such as cars, trucks, pedestrians, etc. The resulting image masks are found to visually match with the image content. However it is not clear how these masks can be used for classifying specific targets and whether there is a performance improvement.

The literature on truly cooperative fusion methods is scarce with most of the recent publications claiming to be cooperative being some variation of an intermediate fusion CNN. In the following, we propose an object detection system using heterogeneous sensors and independent object detectors, and perform cooperative sensor fusion at the decision level. All of the sensors are positioned on the same vehicle and we use the detection confidence of individual object detectors to rank the detections and only communicate confident ones between sensors. Sensor-sensor cooperation is thus aimed at improving the detection rates from the point of view of the vehicle without using any external data.

5.3 Cooperative multi-sensor object detection architecture

This section of the thesis deals with the problem of instantaneous object detection and localization in 3-D. We will assume that the reference coordinate system is fixed to the center of mass of the vehicle and the coordinate transform between the sensors and this center of mass is known. To achieve 3-D object detection, we will employ a cluster (or an array) of heterogeneous forward-looking sensors. Even though the proposed concepts focus on a forward-looking sensor array, they can be easily extended to a sensor setup of multiple cameras and range sensors, for example: a camera/radar in

one of each corners of the car covering an area of 360° in non-overlapping fields of view.

In order to be integrated in a larger vehicle control system, the environmental perception needs to meet the following requirements: it needs to have improved detection precision under all operating conditions; it needs to have redundancy i.e., to remain in operation if one of the sensors fails; and finally, it needs to operate in real-time. Due to these requirements, the proposed perception method uses individual object detectors trained for each sensor modality, akin to late fusion. The information extracted by the multiple object detectors is then matched in a common medium (usually the camera image plane) and communicated to an object tracker for temporal integration. If one of the sensor fails, then this late-fusion architecture can continue to operate effectively without the need for re-calibration.

At the architecture level, the proposed method extends late-fusion by allowing cross-sensor information exchange during detection. The sensors communicate with each-other through feedback lines which consist of prior detection information in a sensor-agnostic format. Practically, each sensor makes a short selection of very confident detections and computes the locations on the ground plane $\mathbf{x}$ where it thinks an object of interest is likely to be present, see diagram in Figure 5.5. Other sensors can then tap into this information and use it to make better decision in regions with poor signal quality. Each sensor remains, however, fully capable of detecting without the need of any feedback information if none is provided. Thus, the proposed cooperative fusion method retains the robustness benefits of late-fusion, and offers increased precision in ambiguous situations. It is important to note that the feedback line can also be populated by information from the object tracker, see diagram in Figure 1.1. The effect of this additional temporal information will be analyzed in more detail when we lay out the tracking details in the next chapter.

Prior to going into the specifics of the suggested solutions, it is crucial to briefly discuss the requirement for matching observed objects across several sensor modalities. Detection fusion needs to evaluate the likelihood of a hypothesis given detection evidence from multiple sensors. We assumed that the existence of an object is separate from its location so that we can model the two as separate, semi-independent concepts which allows for the better understanding and easier interpretation of the models. Moreover, optimizing for the existence and location of an object step-by-step avoids unnecessary localization computations for unlikely hypotheses which reduces the algorithmic complexity. The system assumes that within a region $\Omega(\mathbf{x}, \mathbf{g})$ the object existence is constant and thus evaluates a single object existence hypothesis H_1 over all positions and shapes in $\Omega(\mathbf{x}, \mathbf{g})$ from all observational evidence captured within this region. If the probability of presence is high enough, only then we look for a position within the region $\Omega(\mathbf{x}, \mathbf{g})$, which has the highest probability of presence at that location. Then, for regions where the existence hypothesis is significantly more likely than the null-hypothesis H_0, we test various positions and shapes.

For the object existence, it is critical to evaluate the log-likelihood ratio using the K sensor activation functions $\sum_{k=0}^{K-1} \mathrm{llr}^{(k)}(a^{(k)}(\mathbf{x}))$ within the region $\Omega(\mathbf{x}, \mathbf{g})$ around each ground plane position $\mathbf{x}$ (and possibly for each object shape $\mathbf{g}$). Similarly, for the

belief in the object position, it is critical to evaluate product of the K sensor models:

$$\prod_{k=0}^{K-1} p_{U^{(0)},...,U^{(K-1)}|H,X}\left(\mathbf{u}^{(0)},...,\mathbf{u}^{(K-1)}|H_1,\mathbf{x}\right), \tag{5.1}$$

where $\mathbf{u}^{(k)}$ is the closest detection within the region $\Omega(\mathbf{x},\mathbf{g})$ that comes from sensor k. Computing the belief of object existence is more critical for the task of instantaneous object detection where we need to make a decision whether to continue analyzing this region in the next time instance, while computing the exact location is critical in object tracking where the integration of detections over time can significantly improve the position estimate. These choices are of course motivated by the limitations in computing power and memory of practical perception systems, where it is not possible to keep track of all detections, however unlikely they are to exist. In practice, instantaneous object detection operates at a certain precision/recall point and communicates only confident detections to the tracking system.

Finding the variables that optimize these two beliefs, case 1: H_1 or H_0, which maximizes the road user presence Eq. (2.11), and case 2: the specific $\mathbf{x}$ which maximizes the road user position Eq. (2.16), is not straight forward because of two important reasons. First: each sensor's output $\mathbf{u}^{(k)}$ is defined in sensor-specific coordinates and we need to apply a transform in order to estimate the models at an arbitrary position on the ground plane $a^{(k)}(\mathbf{x})$. Second: even with the correct transform it might not always be possible to evaluate the activation function at an arbitrary position $\mathbf{x}$, but rather only at finite regions $a(\mathbf{u})$ generated by the detector non-maximum suppression.

The proposed cooperative feedback mechanism consists of a feedback line i.e., a sensor-agnostic scene descriptor that indicates areas of the scene where objects can be detected with high degree of confidence. The feedback line is populated by regions $\Omega(\mathbf{x},\mathbf{g})$ of the scene where a sensor is highly confident of object existence: $\frac{p_{H|a^{(k)}}\left(H_1|a^{(k)}(\mathbf{u},\mathbf{s})\right)}{p_{H|a^{(k)}}\left(H_0|a^{(k)}(\mathbf{u},\mathbf{s})\right)} \gg 1$. Depending on the implementation, as we will see later, these regions can be either a list of confident object detections or can be aggregated using an accumulator array. The feedback of confident detection regions is communicated to all cooperating sensors which then use this information to improve their potentially weak detection by adjusting the parameters such as threshold for weak detections near $\Omega(\mathbf{x},\mathbf{g})$.

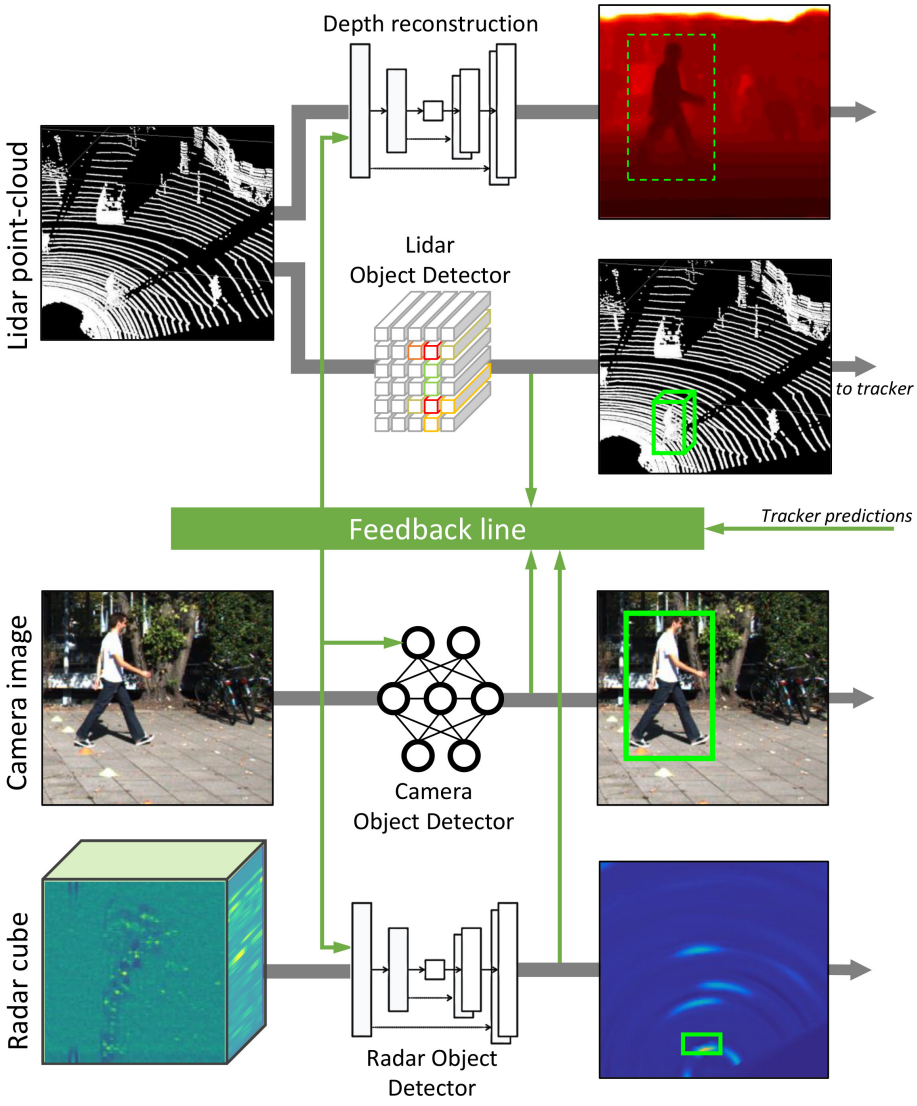


Figure 5.5: General cooperative fusion architecture. The feedback line is populated with confident detection information in a sensor-agnostic format which can then be used as an optional input by the sensors.

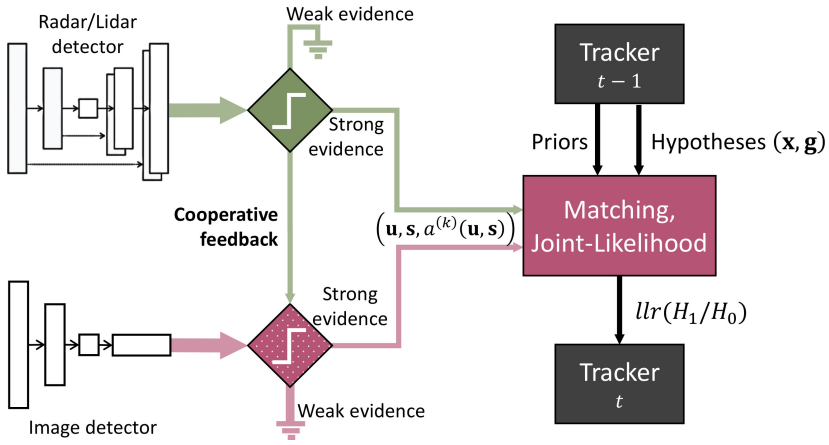


Figure 5.6:
Block diagram of the proposed Radar/Lidar→camera feedback mechanism for recovering weak camera detections which are in close proximity of strong Radar/Lidar detections. The hypothesis generation mechanism shown in the diagram comes from temporal tracking which is explained in the next chapter.

5.4 Camera object detection with Radar/Lidar feedback

In this section we propose a cooperative sensor fusion method for instantaneous object detection based on camera, radar and/or lidar. The sensors have overlapping fields of view meaning that each object can potentially be seen by each sensor. Standard single-sensor object detection operates by analyzing the input sensor signal and comparing the activations at the output to a detection threshold q. The working point of the detector is determined by the detection threshold, which is constant and unique to each sensor. If the k^{th}-sensor activation $\left(a^{(k)}\left(\mathbf{u},\mathbf{s}\right)\right)$ around the region where the object is located $(\Omega(\mathbf{x},\mathbf{g}))$ is above the k^{th}-sensor detection threshold $q^{(k)}$, then the k^{th}-object detector makes the decision that the object is detected, and otherwise the object is not detected.

The fusion of the multi-sensor evidence is performed by analyzing each region $\Omega(\mathbf{x},\mathbf{g})$ of the scene and finding the closest matches: detected $\left(\mathbf{u}^{(k)},\mathbf{s}^{(k)}\right)$ from each sensor having activations that satisfy: $a^{(k)}\left(\mathbf{u},\mathbf{s}\right) \geq q^{(k)}$. The joint belief in object existence within the region given the closest matching sensor evidence is computed as the joint log-likelihood ratio: $\ln \frac{p_{H|a^{(0)},\ldots,a^{(K-1)}}\left(H_1|a^{(0)}(\mathbf{u},\mathbf{s}),\ldots,a^{(K-1)}(\mathbf{u},\mathbf{s})\right)}{p_{H|a^{(0)},\ldots,a^{(K-1)}}\left(H_0|a^{(0)}(\mathbf{u},\mathbf{s}),\ldots,a^{(K-1)}(\mathbf{u},\mathbf{s})\right)}$. Also, note that in regions where some or all of the sensors do not detect an object, we use $\ln\frac{1-P_D^{(k)}}{1-P_{FP}^{(k)}}$, based on the the prior probability for detection P_D and false positives P_{FP} which are derived offline from the parameters of the detector. This indicates that the

confidence of missed detections is not the same as the actual confidence and is instead thought to be an average across the dataset.

Before calculating the joint log-likelihood ratio, the proposed cooperative fusion algorithm applies an additional step of reasoning based on the previously discussed findings that the assumed LLR of missed detections does not always accurately reflect the belief that an object is present, particularly in regions with compromised viewing $\Omega(\mathbf{x}, \mathbf{g})$. The core idea is that, before fusing, we want to improve the camera detection using cues about the nature/context of the region from other, more accurate sensors. We will do so by making the operating point of the camera adapt locally to the context of the scene, thus the working point q becomes a function of the spatial parameters of the scene: $q\,(\mathbf{x}, \mathbf{g})$. The notion is that by collecting information about the context and nature of each region in the scene from other, more precise sensors, we may locally decrease the camera detection threshold, recover missed detections and be more accurate about their LLR.

For instance, if the lidar is certain that an item is present in a dimly illuminated area of the scene, we may utilize this knowledge to guide the camera's object recognition process and increase the camera's chance of recalling an object in that area. Only in such indicated regions, lowering the camera detection threshold yields evidence whose log-likelihood ratio is more accurate than the camera detection priors P_D and P_{FP}. The approach results in more detections by the camera in regions of poor viewing, which eventually leads to a better joint log-likelihood ratio after fusion. Radar/lidar targets that match well with image bounding boxes are likely to belong to the same object. False positives from complementary sensor modalities, on the other hand, frequently do not arise in the other sensors. This is due to the varying modes of operation of the sensors, which causes clutter in different areas of the scene. For example, radar clutter caused by secondary reflections is not present in the image and false detections in the image due to low light are not present in the radar measurements. Therefore, adjusting the detection parameters using cues from other sensors can potentially increase the detection recall while keeping the number of false positives constant.

The proposed algorithm for cooperative fusion operates as follows. Strong or confident detections from radar/lidar define the regions of the scene $\Omega'(\mathbf{x}, \mathbf{g})$ where it is highly indicative that an object is present (indicated with thin green line in Figure 5.6). In contrast, weak camera detections identify regions of the scene $\Omega"(\mathbf{x}, \mathbf{g})$ where it is possible that an object is there but belief in its presence is low due to poor viewing circumstances. Before the threshold is applied to camera activations, we interface with the object detector and adjust the threshold depending on how well each detection agrees with the strong evidence from radar and lidar. (indicated as Cooperative Feedback in Figure 5.6). Practically, we match Radar/Lidar targets and image bounding boxes by projecting only the *confident* detections by the Radar/Lidar onto the image plane using the same transforms explained in detail in Section 5.6. Recall that we don't want to reduce the detection threshold in regions where the other sensors do not detect anything, which would result in more false positives. The same idea may be used to various types of sensors without sacrificing generality as long as we are certain

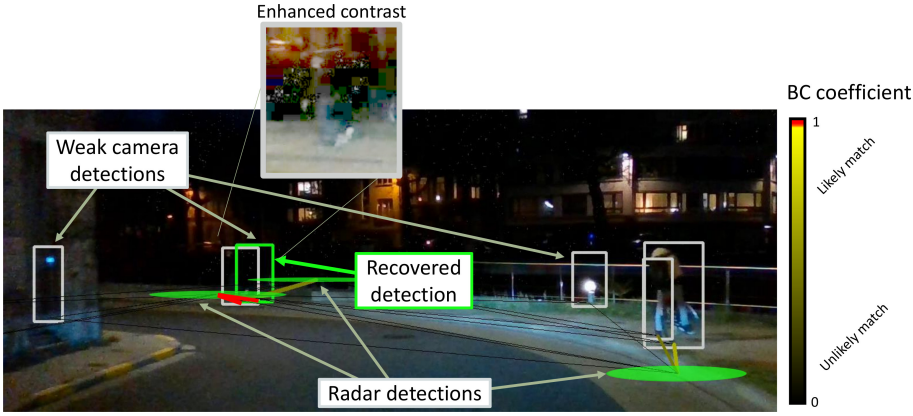


Figure 5.7: Example of a nighttime scene where the weak camera detection of the person in the left side of the image is recovered by cooperative fusion (indicated by green box) due to the proximity to a confident radar detection.

about which sensor will perform best in the particular scene setting.

Formally, the camera gating (dotted diamond block in Figure 5.6) adapts to pass the weak local camera evidence in the intersection of $\Omega'(\mathbf{x}, \mathbf{g})$ and $\Omega"(\mathbf{x}, \mathbf{g})$:

$$q^{(cam)}(\mathbf{x}, \mathbf{g}) = \begin{cases} a\left(\mathbf{u}^{(cam)}\right) & \text{if } \Omega'(\mathbf{x}, \mathbf{g}) \cap \Omega"(\mathbf{x}, \mathbf{g}) \geq \tau, \\ q^{(cam)} & \text{otherwise,} \end{cases} \tag{5.2}$$

where τ is the overlap threshold and $q^{(cam)}$ is the baseline camera detection threshold at the set working point. The intersection of regions can be computed from the image bounding box of a weak camera detection closest to $\Omega'(\mathbf{x}, \mathbf{g})$: $\mathbf{u}^{(cam)}; a^{(cam)}(\mathbf{u}, \mathbf{s}) < q^{(cam)}$, and the image bounding box of the projection of a confident Radar/Lidar detection closest to $\Omega"(\mathbf{x}, \mathbf{g})$: $\mathrm{f}\left(\mathbf{u}^{(range)}\right); a^{(range)}(\mathbf{u}, \mathbf{s}) > q^{(range)}$, onto the camera image:

$$\Omega'(\mathbf{x}, \mathbf{g}) \cap \Omega"(\mathbf{x}, \mathbf{g}) = \mathrm{IOU}\left(\mathbf{u}^{(cam)}, \mathrm{f}\left(\mathbf{u}^{(range)}\right)\right), \tag{5.3}$$

where we use the Jaccard Index (intersection over union $\mathrm{IOU}(.)$) as a metric for overlap/closeness on the image plane and use an overlap threshold of $\tau = 0.5$.

The proposed adaptive detection threshold is truly cooperative because it uses processed information from the illumination-invariant range sensors to potentially adapt the decision making of the camera. However, the sensors remain conditionally independent, meaning that the cooperative feedback is not a necessary component of the system. If the range sensors do not report anything to the feedback line, then the camera detector will continue to operate in its nominal state, applying the pre-defined camera detection threshold over the entire imaging field. The effect of the method is two-fold: first more camera evidence referring to the presence and position of true objects can be recalled from the images without changes to the image analysis network, and second: more of the camera false positives will remain suppressed below a

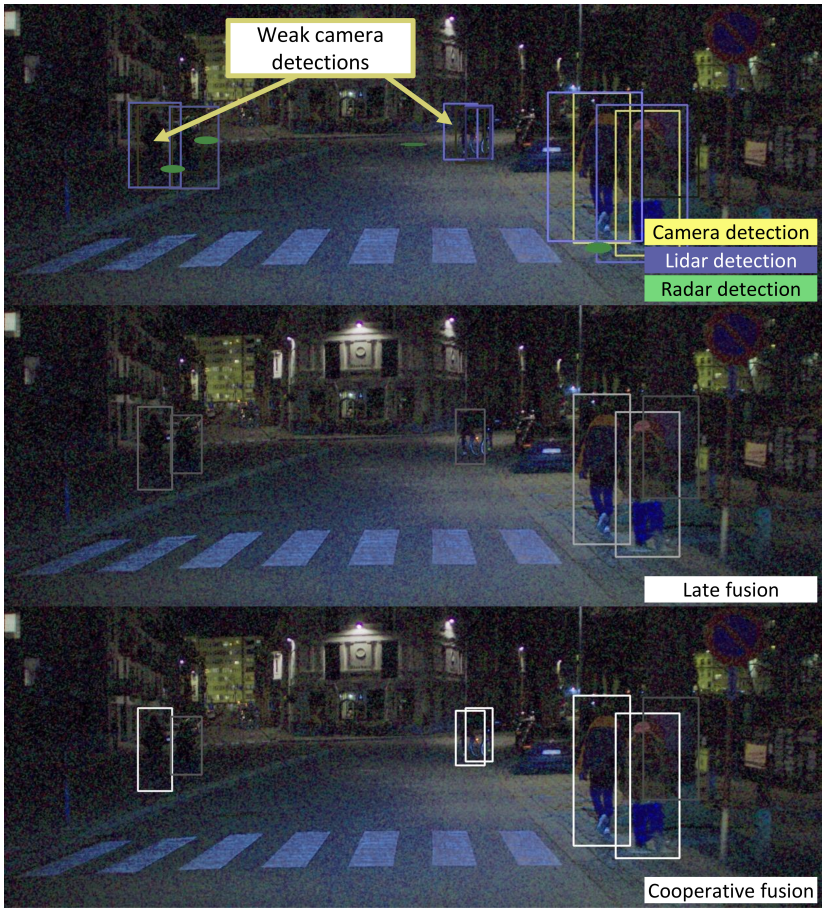


Figure 5.8: Example of a nighttime scene where camera has poor detection performance in the poorly lit region in the left side (top image). Late fusion with Radar and Lidar (middle image) yields detection with reduced confidence due to the lacking camera evidence. Cooperative fusion (bottom image) significantly improves the detection confidence for the poorly visible people in the scene.

sufficiently high detection threshold which can be set at an operating point of higher precision since we can be more certain that the Lidar/Radar sensors will help in regions of difficult viewing.

To illustrate the effect of radar→camera feedback we analyze the traffic scene shown in Figure 5.7. In this example there are three pedestrians, two to the left of the image and one to the right. Due to the low ambient light of the scene, the shape, color and texture of the pedestrians is not accurately recorded by the camera. Current object detectors are not yet capable to confidently detect the presence of objects under such circumstances resulting in bounding boxes with low detection scores. The camera de-

tection confidence is especially low for the second person to the left of the frame and when we use a constant detection threshold over the entire image this detection is lost. Unaffected by the low ambient light, the radar detector is able to detect all VRUs, indicated by the green ellipses on the ground plane. The lines overlaid on Figure 5.7 further shows the computed proximity (as computed in Eq. (5.3)) of all possible camera-radar matches on the image plane. We can see that the weakly detected VRU in the distance matches very closely to a radar detection. In these regions our approach lowers the camera detection threshold to recover any missing camera detections, indicated by a green bounding boxes in Figure 5.7. In section 5.7.1 we will experiment with applying this feedback over a wider range of traffic scenes where we will experimentally show that the proposed cooperative radar→camera mode of operation can be very beneficial in scenes where camera detection is poor.

In another example, we apply the cooperative feedback mechanism in a three-sensor perception system consisting of a camera, radar and lidar. Contrary to the methods presented in Section 4.4 where the lidar was *only* used as means of obtaining a depth map for ranging, in this example the lidar *also* performs independent object detection based on the 3-D measurements. Thus, the system fuses three sets of detections: camera 2-D bounding boxes, radar 2.5-D ground plane targets, and lidar 3-D bounding boxes. The proposed sensor cooperation method improves the camera object detection prior to fusion, by feedback consisting of confident radar and lidar detections which adjust the camera detection threshold to pass weak camera detections.

Practically, we compose a list of confident radar and lidar detections which we project onto the image plane and boost weak camera detection using Eq. (5.3). In Figure 5.8 we demonstrate the system output operating under very poor viewing conditions. On the top image, the individual detections of the three sensors are visualized in distinctive colors. The bounding boxes in the middle image show hypotheses $(\mathbf{x}, \mathbf{g})$ which have maximum belief in existence computed by late fusion, while the bounding in the bottom image show the corresponding hypotheses computed by cooperative fusion using an adaptive camera detection threshold guided by radar and lidar evidence. Encoded in the bounding box brightness are the posterior probabilities of the belief in existence which clearly indicate improvements in the cooperative fusion case. In the cooperative fusion example, all 6 vulnerable road users have been detected with maximum confidence, whereas the confidence of many of the late fusion hypotheses is quite low. In section 5.7.1 we present a systematic analysis of these improvements through experiments conducted over a large data sample which shows the effectiveness of the proposed method.

5.5 Radar object detection with camera feedback

Frequency modulated continuous wave (FMCW) radar has the capability to directly measure an object's range and radial velocity (Doppler). A radar with an array of antennas can use the technique of phase shifting to steer the signal and sample range/Doppler over specific azimuth/elevation regions. While classical radar signal

processing can be applied efficiently to detect large objects such as vehicles, discriminating people from clutter in traffic environments remains a difficult task. This is mainly due to the fact that non-metallic objects are relatively weak radar reflectors. For this task techniques such Doppler-based background subtraction have been proposed and shown to be effective to some extent when the scene is static. However, when radar sensor is in motion, the Doppler shift of the perceived environment overwhelms the signal of road users making them difficult to detect. Additionally, the effects of multipath propagation of radar signals are difficult to model explicitly due to the unknown and ever changing scene geometry. The only prospect for people detection by radar is to look for their unique motion patterns in the time-frequency domain.

A pedestrian or a cyclist exhibits an oscillatory motion of the limbs which is easily captured by radar. The resulting radar signal is commonly referred to as a micro-Doppler signature. One such example is demonstrated on Figure 5.9 where the oscillatory nature of the micro-Doppler signature of the pedestrian is highlighted on the bottom plot. In this example, the micro-Doppler signature of the person is clearly evident in the time-frequency plot. Classifying whether the object is a person, regardless of its position, can be accurately done using the Doppler signal integrated over all range and azimuth bins. However, in object detection we also need to estimate the position of the objects. This task becomes more difficult when there are multiple people in the scene and their micro-Doppler signatures superimpose.

When classifying whether an object is present in the scene or not it is sufficient that algorithms look at the entire radar signal. However, in order to estimate the location of objects, the patterns in the range-azimuth signal need to be interpreted at a local level which, in turn, reduces its signal to noise ratio when compared to interpreting the complete radar scan at once. It is clear that the problem of people recognition and localization in radar is exceedingly difficult even without taking into account ego-motion. The scientific community agrees that radar signal processing has to be extended to concepts from machine learning and pattern recognition in order to keep radar in the leading edge of remote sensing [163].

The proposed radar detector improves upon the state-of-the-art by applying the concept of cooperation feedback from the camera. However, the same principle can be applied if the feedback line is populated by information from other sensors such as lidar. The driving principle behind the cooperative radar detector is that radar provides excellent ranging, but limited azimuth information which can be greatly improved by using detection priors along the azimuth. For example, due to occlusion or noisy signal, an object might not entirely be separable from clutter by the radar, but confidently detected in the camera image. The under-performing radar then uses confident detections of the well-performing camera as feedback information to better detect the missing object.

It should be noted that a naive implementation of such sensor-sensor feedback can easily cause unpredictable system behavior, especially when one sensor's false positives are employed as strong priors. False prior information can lead to skewed detection results by the radar making it inaccurate even in situations where it would

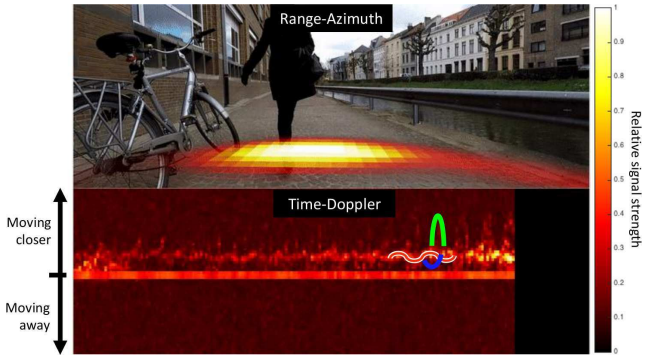


Figure 5.9: Traffic scene where a pedestrian is walking towards the ego-vehicle which is static. On the top plot, the RADAR signal is projected on a range-azimuth plane and overlayed with the RGB camera image while on the bottom plot the RADAR Doppler signal is plotted over time.

be otherwise quite reliable. The suggested object detector was taught to anticipate potential loss of feedback information and prevent these kinds of instability while achieving noticeably greater detection rates than the baseline detectors.

In order to do this, we employ an augmentation approach where the feedback signal is randomly turned off during training. We turn off the feedback signal by randomly decreasing the confidence of the camera detections (which feed back into the radar) or completely missing. This simulation is intended to replicate low-light conditions where the camera detector is compromised but the radar signal remains unaffected. The network may then focus more on the radar signal and only use the extra feedback data when it is available as a result. When the feedback line is empty, the radar detector can switch to a nominal mode of operation where it doesn't use the feedback data but still performs at baseline (for instance owing to a malfunctioning camera or an empty scene).

Our radar detector performs object detection by mapping the radar signal (and any available feedback information) into a 2-D heat-map of VRU occupancy $\mathbf{o}$. Local clusters $o(\mathbf{x})$ of high occupancy in the CNN output represent regions which are likely containing people. The position and extent of detected people can be extracted from this 2-D heat-map by a spatial clustering or local peak finding algorithm. The method is based on a convolutional neural network which directly predicts the single-frame spatial occupancy $o(\mathbf{x}_i)$ at positions $\mathbf{x}_i : (\rho_i, \theta_i)$ in polar coordinates on the local ground plane. We train the model using dense ground truth occupancy maps where cells containing people are labeled as occupied and the remaining cells are empty, see section Section 2.4 for definitions.

The input to the CNN is a 3-D tensor consisting of temporal series of dense radar measurements where each value (in range-azimuth-Doppler space) represents the radar signal strength captured in this grid cell. The output value of each element $o(\mathbf{x}_i)$ is proportional to the probability that the grid cell at spatial position $\mathbf{x}_i$ is oc-

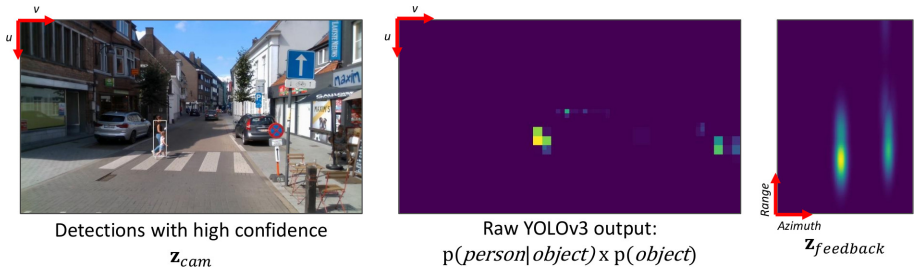


Figure 5.10: Visualization of the camera feedback information reinforcing the radar signal. Left: the scene with detected VRUs, middle: the raw output from YOLOv3 and right: the camera feedback array.

cupied by a VRU. For convenience, we chose the output to have the same spatial resolution as the input signal.

The radar CNN has an additional input, a temporal series of feedback arrays, each representing confident detection information from other sensors along the previously mentioned range-azimuth space. Practically, we use the camera feedback information which consists of the camera detection scores for objects at various position on the ground plane. We compute the camera feedback vector by projecting each point $\mathbf{x}_i : (\rho_i, \theta_i)$ of the ground plane onto the camera image using the same transformation explained in Section 5.6. Then, we find the likelihood for it being occupied by a person given the camera activation of the closest detected object in that image location $a(\mathbf{u}, \mathbf{s})$. This procedure is illustrated with an example in Figure 5.10.

Due to large variations of detected bounding boxes in the vertical image direction, the feedback array has poor localization along the range axis and excellent localization along the azimuth axis. We find that this complementary information can be of great benefit to the radar detection CNN since the radar data has a much lower azimuth resolution.

Radar CNN architecture The task of detecting moving VRUs, consisting of localization and classification, motivates a CNN architecture capable of both spatial and semantical analysis of the input signal. While localization can be done relatively effectively using single-frame processing, VRU classification in radar through the use of micro-Doppler cues necessitates the use of temporal information. A joint detector-classifier therefore requires both a wide spatial receptive field, as well as significant feature depth in the time-Doppler dimension. The former is needed for learning to separate targets from each other and from multi-path reflections, while the later is essential for gait classification. The combination of spatial layers and memory modules has the disadvantage of costly training, where data sequences must be processed as time series and GPU resources cannot be fully utilized. We therefore choose to concatenate five consecutive radar cubes along a common time-Doppler dimension and use a 2D U-Net architecture.

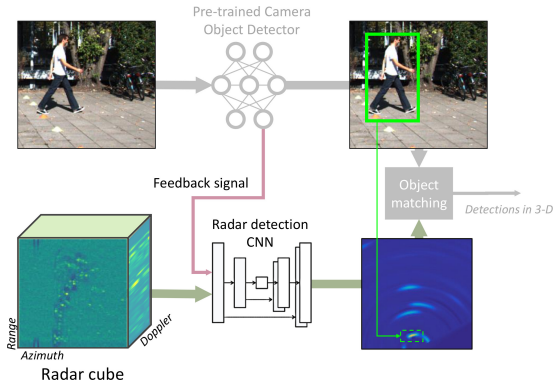


Figure 5.11: Diagram of the proposed cooperative radar CNN detector. Left: stacked raw radar data cubes and camera feedback vectors, middle: the camera and radar detection CNN structures, right: the weak supervision camera-lidar used in training.

Following pre-processing, our information dense tensor is fed to the contracting head of the U-Net, Figure 5.11, where a series of convolutions and max-pooling operators reduce the spatial information into a more dense feature space. The bridge of the network, containing two Fully Connected (FC) layers, then classifies the presence of moving VRUs. Up-sampling is performed in expansion blocks using the dense FC features and high resolution information from the contracting blocks via skip and concatenation layers. Finally, a sigmoid activation function is used to map the network output to predict the probability for occupancy of a VRU at each range-azimuth cell. For training, we use a per-sample and per-class weighted, two-class cross-entropy loss function. Per-sample weights account for the varying confidence in our weakly supervised ground truth at the specific cell, while per-class weights adjust the desired sensitivity of the output.

Pre-processing using domain knowledge Typical radar data is usually represented as dense 3-D arrays containing time integrated range-azimuth-Doppler signals, or 4-D arrays if the radar also measures elevation. Depending on the maximum unambiguous Doppler velocity i.e., the maximum range of radial velocity that can be observed without ambiguity by a Doppler radar, the captured consists of velocities far greater than that of a human body. For example, if the radar is configured for a maximum unambiguous Velocity of 50Km/h then most of the motion of a human body (in the range of 0 to 10Km/h) will be captured in the first few Doppler bins while the remaining signal is of little relevance for the task of detecting people. Since most of the radar data contains little relevant information, feeding the complete array to a CNN is sub-optimal and computationally expensive. We therefore developed a pre-processing function built on domain specific knowledge which resulted in data reduction at minimal loss of information. Knowing that the mean frequency of human gait is around to 1Hz, while cyclists on average pedal at a cadence of roughly 60RPM, a single period

of this motion carries the minimum detection information. Therefore, we concatenate radar measurements spanning the time period from $1s$ in the past until the present. In order to reduce the effect of range dependent signal decay, we also apply the standard power normalization step from the classical CFAR [164] algorithm.Practically, this operation is performed as part of the CNN using a convolution layer with frozen weights.

In moving radar systems, there exists a typical distortion of the signal structure proportional to the ego-velocity and the cosine of the azimuth. The structure of the radar data cube will therefore vary significantly from sample to sample and naively teaching the CNN to detect people in such distorted inputs is sub-optimal. Knowing that this distortion comes largely from the ego-motion of the vehicle, we can correct the input tensor by compensating for ego-motion. The resulting, ego-velocity independent space will be used to teach the CNN much more effectively, meaning that we will need less training samples to cover the solution manifold because the road users will appear to be moving through a static radar signal. We estimate the ego-velocity from the Doppler bin which has the strongest radar signal strength. This technique is effective if we assume that most of the radar energy is reflected from the static environment. Averaging the signal over all ranges in a forward-looking circular sector ($\theta \in [-30°, 30°]$) we obtain an estimate of the radial velocity of the static environment with respect to the vehicle and the ego-velocity is simply the negative of this value. After estimating the ego-velocity, we discard Doppler bins that far exceed normal VRU velocities from the input tensor. We make sure that discarding Doppler slices will not impact detection performance by taking a margin which will include only the velocities expected from the road users of interest such as pedestrians and cyclists. This helps us reduce the size of the input tensor and thus lessen the load on the GPU for training.

Loss function and regularization We perform training of the radar CNN parameters using a training set consisting of radar data samples and ground truth information in the form of a dense 2-D occupancy array. As previously discussed, the radar CNN is designed to output a 2-D array $\hat{\boldsymbol{o}}$ predicting the ground truth occupancy with the goal to make a local distinction between two categories, $c = 2$, one encoding the empty space, and the other the space occupied by VRUs. In this section we define the supervised learning strategy for training the CNN model parameters by optimizing a loss function between the network output and the ground truth. The loss function is designed to lead to a minimal occupancy error which correlates with optimal detection performance.

In order to train the network parameters using supervised learning, we need ground truth information which in our application represents the ground plane occupancy for each region in the scene. Manually annotating the ground plane is especially difficult when working with radar datasets because the data is non-intuitive to the untrained eye. We therefore used a calibrated radar-lidar-camera sensor array and used the camera and lidar data to label the scene for the presence of road users. In order to reduce the costs of labeling an adequately large dataset we used a weakly-supervised strategy

where objects are pre-labelled by an algorithm and then checked for consistency by a human annotator. Specifically we use detections from camera (YOLOv3) and range them with lidar using our own instance segmentation algorithm explained in Section 4.3. These ranged objects can then be easily projected on to the ground plane to form the occupancy map which the radar will use as a ground truth. Ground plane grid cells occupied by matched objects are therefore labeled as positive while the remaining cells are labeled as negative.

Since the camera and lidar detectors themselves provide imperfect detection, we use the camera detection scores as control for the radar CNN learning process. More precisely, we employ a weighted loss function during the training process where the weight for each positive sample is proportional to the camera detection score. This way the radar CNN will be trained by using the most confident camera-lidar detections as ground truth.

In most typical driving situations, the area occupied by VRUs (pedestrians, cyclists, etc.) is significantly lower than the area of the free space. This implies that, in almost all frames, the ground truth occupancy map will be populated mostly by zeros. Thus, when designing a loss function we need to make sure that the occupied cells in the ground truth are given enough importance over the empty space. This way we can achieve an application-specific balance between learning a model that accurately detects empty space and one that accurately detects occupied space. At the time of writing the thesis, cross-entropy is widely considered as the fastest converging loss function for optimizing CNNs for multi-class prediction tasks. The loss function computes the average cross-entropy of the entire ground plane and propagates its derivative back into the network parameters to adjust them for minimal error. In our case, each output $\hat{o}\left(\mathbf{x}_i\right)$ is a 2-D array element of class predictions which we compare to the ground truth element $o\left(\mathbf{x}_i\right)$ and average into a single value:

$$\mathcal{L}\left(\hat{\mathbf{o}}, \mathbf{o}\right) = \sum_{\mathbf{x}_i} w_{\mathbf{x}_i} l\left(\hat{o}\left(\mathbf{x}_i\right), o\left(\mathbf{x}_i\right)\right), \tag{5.4}$$

where $l()$ is the cross-entropy function:

$$l\left(p, q\right) = -\sum_{i=1}^{c} p\left(i\right) \log q\left(i\right), \tag{5.5}$$

and the sample element weights $w_{\mathbf{x}_i}$ incorporate the object detection score $a(\mathbf{x}_i)$ using an appropriate scaling factor α depending on the ground truth class:

$$w\left(\mathbf{x}_i\right) = \begin{cases} \alpha_{pos} & \text{if } \mathbf{x}_i \text{ is occupied,} \\ \alpha_{neg} & \text{otherwise.} \end{cases} \tag{5.6}$$

where the parameters α_{pos} and α_{neg} is adjusted for the desired detector specificity to reduce the class imbalance. In our experience we found that a ratio $\alpha_{pos}/\alpha_{neg} \gg 1$, which heavily penalizes errors in occupied cells compared to the errors in empty cells,

brings better model performance.

We also use the camera detection scores from the automatically generated labels to weight positive samples which has the effect of penalizing errors in cells which are strongly believed to contain VRUs by the camera. Additionally, we looked at an automated method of weight adjustment based on classification error in the current training epoch [165]. This way the weighting of each class is dynamically reallocated by the supervised classification loss during the training stage, formally:

$$w\left(\mathbf{x}_i\right) = \frac{\mathcal{L}_{\hat{c}}\left(\hat{o}\left(\mathbf{x}_i\right), o\left(\mathbf{x}_i\right)\right)}{\sum_{k\in\{0,1\}} \mathcal{L}_k\left(\hat{\mathbf{o}}, \mathbf{o}\right)}, \tag{5.7}$$

where $\mathcal{L}_k\left(\hat{\mathbf{o}}, \mathbf{o}\right)$ is the average classification loss of a certain class k, and $\mathcal{L}_{\hat{c}}\left(\hat{o}\left(\mathbf{x}_i\right), o\left(\mathbf{x}_i\right)\right)$ is the loss of the element $\mathbf{x}_i$ for the predicted class $\hat{c}$. In the beginning of the training stage, $w\left(\mathbf{x}_i\right)$ is initialized to 0.5 to enable a fair start. After several training iterations, the background clutter would dominate the loss and make the model ignore the other classes due to the huge contribution gap and the imbalanced sample number. However, once the model learns to classify the clutter, the adaptive weight will have an equal value for the different categories, and the model will focus on the positive class again. Note that $w\left(\mathbf{x}_i\right)$ is updated regularly during the training iterations.

The effects of both of the weighting schemes are two-fold: firstly, network coefficients will adapt to produce strong activations at grid locations matched to highly confident detections from the camera, and secondly: there will be strong activations at locations with high quality matching between the camera and lidar objects. This point in the CNN optimization has been further studied in [165] where by applying a self-balancing, modulating loss term (focal loss), the network was able to achieve even higher accuracy.

We use three different regularization techniques which help with parameter stability and minimize over-fitting. Firstly, the network architecture design itself includes dropout and batch-normalization layers. Regularized network parameters become more robust to perturbations in the input data and are able to converge faster due to the reduction of internal covariance shift [166]. Secondly, we apply a multi-epoch training protocol using the (Adaptive Moment Estimation) ADAM optimizer [91] and apply weight decay of $5 \cdot 10^{-3}$ to all network parameters. We reduce the global learning rate by a factor of 10^{-1} every 10 epochs starting from 10^{-3}. Lastly, we apply a realistic data augmentation technique which randomly flips and rotates the input tensor as well as the ground truth along the longitudinal axis. Such 2-D rotations and reflections are Euclidean plane isometries and thus preserve geometrical properties such as lengths and reflection angles. Additionally, we randomly attenuate the feedback tensor from the camera in order to simulate situations of poor camera detection performance. Augmenting the dataset in this way creates an abundance of new realistic samples while the structure of the scene is generally preserved. More details on the training and evaluation process are given in Section 5.7.1.

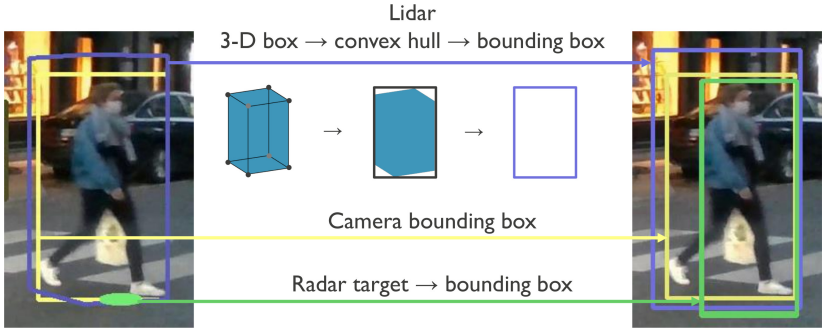


Figure 5.12: Example of camera (YOLOv3), radar (RadarCNNv2) and lidar (CenterPoint) detections projected onto the image plane.

5.6 Matching detections across modalities

Sensor fusion computes the belief of object existence as well as the belief in object position given multi-sensor observations through the evaluation of the likelihood functions Eq. (2.5), Eq. (2.6) (in the multi-sensor case Eq. (2.12)) and Eq. (2.20). This requires us to find, within $\Omega(\mathbf{x}, \mathbf{g})$ which is a specific region on the ground plane near $\mathbf{x}$, the detector outputs $\mathbf{z}_{l_k}^{(k)}; k = 1, 2, ...$, that match best to our hypothesis position and shape $(\mathbf{x}, \mathbf{g})$ in the respective sensor coordinate systems. Moreover, when we need to initialize the system with hypotheses and we have no significant prior information, the best approach is to spawn hypotheses around regions $(\mathbf{x}, \mathbf{g})$ which match to confident detection evidence. In multi-sensor systems, a detection is usually considered to be confident if it is confirmed by two or more sensors. This means that the matching between detections across modalities is a crucial task which can greatly influence the fusion outcome.

In this analysis we give an example how to match detections between a camera, radar and lidar which is representative of many fusion systems in automotive perception. Therefore, we seek to find the closest matching camera $(k = 1)$, radar $(k = 2)$, and lidar $(k = 3)$ detection $\mathbf{z}^{(k)} = \left(\mathbf{u}^{(k)}, \mathbf{s}^{(k)}, a^{(k)}, \mathbf{f}\right)$ to a hypothesis at location $\mathbf{x}$ with a certain shape $\mathbf{g}$. The problem is constrained within a region $\Omega(\mathbf{x}, \mathbf{g})$ on the ground plane which is specific to the object class, for example: if we are detecting people then $\Omega(\mathbf{x}, \mathbf{g})$ may be a region where it's unlikely to find another person i.e., the personal space of a person at a candidate position $\mathbf{x}$. We seek the detections from the three sensors that are closest to the position $\mathbf{x}$ and shape $\mathbf{g}$, formally for each sensor we seek the detection $\mathbf{z}_l$ which satisfies $\operatorname{argmax}_l \mathrm{d}\left(\mathbf{z}_l, \Omega(\mathbf{x}, \mathbf{g})\right)$ where $\mathrm{d}\left(.\right)$ is a distance function measuring the proximity in distance and similarity in shape in image coordinates.

Since our perception system uses a combination of cameras, radars and lidars, computing the closest detection to a a hypothesis requires a distance function $\mathrm{d}_k\left(.\right)$ unique to the sensor k and as we will see bellow, the design of this function is not

trivial. The camera object detections are defined in the camera image plane and lack range information and radar detections are defined on the ground plane and lack height information making the matching between $\left(\mathbf{u}^{(k)}, \mathbf{s}^{(k)}\right)$ and $\Omega(\mathbf{x}, \mathbf{g})$ ambiguous. For example: a tall person in the distance appears similar to a shorter person that is closer by in the image. In order to compute the best matching sensor evidence that will ultimately optimize the multi-sensor belief in our objects of interest we need to overcome this ambiguity.

When computing the best matching camera detections we chose to project the hypothesis $(\mathbf{x}, \mathbf{g})$ and all detections on the image plane and find potential matches whose projections overlap the most in terms of image area. Projecting a hypothesis $(\mathbf{x}, \mathbf{g})$, a 3-D bounding box, can be done without loss of information if we simply apply the perspective projection explained in Appendix A. The same holds true for lidar detections since they too are fully defined by their 3-D location and shape. When projected onto an image, each of the 8 bounding box corners form a convex hexagon which we can then describe with a smallest enclosing box to make it compatible with the image detections, see the example shown in blue in Figure 5.12. However, this process becomes ambiguous for radar targets which lack height information because we cannot compute the exact pixel positions of the top of the object. Instead, we assume that each radar detection is of average height and width using models learned off-line, see the example shown in green in Figure 5.12.

Having projected all of the detections onto the camera image, we can find matching detections which match with a hypothesis using the Jaccard index of their projected (or assumed) bounding boxes: $d(A, B) = \frac{|A \cap B|}{|A \cup B|}$, where A is the set of image pixels for the hypothesis projection and B is the set of image pixels for the respective detection projected on the image plane. If the sensor-to-sensor calibration is performed accurately, then we can expect that detections stemming from the same object will usually overlap with a ratio $d(A, B) > 0.4$. If the camera detections are ranged, for example using a depth map, then the matching process can alternatively be performed on the ground plane using the Euclidean distance. In order to range detections on the image plane, we compute the median depth value within a central region (25%) of a 2-D bounding box which we found to almost always coincide with the true distance of the object, see the example area in gray in Figure 4.5.

The example shown in Figure 5.13 demonstrates the projected detector outputs in a typical urban driving scenario. It is important to note not every object will be detected by all sensors and there will be many spurious single-sensor detections. In object tracking, as it will become apparent in the following chapter, the positions and shapes of hypotheses $(\mathbf{x}, \mathbf{g})$ will most often match with detections from more than one sensor. However, at initialization, populating the hypothesis space is usually done by finding $(\mathbf{x}, \mathbf{g})$ that match with confident detections from one sensor or less confident, but matching detections from two or more sensors. The motivation behind this can easily be illustrated by the example in Figure 5.13: consider the printed poster of a person to the right of the frame which is detected only by the camera. This detection does not stem from a real object of interest and is a bad evidence for the presence of a

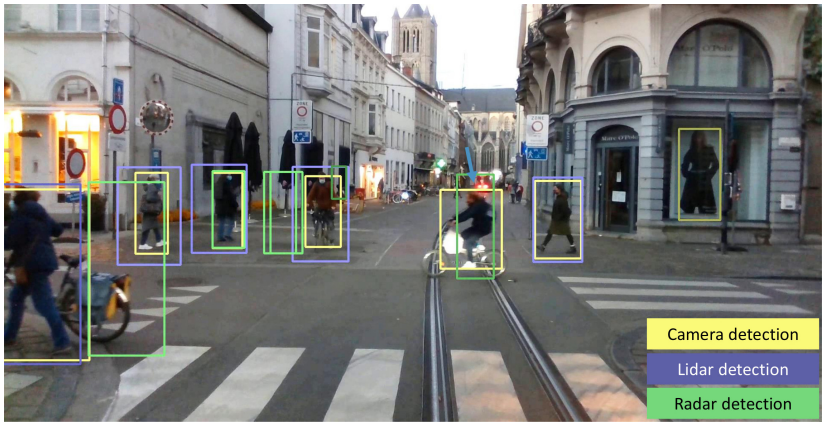


Figure 5.13: Example of camera (YOLOv3), radar (RadarCNNv2) and lidar (CenterPoint) detections in the city center of Ghent.

person given a hypothesis at that location. Populating the hypothesis space with such data can lead to needless computations.

We use the same principle when computing the camera to radar feedback in Section 5.5. The feedback is represented in a sensor-agnostic format i.e., ground plane occupancy where the probability of occupancy for each position is computed the belief for object existence based on the camera activations. In practice the feedback information is stored as a 2-D array whose elements $o(\rho_i, \theta_i)$ are populated by projecting each location (ρ_i, θ_i) onto the camera image assuming an average person height and width. Thus, in the image we have a set of hypothesis bounding boxes which we then match with camera detection outputs using the Jaccard index. The example shown in Figure 5.10 illustrates the feedback information computed from confident YOLOv3 detections.

5.7 Experiments and results

This section provides details about the experimental evaluation of all of the proposed cooperative fusion methods for instantaneous object detection. Due to the lack of a standardized benchmark for comparing the performance of the various sensor fusion combinations, each proposed method will be evaluated in a separate experiment, tailored specifically for the task. Methods that fuse lidar and camera information will be tested on the evaluation principles of the KITTI and nuScenes datasets, while methods for lidar, radar and camera fusion will be tested on our internal imec datasets. In all experiments, the performance of the proposed methods are compared either to the state-of-the-art or to a comparable control method. Of special interest is the system performance when one of the sensors becomes compromised. To test this border case we will remove the camera signal from the fusion system and measure the object de-

Method	AP	Recall at 0.5 prec.	Training notes
RADAR raw	0.150	0%	
RADAR non-static	0.302	26%	Empirically optimized
CFAR	0.439	53%	
CNN-manual	0.513	57%	Supervised 1351 frames, 3917 labels
CNN-auto*	0.556	61%	Weakly supervised: 6955 frames, 7781 labels
CNN-auto**	**0.600**	**69%**	Weakly supervised: 6955 frames, 30452 labels

Table 5.1: Performance evaluation results of radar detectors on a content-independent test set of 489 frames and 1292 VRUs.

tection performance after fusion. This way we are able to analyze the experimental results and come to unbiased conclusions about the potential gains in precision, recall, robustness and redundancy.

5.7.1 Cooperative fusion between Camera and Radar

This section presents the experimental evaluation of the proposed cooperative radar and camera detectors discussed in Section 5.4 and Section 5.5. The experiments are designed specifically for testing the effect of the feedback information in a broader context of traffic and weather conditions. We will firstly evaluate the radar-only CNN against classical radar detection algorithms as well as test various weak supervision techniques from camera and lidar. Then, we will introduce camera feedback information to the input of the radar CNN model and test the further improvements in detection performance. Finally, we will experiment with feeding back radar and lidar detection information into a pre-trained camera object detector and evaluate the gains in performance over the baseline.

Radar CNN baseline

For the purpose of evaluating the performance of the proposed radar CNN when no camera feedback information is available, we captured and annotated a dataset containing multiple scenarios with various traffic conditions and complexity. In these experiments, the vehicle is driving on public roads in a dense city environment, where multiple VRUs are encountered on the sidewalks and on marked and unmarked crossing zones. The data covers situations from poorly lit environments (20% of the data) to well lit sequences captured in daylight. The capturing vehicle was equipped with a

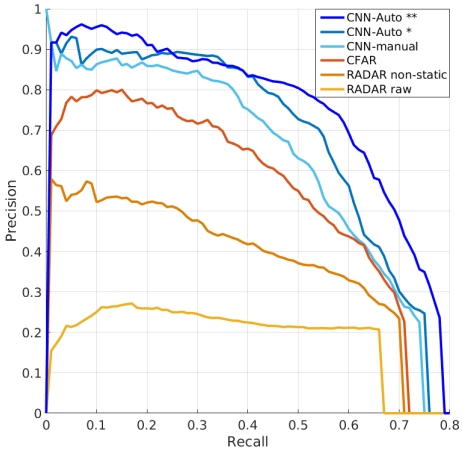


Figure 5.14: Precision-recall curves for the test set (higher area under curve is better). The results of the proposed method "CNN auto" are compared with a control network "CNN manual" and other classical peak finding techniques.

calibrated and synchronized sensor array consisting of an RGB camera (GoPro Hero 6 Black), a 77GHz FMCW radar (Texas Instruments AWR1243) and a 3-D lidar (Velodyne VLP-16). Data was captured and timestamped in on a PC onboard the vehicle. Due to time and restrictions, only part of this dataset was labeled by human annotators. The rest of the data was labeled automatically by matching detections from Faster R-CNN in the camera view to instance segmentation in the lidar point cloud. This dataset is publicly available[1] and can be downloaded upon request.

In order to avoid cross-contamination of training and test data, we split the recording into two content-independent parts by selecting data captured at different time and in different parts of the city. The expert annotators were able to accurately label 1840 frames with 4988 VRU instances which took them about 30 hours to complete. At the same time, by running our fully automated annotation tool over the remaining data we labeled a total of 6955 frames containing 30452 VRUs that will be used for weak supervision training. Note that the auto-labels are only used for training the CNN model, while we perform evaluation only on the ground truth labeled by human experts. The high number of auto-labels stems from the high number of objects detected by the individual camera and lidar detectors, however, many of these auto-labels have a small contribution during training due to the low camera detection score, see Eq. (5.6). Labels consist of the 2-D ground plane position of each person, relative to the ego-vehicle origin. Due to the limited resolution of the available VLP-16 lidar, the areas beyond 20m and outside of the view of the camera are considered as "don't care" regions and were excluded from the training and evaluation.

In a series of experiments we applied different supervision learning methods to

[1] Radar data and annotations are available upon request at: radar-fusion.ipids.ugent.be

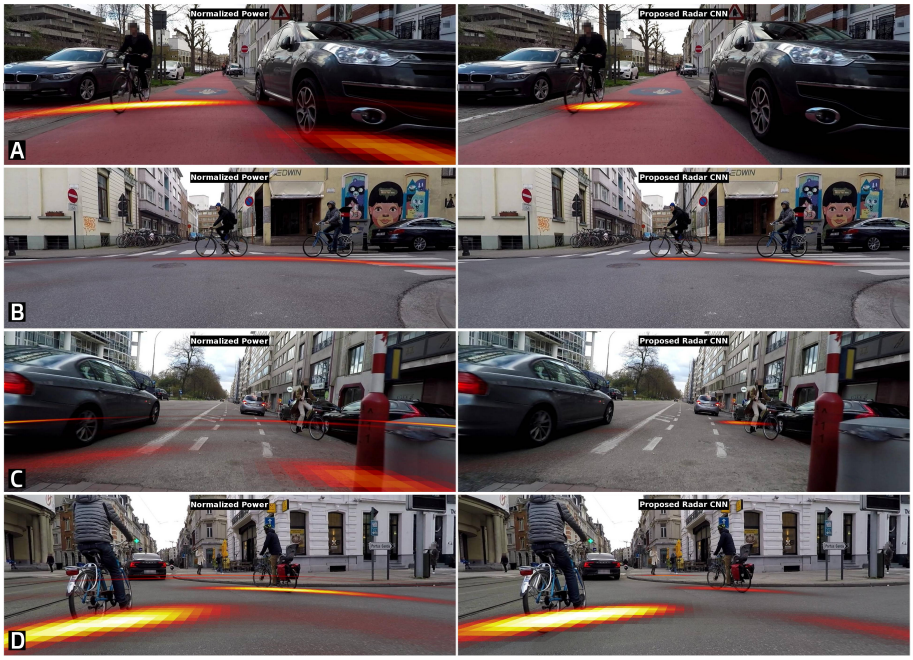


Figure 5.15: Example frames comparing the qualitative difference of the input radar signal (left column) and the radar CNN output (right column). A: suppression of clutter from vehicles; B: spatial separation of VRUs in close proximity; C: suppression of clutter from infrastructure; D: suppression of moving vehicles.

train the several models of the same CNN architecture. Specifically, a control model (CNN-manual) was trained by using supervision from human annotated labels, then we trained two weakly supervised models: CNN-auto* and CNN-auto** using only confident auto-labels and using all auto-labels respectively. The hypothesis that we want to evaluate in this section is that even though the auto-labels are imperfect, their abundance can be beneficial for training a better performing detector. The weakly supervised CNN models were trained using additional training data that was expensive to label manually.

Due to the same network architecture and number of parameters, data pre-processing, CNN optimizer and training hyper-parameters were all kept the same over every experiment. Each model was evaluated on the same, content-independent test set, consisting of 6 unseen sequences. We measure detection performance by matching peaks in the CNN output array to the true positions of VRUs in the ground truth. By varying a detection threshold over the CNN output we also compute the proportion of true positives, false positives and false negatives at multiple detector operating points. To that end, a non-maximum suppression (NMS) algorithm finds peaks in the CNN output in 5×7 cell range-azimuth neighborhoods. These polar patches equate to an area

of $1.46m \times 7.04^{\circ}$ in the physical world and were chosen because they optimized the detection performance in terms of Average Precision. A detection is considered true positive if it is located within $3m$ of a ground truth object, while multiple matches within this same gate are not allowed and counted as false positives. For testing object identification on the ground plane, these parameters are typical. These statistics are used to create a precision and recall curve for each detector. The average precision (AP) is calculated by averaging the precision samples taken at 100 evenly spaced recall points.

We present a summary of the results in Table 5.1 and the computed precision-recall plots on Figure 5.14. The detection performance of the proposed method (in shades of blue) is compared to four other algorithms (yellow, orange and red lines). The weakly supervised "CNN-auto**" significantly outperforms all other methods in terms of Average Precision (AP). We report an increase of 8.7% AP over the control "CNN-manual" which was trained using manually annotated training data. Moreover, by allowing the proposed method to learn the uncertainties about detection and matching in the automatically generated labels, "CNN-auto**" brings additional performance benefit of 4.4% over training by using the most confident camera-lidar matches in "CNN-auto*". Finally, compared to classical peak finding, the proposed method outperforms CFAR (yellow curve in Figure 5.14) by 16.1%. Naive detection algorithms, such as peak finding in the raw signal and in the moving data, compare unfavorably on our dataset. On Figure 5.15 we present typical cases of operation of the proposed method where we compare the input radar signal (left column) to the CNN output (right column). In order to visualize the 280 channels of the input signal, we collapse it to a 2-D array by taking the maximum signal value along the time-Doppler dimension and project it on the respective camera frame. On the right we project the CNN output i.e., the estimated probability of occupancy of a VRU, onto the camera image. From these typical examples it is clear that the network output dramatically reduces false positives while at the same time improving the object localization.

Evaluating the effect of camera→radar feedback

For evaluating the effect of the camera feedback information on the radar CNN performance, a larger dataset covering broader range of traffic and weather conditions was captured and annotated. The dataset in this experiment consists of 317 sequences captured in both day and night and in various weather conditions. 194 sequences are used for training and otherwise parameter tuning while 123 sequences are used for testing. The total number of frames in the selected sequences is 77587, divided into a set of 46360 frames for training and 33957 frames for testing. As such , this dataset provides a realistic benchmark for the camera object detection and consequently for the camera feedback information that flows into the radar CNN.

In this experiment we compare a radar CNN model which was trained by setting all values in the camera feedback vector to zero to a model which was trained using nominal camera feedback. As a reference, we also provide evaluation results for the CFAR detection algorithm on the same test set. We used a U-Net network structure

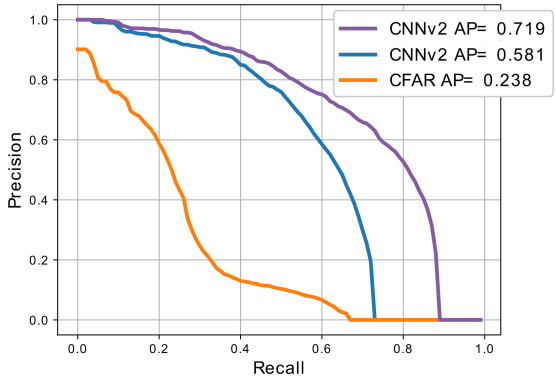


Figure 5.16: Precision-recall curves for the radar CNN with feedback (purple line), radar CNN with no feedback (blue line) and CFAR (orange line). The proposed method has higher recall and consistently higher precision at every recall point than the radar CNN model without feedback as well as CFAR.

consisting of 5 contracting and 5 expanding blocks and a fully connected bridge. The input tensor is a concatenation of 8 radar cubes and the respective camera feedback arrays captured over a time-span of 1.2 seconds. The loss function is a weighted cross-entropy with two classes: person and background.

Detection of people using radar-only signals, blue line on the plot in Figure 5.16, confirms our findings from the previous experiment. The radar CNN model with camera feedback has 23.7% higher AP than the model without feedback and 302% higher AP than CFAR which scores unfavorably on this dataset. Moreover, the model with camera feedback is able to detect significantly more objects with a best possible recall of 88.9% against the best possible recall of 73% of the model without feedback 67% of CFAR. Finally, because we are now evaluating using much larger dataset, it is possible to evaluate the detection accuracy conditioned on specific regions on the ground plane in front of the radar. In Figure 5.17 we show the average precision over various range and azimuth bins for the Radar CNN without camera feedback (left) and Radar CNN with camera feedback (right). The radar CNN without camera feedback performs well for objects in the middle of the field while detection performance decreases to the sides and in the distance. Contrarily, the radar CNN model with feedback performs significantly better in the distance and to the sides of the detection field.

5.7.2 Cooperative fusion between Radar/Lidar and Camera

In this subsection we show the results from the experimental evaluation of the proposed cooperative radar→camera feedback mechanism in Section 5.4. To that end, a baseline camera object detector with a constant detection threshold will be compared to the same detector whose operating point is locally controlled using targets from our

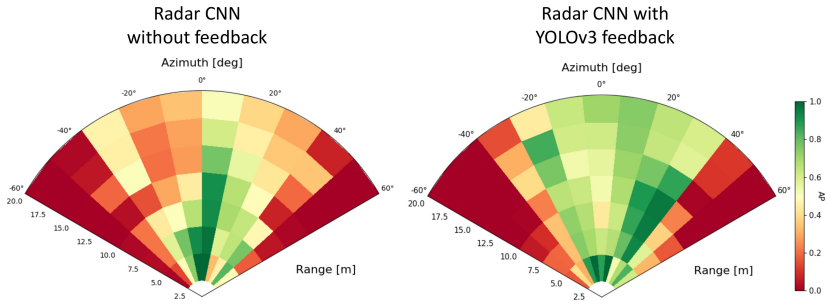


Figure 5.17: Spatial distribution of radar CNN detection performance expressed in Average Precision.

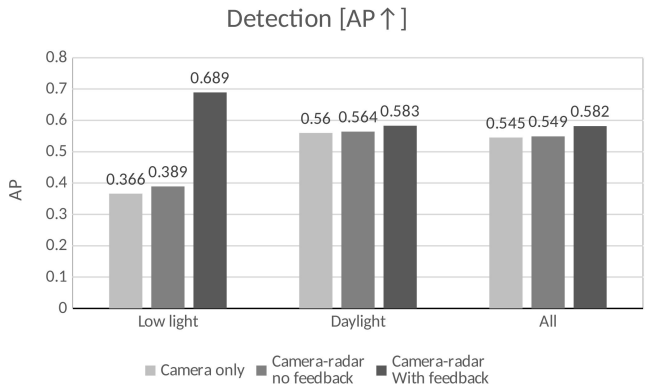


Figure 5.18: Comparison of VRU detection performance of the proposed cooperative radar→camera fusion to camera only and camera-radar with no feedback. The dataset splits (low light and daylight) contain different traffic situations and are not directly comparable.

own radar CNN detector (running without camera feedback). For the camera object detector we used the PyTorch implementation[2] of YOLOv3 [24] CNN. The specific model we used was trained on 80 object categories in the MS-COCO dataset [93] and for our experiments we only select the output for the class person. For processing the radar signal, the radar detection CNN trained to detect micro-Doppler patterns of human body in motion was used [16]. The output of this radar CNN is a 2-D grid of detection scores predicting the position of people. Our first control algorithm is a camera-only detector, where the estimated ground plane position of detected bounding boxes in the image are computed using the back-projection method [167] assuming an average person height of $1.65m$. Our second control algorithm is a fusion technique for matching camera and radar detections on the image plane which doesn't use the radar→camera feedback.

All detected VRUs within $1.5m$ to a ground truth object on the ground plane is

[2]Code available at: https://github.com/eriklindernoren/PyTorch-YOLOv3

Dataset			Detection [AP↑]		
Seq.	# frames	# VRUs	Camera only	Camera-radar	**Camera-radar w/ feedback**
01	175	55	0.177	0.177	**0.397**
02	205	190	0.423	0.448	**0.778**
03	345	443	**0.704**	0.694	0.694
04	125	530	0.632	0.681	**0.695**
05	145	705	0.515	0.599	**0.612**
06	205	1319	0.606	0.588	**0.609**
07	125	290	0.420	0.434	**0.454**
08	25	52	**0.905**	0.818	0.818
09	55	114	**0.956**	**0.956**	0.955
10	85	147	0.424	0.457	**0.474**
11	25	10	**1.0**	**1.0**	**1.0**
12	55	65	**0.802**	0.692	0.692
13	55	113	0.852	0.887	**0.897**
14	105	373	0.246	0.276	**0.293**
15	65	330	**0.595**	0.543	0.572
16	45	252	**0.477**	0.440	0.445
Low light	380	245	0.366	0.389	**0.689**
Daylight	1460	4743	0.560	0.564	**0.583**
All	1840	4988	0.545	0.549	**0.582**

Table 5.2: Summary of the dataset and results for single sensor (camera) and multi-sensor (camera-radar) VRU detection. In terms of AP, the proposed cooperative fusion detector significantly outperforms both other methods in low light and daylight sequences. In terms of MODP, the proposed detector outperforms both methods in low light sequences and has better overall precision.

counted as a true positive, while other detections outside of don't care regions are treated as false positives. We report the results in Table 5.2 where we additionally break down the results for each individual sequence comparing classification Average Precision (AP) and ground plane multi-object detection precision (MODP) scores for the three tested detectors. Set averages, presented on the bottom of the table and visualized on Figure 5.18, measure the performance for the low-light and daylight segments as well as for the complete dataset.

In terms of AP, the proposed radar→camera cooperative fusion (dark gray bars) outperforms camera only detection by 3.7% and radar-camera fusion by 3.3%. The performance difference is especially pronounced in low-light sequences where the proposed method shows as much as 32.3% and 30% improvements over the controls

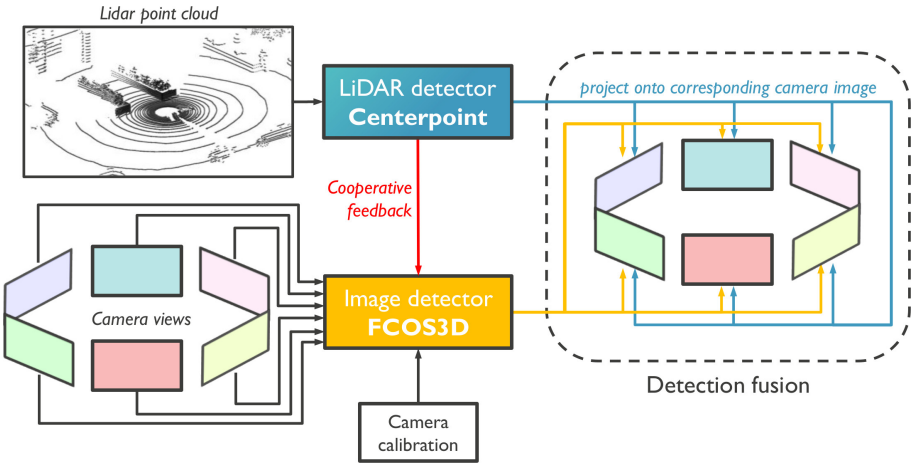


Figure 5.19: System diagram of the proposed cooperative radar→camera fusion detector applied on the sensor setup of the nuScenes dataset.

respectively. This improvement is a direct consequence of the radar→camera feedback loop which controls the camera operating point locally and is much more effective in low-light where camera detection performance weakens. The proposed method has increased maximal recall and precision meaning more VRUs are being detected while at the same time producing less clutter. Otherwise, in well lit sequences, the proposed cooperative fusion method produces slightly better results than intermediate fusion which makes it robust and predictable.

Evaluating the effect of lidar and radar→camera feedback

In this subsection we show the results from the experimental evaluation of the cooperative lidar and radar→camera feedback mechanism. For this purpose we will use a perception system consisting of three sensor modalities, each performing an object detection and ranging as shown in the system diagram in Figure 5.5. The three sensors share their confident detection information and after fusion achieve increased detection precision retaining a robustness to sensor failure.

In the first set of experiments we applied our method on the camera and lidar data from the nuScenes dataset [168]. Our main objective is to test our two hypotheses: first that the fused detections outperform both individual detectors in terms of precision, and second, the fused detections are robust to sensor failure. NuScenes is a multimodal dataset with 360° coverage across all vision and range sensors collected from diverse situations, including rainy and nighttime conditions. It consists of 6 camera views, 1 3-D lidar and 5 radar data streams captured over 1000 scenes. Data frames are recorded at a variable frame rate (2Hz˜10Hz) and synchronization is achieved using time stamps.

The data is annotated for the presence of 8 classes of road users (car, truck, bus, trailer, construction vehicle, pedestrian, motorcycle and bicycle) as well as 2 classes of road infrastructure (traffic cone and barrier). The dataset is split onto 750 training sequences, 150 validation sequences and 150 testing sequences. Training and validation can be performed using the publicly available ground truth, while testing is done by an independent online evaluation server.

To test the effectiveness of the proposed fusion method we applied two state-of-the-art object detectors in the image and lidar data and fuse their outputs using cooperative feedback. Specifically, we use the FCOS3D object detector [52] in the camera data and the Centerpoint object detector [51] n the lidar data. Fusion is performed by first matching lidar and camera detections on the respective image plane using the methodology explained in Section 5.6, see Figure 5.19. Note that in these experiments we do not use the nuScenes radar data because it provides semi-processed target-level information consisting of target's position and Doppler. This type of radar input is incompatible with our Radar CNN object detector. Moreover, semi-processed radar information is difficult to interpret since it does not provide classification information.

The nuScenes object detection evaluation measures the average detector precision (AP) as computed at multiple recall values. A detection is considered to be a true positive if the predicted class label matches a ground truth object within an association radius on the ground plane. Thus, the detector needs to estimate the object category and localize them accurately at the same time. In order to avoid over-optimizing the algorithms for any specific class of objects or gating radius, the average precision metric is computed for several gating radii (0.5m, 1m, 2m and 4m) and the mean over all object categories is computed (mAP). The computed mAP values lie in the range (0, 1) where higher values indicate better object detection performance.

We present the experimental findings by evaluating on the nuScenes validation sub-set for which publicly available ground truth information is available. To test the first hypothesis, we fused the camera and lidar detections and compared them to the baseline detectors and a state-of-the-art early fusion object detector [142]. We found that the proposed fusion method outperforms both individual detectors which shows that cooperative fusion is effective. However, as expected, it is outperformed by the early fusion method.

In a second experiment, simulating compromised camera operation, we deliberately disable the camera feed to both fusion methods simulating a hard camera failure. The proposed cooperative fusion method, in this case, shows the same performance to the lidar-only detector (mAP=0.633), but the precision of the early fusion method degraded far below the baseline (mAP=0.247). This experiment shows the fragile nature of early fusion when faced with out-of-domain input. The exact numerical values for the measured precision in all settings are summarized in table Table 5.3.

Even though the nuScenes dataset does contain nighttime sequences, the evaluation protocol computes average precision over all data samples which does not pinpoint the border cases where cooperative fusion is beneficial. In our experiments we observed little to no benefit of activating the lidar→camera feedback on the average detection performance (obtaining the same mAP=0.633). However, the benefit of the

nuScenes category	FCOS3D (Cam.)	Centerpoint (Lidar)	Proposed (Cam.-Lidar)	MVP (Cam.-Lidar)	**Proposed** *faulty cam.	MVP *faulty cam.
Car	0.725	0.855	0.862	**0.878**	**0.862**	0.678
Truck	0.488	0.585	0.595	**0.634**	**0.595**	0.235
Bus	0.651	0.715	**0.731**	0.710	**0.731**	0.280
Trailer	0.344	0.372	0.385	**0.391**	**0.385**	0.070
Constr. Veh.	0.138	0.171	0.211	**0.223**	**0.211**	0.020
Pedestrian	0.566	0.851	0.871	**0.894**	**0.871**	0.714
Motorcycle	0.445	0.588	0.644	**0.732**	**0.644**	0.029
Bicycle	0.411	0.433	0.543	**0.634**	**0.543**	0.010
Traffic cone	0.528	0.697	0.759	**0.799**	**0.759**	0.119
Barrier	0.492	0.685	**0.729**	0.703	**0.729**	0.270
All	0.479	0.595	0.633	**0.660**	**0.633**	0.247

Table 5.3: Mean Average Precision on the nuScenes object detection dataset (validation subset).

cooperative feedback in this data is more pronounced in object tracking which will be discussed in more detail in the following chapter. Finally, the nuScenes dataset lacks the raw radar signal we need in order to apply our RadarCNN object detector. In order to address these shortcomings, we performed additional testing using our own camera-radar-lidar dataset (referred to as imec v2 dataset).

The imec v2 vulnerable road user dataset consists of a total of 316 sequences of duration between 10s and 20s, captured throughout the year 2020 in several cities across Belgium. This dataset is labeled for 1318 unique VRUs across 173095 instances. Ground truth is evaluated within the range of [0m, 20m] and an azimuths of $\pm 35^{\circ}$ (the field of view of the RGB camera) while the area beyond these ranges is ignored. The benchmark uses a spatial gate of 2m for accepting true positives. We applied the YOLOv3 object detector in the camera data, RadarCNN model with feedback for the radar data and Centerpoint for detection in the lidar data. Additionally, we computed depth images using the algorithm in Section 4.3.1 and used the depth to range the detections from YOLOv3. The precision of each individual detector and fusion methods are summarized in Table 5.4. Unfortunately, due to incompatibilities in data formats and limited time we were unable to run the early fusion detector MVP [142] and we chose to not run the faulty camera testing as we did for the nuScenes data.

The proposed cooperative fusion method applies cooperative links from the radar and lidar towards the camera and from the camera towards the radar. We compared the cooperative fusion method to standard late fusion and noticed a significant im-

Category	RadarCNN with feedback	Center-point	YOLOv3 with Lidar depth	Late fusion	**Proposed**
Daytime	0.430	0.656	0.657	0.700	**0.740**
Nighttime	0.452	0.611	0.601	0.701	**0.709**
Unoccluded	0.478	0.639	0.696	**0.742**	0.729
Partly occ.	0.429	0.656	0.639	0.693	**0.743**
Heavily occ.	0.346	0.601	0.559	0.636	**0.688**
Pedestrians	0.458	0.659	0.665	0.695	**0.744**
Cyclists	0.393	0.633	0.600	0.705	**0.715**
All	0.450	0.650	0.641	0.699	**0.734**

Table 5.4: Mean Average Precision on the imec v2 object detection dataset (test sub-set).

provement of detection precision of 5% and an increase of the best possible recall of 1.5% over the entire dataset. The improvements are consistent across all difficult circumstances such as nighttime, heavy object occlusion and fast moving VRUs such as cyclists. Biggest benefits of the three-sensor cooperative fusion over late fusion can be observed for heavily occluded objects (8.1% in terms of AP) which confirms the significant benefits of the additional feedback loop mechanisms which we propose to add to the system. Interestingly, the performance of the proposed method drops slightly (1.8% in terms of AP) compared to later fusion when evaluated on objects with no occlusion. We deem that such results are not surprising as the parameters of the adaptive threshold, Eq. (5.2), were optimized on a content-independent training set. However, this peculiar finding warrants further investigation in the future.

Implementation details

For processing the camera video signals we used the state of the art FCOS3D [52] and YOLOv3 [24] object detectors. YOLOv3 divides the image into regions and predicts bounding boxes and probabilities for each region. The network predicts 4 coordinates for each bounding box (center and size in image coordinates), an objectness score and a class prediction. The input images are re-scaled to 608x608px prior to feature extraction using the darknet-53 backbone. We use a detection threshold of 0.1 and a NMS overlap of 0.4. The feedback information is formed by aggregating the activations for the VRU class across all output scales and back-projecting them onto the ground plane using a model for the average person height which is trained over the respective dataset. In the imec v2 dataset the average VRU object height was 1.8m. FCOS3D performs object detection using the ResNet101 backbone, a Feature Pyramid Network for detection at different scales and multiple regression heads

for estimating the object's class, position and shape. Contrary to YOLOv3, FCOS3D estimates the 3-D position of objects in the scene using the intrinsic camera matrix, inferring the size and distance of objects from labeled 3-D training data. We use a detection threshold of 0.05, a NMS overlap of 0.1 .

For object detection in the lidar data we use the state of the art CenterPoint [51] object detector. This detector performs detection in 3-D, interpreting the objects as 3-D boxes defined by their center, size and orientation. The first stage of CenterPoint predicts a class-specific heatmap, object size, a sub-voxel location refinement, rotation, and velocity. The original method also performs end-to-end object tracking, predicting the object velocity, but this information was not used as we will develop a novel tracking algorithm in the next chapter. The output detections are provided in a sensor-specific coordinates spanning in the $[-51.2m, 51.2m]$ on the ground plane and $[-5m, 3m]$ vertically. Internally, Centerpoint represents the lidar point-cloud data as a voxel grid with resolution of $[0.1m, 0.1m, 0.2m]$ along the X-Y-Z axes respectively.

Our radar CNN uses 3-D radar arrays representing signal strength over a discretized range-azimuth-Doppler space. The extent and discretization of the range-azimuth-Doppler space is programmable and can be set by modifying the operating parameters of the TI AWR1243 radar. The settings used in our experiments resulted in range and Doppler encoding into 128 equally spaced bins, spanning from $0m$ to $46.72m$ and $\pm 13.8m/s$ respectively. Azimuth is encoded in 16 equally spaced bins over the range of $\pm\pi/2$. Power-normalization is performed by computing the local Signal to Noise Ratio (SNR) [164] using a 3-D convolution of the input radar array with a 3-D filter mask. We used a mask with support size of $[15, 11, 1]$ and a guard size of $[5, 3, 0]$ for range, azimuth and Doppler respectively. After estimating and correcting for ego-motion, from the original 128 Doppler bins, we discard 5 Doppler bins encoding the lowest velocities: $|v| \leq v_{ego} + 2Km/h$ and 34 high velocity Doppler bins: $|v| \geq v_{ego} + 23Km/h$. Each training tensor is created by concatenating 8 pre-processed radar arrays that span over a time interval of $1ms$.

The radar CNN architecture is a U-Net [63] with 5 contraction blocks, a Fully Connected (FC) bridge and 5 expansion blocks. The network outputs a 2D occupancy grid in polar coordinates with spatial resolution matching the one from the input data. Every contraction block applies 3 groups of convolution, batch-normalization (BN), ReLU and a dropout layer followed by a max pooling operator at the end. At the bridge, the input tensor is reduced to spatial resolution 1×1 and 512 dimensional feature space which is input to two fully connected layers. The expansion blocks are built as inverse convolutions (ConvT) initialized to perform up-sampling with linear interpolation, followed by BN and ReLu. In order to preserve high resolution details, up-sampled results are concatenated with feature maps from the respective contracting blocks. Expansion blocks are exempt from dropouts since their task is data unpacking and mixing. Fastest convergence was achieved by training both CNN-manual and CNN-auto using weighted cross-entropy loss in conjunction with the ADAM optimizer. In all our experiments, we apply early stopping i.e., we terminate the training once the validation loss starts increasing. Generally, we observed model convergence after ~ 15 epochs or after 110K back-propagations. We used variable training batch

size (BS) starting from $BS = 1$ in the first epoch to $BS = 16$ for the remaining. All of these design choices have a direct impact on either the convergence speed or the loss value at convergence. We note that each hyper-parameter value has been chosen meticulously by running control experiments which are outside of the scope of this analysis.

5.8 Conclusion and practical implications

In this chapter we proposed a cooperative sensor fusion method for improved object detection by applying sensor-sensor feedback. The methods were published as articles in the proceedings of the IEEE Intelligent Transportation Systems Conferences 2018 [12], 2022 [16] and 2021 [17]. Moreover, the cooperative fusion algorithms for detection of people and other road users have been integrated in prototype systems developed during several research projects, the scope and other details about these projects are given in the valorization section of Chapter 7. Results from real-world experiments confirm the benefit of our cooperative fusion method in terms of better recall, lower false positives, and higher positional accuracy. The proposed detector outperforms standard late fusion and significantly outperforms single-sensor detection. Differences are most evident in low-light sequences where the radar's luminance invariance recovers information lost in the camera image processing. When using the proposed camera→radar cooperative feedback we observed a 20% increase in the number of recalled objects without an increase in false positives. Moreover, when using the proposed radar→camera feedback we were able to improve the image object detection in low light situations by almost 80%.

In a three-sensor (lidar, radar and camera) setup, the cooperative fusion detector was optimized to use information from the more accurate sensors into the least accurate ones. Since there are multiple directions of flow for the feedback information, the effectiveness of the feedback very much depends on the specific application. In addition to the said radar⟷camera feedback, we were able to evaluate the effect of the cooperative lidar→camera feedback in two datasets. On the nuScenes dataset we observed minimal gains using the cooperative feedback over standard late fusion, however the proposed method outperformed a state of the art camera-lidar detector when the camera signal feed was lost. Due to limited granularity in the testing protocol of the nuScenes dataset, we were unable to isolate the border cases where the cooperative feedback is beneficial. Therefore, we tested the three-sensor cooperative detector on the imec v2 object detection dataset where we measured a 5% improvement of late fusion in all sequences. Moreover, the benefits were present across multiple difficult scenarios such as nighttime, occlusion and fast moving objects. We did not analyze any feedback links leading into the lidar object detector. This is partly due to lidar being quite effective in detection compared to the other sensors. Still, helping the lidar to detect at high distance using high resolution camera detection information remains low hanging fruit and should be the subject of analysis in the near future.

An overwhelming majority of the sensor-fusion literature at the moment focuses

on fusion using neural networks and deep learning. As we saw in the overview, the techniques which provide the highest detection rates on the standard benchmarks tend to employ early fusion, thus relying on the steady flow of raw sensor data from to the fusion center. Although this paradigm leads to the best precision overall, early fusion methods are rarely applied in real-world systems because they require high bandwidth links and assume uninterrupted flow of data. Whenever one of these requirements fails, early fusion systems are faced with out-of-distribution data which causes a domain shift and drastic decrease in precision. We remain firm in our belief that the robustness of a fusion system is equally as important as its precision, and we hope that the autonomous driving benchmarks in the future will also include such tests. The proposed method performs cooperative fusion at the decision level and is thus robust to sensor failures. Moreover, the cooperative sensor-sensor feedback transfers only a limited amount of confident detections and can be implemented on existing car network hardware.

Our own testing as well as independent evaluation over multiple datasets showed that the proposed methods provide improvements both in terms of robustness as well as improved localization and classification of objects in 3-D. Although these findings suggest a large increase in the instantaneous, frame-by-frame, object detection, at this stage it remains unclear how such performance increase translates to a more complex perception system e.g., in fully autonomous driving. Beyond simulators, there are no publicly available tools which can be used to measure the impact of object detection accuracy into safety critical parameters such as risk of collision. Moreover, real-world systems process sensor data as a continuous temporal stream, and as such, these systems suffer from additional temporal noise not covered in this chapter. This combination of spatio-temporal artifacts makes the fusion and interpretation of the information much more complicated. Since the main goal of environmental perception is tracking and prediction of the state of the environment, solving the remaining problems extends beyond instantaneous object detection. Capitalizing on the knowledge gained in this and the previous chapter, the remainder of the thesis is focused on developing a spatio-temporal fusion system based on the principles of cooperation and Bayesian inference.

6

Cooperative sensor fusion for object tracking

6.1 Introduction

Object tracking increases the confidence of detected objects and corroborates their location by integrating multiple observations of the same object over time. This chapter focuses on the difficult problem of tracking road users for the purpose of collision avoidance and path planning in autonomous vehicles. As autonomous vehicles are envisioned to be able to safely drive under all weather and traffic conditions, the task of detecting and tracking road users can become greatly influenced by the effect these conditions have on the vehicle sensors. For example, glaring light can cause camera detection to deteriorate due to loss of image contrast. Similarly, detection is compromised under rainy or foggy conditions, and the problem is further complicated at night. As we discussed in the previous chapter, improving the capability of the perception system under such circumstances requires the employment of additional sensors. Sensor fusion for tracking of road users must then reason about the transient loss of detection in individual sensor modalities. A special focus in this work is given to solving the challenges of tracking road users using multiple sensor modalities whose characteristics change over time causing out-of-distribution, or sometimes completely missing observations.

In the scientific literature, researchers [169, 170] identify pedestrians as the most vulnerable road users, arguing that more than 2.5% of the injured pedestrians in collisions with vehicles in Germany, and 4% on the EU level, ended up with fatal con-

sequences. Recent sustainable mobility studies such as [171], illustrating that bike-sharing is on the rise in urban centers, corroborate that safe interaction between automated vehicles and cyclists is becoming equally important. The regulation (EC) 78/2009 [172] defines the key term Vulnerable Road Users (VRUs) as all non-motorized road users such as pedestrians and cyclists as well as motor-cyclists and persons with disabilities or reduced mobility and orientation. Thus, fast and accurate prediction of the position of VRUs is critical in avoiding collisions that often result unfavorably for the VRUs. However, the detection and prediction of the position of vehicles is also an important task which will not be overlooked in this chapter.

Object tracking has been widely studied in aerospace applications starting from the 1960's and has resulted in one of the most widely used technique used to this day, the Kalman Filter (KF) [173]. However, the tracking of objects in a changing environment poses additional problems, not covered by the classical tracking theory. Even in the simplest case of tracking a single person with a single sensor, occlusion from foreground objects increases the likelihood that a person will be detected at multiple locations making the posterior probability density function to become multi-modal which violates the basic assumptions of the KF. The problem becomes more difficult when multiple objects are in the scene, which has led to the need for robust Multi-Object Tracking (MOT) methods.

Any single sensor is typically inadequate to offer complete knowledge about the environment, and the tracking literature as a whole generally agrees that a fusion of various modalities is frequently necessary. As a result, the integration of multi-sensor observations is crucial since no single sensor processing system has yet been able to handle the entire perception task on its own. The real-time demands of autonomous driving, however, make it difficult for traditional object tracking since the detectors must be calibrated for speed rather than accuracy. This means that detectors output only a few confident observations and throw away the clutter and any useful information therein. Such settings cause the observation space to become sparse leading to missing detections that make tracking update an ill-posed problem. Standard filtering techniques such as Kalman or Particle filters do not perform state estimation for time instances when data is missing which can be dangerous in safety-critical applications. Therefore, in this chapter we seek to develop robust tracking system which will be able to continue to estimate the sate even in the absence of observational data.

Furthermore, when inadvertent sensing failures do occur, the tracking system needs to adapt to the new operating parameters i.e., the new sensor noise model, and continue operating seamlessly. In such cases, it is necessary to detect that the detector is no longer in its nominal state of work and switch to the true state of operation in order to avoid large estimation errors. Finally, intermittent failures to detect an object can also happen even when a sensor is in its nominal state, for example, because of occlusion or ambiguous object configurations. A well-performing tracking system should be able to cope with the transient changes in operating modes as well as the occasional complete absence of detections.

To illustrate a few of the common difficulties let's consider the example shown as a bird's eye diagram in Figure 6.1. In some places, object detection is hindered

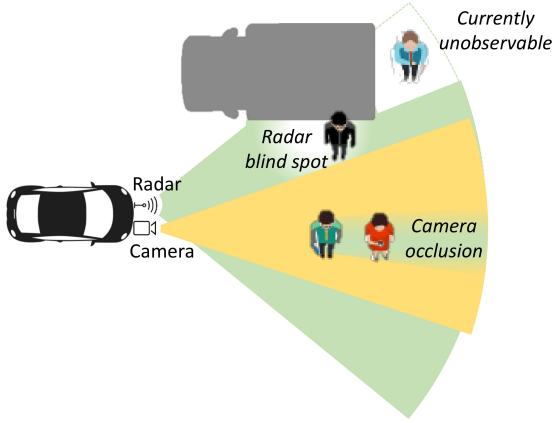


Figure 6.1: Diagram of an intelligent vehicle in a traffic situation showing the different modes of sensor operation. Depending on the scene configuration, areas that are within the field of view of the sensors can produce unreliable or missing object detections.

because of the geometrical layout of the scene. For instance, the vehicle to the left makes the radar signal behind it weaker, making it impossible to see the person in blue. Additionally, the truck's flat surfaces produce clutter and secondary reflections of previously identified objects, leading to radar blind spots or regions of low signal strength. Objects of interest in the camera view might potentially be hidden by other foreground objects. In the best-case scenario, occlusion reduces the confidence in object detection; nevertheless, when an item is entirely obscured, object detection is impossible. Even more challenging is the fact that detection quality might deteriorate throughout the field of vision; for instance, accuracy decreases as distance increases. Systems designed for nominal functioning will experience concept drift in such circumstances. It indicates that the model's target variable's statistical characteristics vary in unexpected ways over time. This frequently causes an overestimation of the error covariance when employing the Kalman filter, which makes the tracker unreliable. Techniques such as ensembles of variance-limiting Kalman filters [174] have been proposed in simulation but never effectively applied to people tracking.

In the literature, tracking-by-detection is the recommended paradigm for tracking road users. By dividing the whole tracking process into two steps—detection of positions separately in each frame and construction of tracks by linking matching detections across time—algorithms based on this idea can make the task much simpler. Tracking by detection, as opposed to tracking before detection, applies high confidence thresholds in the object detection step and thus operates on a lower quantity of detection evidence, requires less operations and memory. Performing object detection at a high-precision working point reduces the amount of detections that must be linked over time, but it also results in brief bursts of missing detection information. Missing detections can be extrapolated while monitoring a single target using mo-

tion and sensor models. However, lost detection information in multi-object tracking leads to ambiguity and is challenging to recover. Rarely does the literature address the best way to handle such missed detection occurrences. However, the solutions, particularly in safety-critical applications, might significantly affect performance in the actual world. The study in this chapter will concentrate on novelties in motion prediction, as well as alternative statistical approaches, which are typically used to address missing detections.

The proposed tracking method extends the instantaneous detection explained in the previous section by temporal integration of multi-sensor information. It is capable of adapting to changing sensors configurations and changing modes of operation. Furthermore, tracking continues seamlessly even in cases of missing detections where the lost information is recovered using imputations sampled from a proposal function based on the sensor evidence without association. Then, in order to further improve the detection in difficult areas of the scene, we use the predicted positions of confidently tracked objects as an additional cooperative information which populates the detection feedback line, refer to the system diagram in Figure 5.5. The cooperative feedback from the tracker allows under-performing detectors to adjust their working point parameters in regions where we expect to detect an object with a high degree of confidence. The output of the proposed tracker is a list of hypothesized positions of road users, their category, orientation and velocity; attributes that can be easily interpreted by most collision avoidance and path planning algorithms.

We evaluated the tracker first in simulation and then, tuned to operate on a real sensor array consisting of cameras, lidar and radar with intersecting fields of view. As described in the previous chapter, instantaneous object detection is a achieved through a cooperative fusion of multi-sensor information where each sensor runs its own, high recall, object detection neural network providing detection information. In the proposed system, the radar detector [16] outputs dense probability maps in range-azimuth space with peaks at expected road user's positions, while the camera detector [24] and lidar object detector [51] output a rich list of bounding boxes. Detection-to-track association is performed by minimizing a matching cost consisting of a distance and appearance terms using the Kuhn-Munkres (Hungarian) algorithm [28]. A switching observation model particle filter handles individual track state estimation by adapting to changes in sensor modes of operation as well as sensor-to-sensor handoff. In cases when detections are missing, the tracker samples particle weights from a precomputed grid of detection information containing all sensor evidence and recovers part of the missing detections. The grid is computed using a joint-sensor measurement model conditioned on detection probability before tracking. Track maintenance is done based on the belief in the track existence using the log-odds ratio.

Experimental evaluation, both in simulation and on multiple datasets captured in the real world, shows a significant improvement in detection and tracking performance over other optimal trackers such as Kalman filter, particle filter (PF), switching observation model (SOM) PF and multiple imputations (MI) PF. The proposed tracker outperformed all publicly available pedestrian trackers on the KITTI tracking benchmark and showed competitive performance on the nuScenes tracking benchmark. The

proposed tracking system is especially effective in border cases where low light, complex scenes with multiple VRUs, heavy occlusion, and large ego-motion hinder the performance of other trackers from the literature. The resultant track estimates remain within tolerable ranges of the ground truth position, even in cases where up to 50% of detections are missing.

The remainder of this chapter is organized as follows: Section 6.2 gives an overview on relevant tracking methods based on sensor fusion, and in Section 6.3 we lay out details of the Bayesian principles for tracking the position and the existence of object hypotheses. Then, we explain the components of the proposed tracker: the algorithm for estimating the location in Section 6.4, the proposed motion model in Section 6.5, the proposed switching observation models in Section 6.6, the proposed algorithm for handling missing detections in Section 6.7 and the track management algorithm in Section 6.8. Finally, the experimental evaluation, results and discussion are given in Section 6.9 and Section 6.10 respectively.

6.2 Literature overview

General overview

Multi-sensor, multi-object tracking is an interdisciplinary field with applications reaching far beyond the scope of this thesis. This overview focuses on object tracking papers relevant to the topic of autonomous vehicles, as well as papers whose ideas inspired the proposed tracking method. Trackers can be split into ones that operate in the present i.e., on-line, and ones that process historic data off-line. On-line methods such as recursive Bayesian estimation are suitable for time critical applications like the one covered in this thesis, while off-line methods can re-process historic information at each estimation step to achieve higher tracking accuracy but have the downside of being slower. Off-line methods are effective in applications where we are interested in the highest tracking accuracy and are not limited by computing resources. Therefore off-line trackers process all past as well as future observations to estimate the state of each object at each time instant. There are of course a myriad of methods that lie somewhere in between like “near on-line” methods that introduce small time lag by using temporal windowing for better accuracy.

In terms of the input data, the same tracking methods can be applied in single or multi-sensor systems. Single sensor systems are simpler to optimize and deploy, and depending on the application, can produce satisfactory tracking. Moreover, the failure cases of single-sensor trackers are easier to predict and understand. Multi-sensor, multi-modal trackers, on the other hand, becomes more effective when the other sensors can help to discriminate ambiguities such as occlusions, missed detections and detection clutter. In the past few years we have observed a trend of multi-sensor methods in the literature that exploit heterogeneous modalities such as cameras, multi-spectral, range and/or positional data. This trend is especially apparent in safety-critical autonomous driving applications which will be covered in more detail in the following.

The current state of the art in object tracking improves the performance by innovations in several key areas: fusion of multi-modal object detection for tracking, assignment problem optimization, motion modeling, handling missing observations, and lately, end-to-end detection and tracking using deep learning. For a broader review of sensor fusion techniques for instantaneous object detection we refer to the literature study in Section 5.2. Regarding detection to track assignment, finding globally optimal, solutions to the tracking association problem [175] has mainly been solved using graphical models for connecting individual object detections into consistent set of trajectories: k-shortest paths in DP NMS [176] or a Conditional Random Field as in DCO-X [177] or a variational Bayesian model in OVBT [178]. Modeling the motion of targets within the image, was also given a lot of attention with a some of successful approaches: SMOT [179], CEM [180] and MotiCon [181]. Authors in these papers based the matching costs for comparing pairs of detections on simple distances and weak appearance models. These methods currently score around 10% worse than the state of the art. Very recently, there was a shift towards designing a strong appearance based similarity metrics for the pairwise matching. By doing so, authors have reported a notable increase in their absolute tracker performance and also an increase robustness towards operation in difficult and complicated scenarios.

Some of the recently best performing approaches are based on sparse appearance models such as LINF1 [182] or on-line appearance updates in MHT DAM [183] and channel feature appearance models, oICF [184] and aggregated local flow of long-term interest point trajectories in NOMT [185]. In addition to the image based analysis authors have proposed to exploit depth information in order to improve tracking performance with one of the most recent notable advances using a combined 2D-3D Kalman filter by [186]. A recent trend is the proliferation of deep learning into the tracking community with sparse but notable examples such as MDPNN16 [187], which uses Recurrent Neural Networks in order to encode appearance, motion, and interactions. Another example is JMC [188] which uses deep matching to improve the association metric. There usually is a correlation between strong affinity models and tracking performance which, together with machine learning approaches, is believed to be one of the key aspects to be addressed to further improve performance [175]. Regarding object motion, trackers employ models ranging from simple zero-velocity [189–192] or constant-velocity [193, 194] to constant-acceleration and behavioral models like the one used in this thesis.

The association of detections to tracks, a combinatorial optimization problem, is largely solved using the Hungarian method [28]. The system proposed in [195] visually detects and tracks multiple persons using a stereo camera placed at an under-head position. This method detects people from a face detector applied in ROI selected from depth information. The matching stage finds the globally optimal associations of detected candidates to existing tracks using the Munkres (Hungarian) method. Matching likelihoods are computed from the distance to the predicted position and the similarity to the color histogram appearance model estimated with the Bhattacharyya measure. The appearance model is updated by linear combination of its current values and the new observed color data.

Almazán et al. [196] also use the Hungarian method for candidate matching. They aggregate data from multiple RGB-D sensors using a polar coordinate space representation of the common ground plane. Objects are detected from motion and size cues using depth information which are then matched to trajectories. The correspondence likelihood is based on the distance to the predicted position and on appearance similarity computed using the Bhattacharyya measure. The appearance model combines a height and color histograms and is updated every ten frames by replacing bins and their associated distributions by newly observed ones if available. Mo et al. [197] also exploit the Hungarian algorithm for matching detected and tracked objects, where they identify background areas with a depth-based occupancy grid system. Candidate targets are searched from foreground areas which is analyzed with a cascade of classifiers, comprising face and skin detectors and a full body HOG-based human detector. Detected objects are tracked simultaneously with a compressive tracker and a Kalman filter.

Trackers based on Particle Filters

A particle filter is a sampling-based algorithm that computes an approximate solution to Bayesian inference. Particle filters use the paradigm of genetic algorithms in order to re-sample state particles according to a fitness function. Munoz et al. [198–200] use a single particle filter per track. They use a constant speed model to predict the next location of the target and new target observations are located by maximizing a detection probability. Specifically in [198, 199] candidate objects are identified from ROI based on depth information and the probability of the presence of a person is computed based on the number of points in a cluster and its maximal height. To compute the probability of detecting the tracked person, this human presence probability is combined with an interaction factor that allows handling trajectory crossings by imposing a minimal separation between the positions of different people. In [198], the detection probability also includes the Bhattacharyya appearance similarity measure, while in [199] it uses a measure of confidence on depth. Hence, the trajectory representation in [199] does not include any appearance model, and in [198] it models appearance by the color histogram of the cluster. This model is updated with new observations that have high detection and matching confidence by the linear combination of the previous model and of the new histogram.

In [200] the detection probability is made up of three terms. It includes the probability of being a frontal-facing human, firstly by verifying that the cluster may be approximated by a vertical plane at the expected distance from the camera, and secondly, by evaluating the fitting of an ellipse on the RGB image in order to validate the presence of the elliptical shape of a head at this position. It also uses the Bhattacharyya appearance similarity measure to compare to the trajectory representation's appearance model, made up of two color histograms inside two ellipses of predefined sizes and respective positions that represent the head and torso respectively. This appearance model is updated dynamically as in [198]. In all three methods new tracks are initialized when unknown targets are detected based on the use of generic person

descriptions. Tracks are kept for a number of frames after occlusion or departure.

Choi et al. [201, 202] use particle filtering with Reversible Jump Markov Chain Monte Carlo (RJ-MCMC) sampling to track multiple people simultaneously, as well as static non-human objects (obstacles). Given the positions and velocities of all tracked targets and the results from generic person detectors applied to ROI, at each iteration a move is attempted to initialize, delete or update a trajectory. Each move is sampled from a space of possible moves and a likelihood for the new solution is estimated. Moves are accepted or rejected similar to MCMC sampling until the chain converges. The moves are guided by the probability of continuous tracking, based on a smooth target's motion constraint, which may also account for people interactions and the probability of being a human. The use of a global RJ-MCMC method is computationally heavy since the update of each hypothesis requires the evaluation of the likelihood of all sensor observations. Authors usually overcome this issue by limiting the extent of the likelihood functions to local windows in the hypothesis space which greatly reduces the number of computations, but also departs from the global tracking model. In our approach we break down the global problem into a set of independent tracking problems assuming that the road users we are tracking cannot occupy the same space at once. This way the track updates can be computed independently of one another and the tracker can run in real-time.

While the authors of [202] model the appearance in the sensor model by computing the distance from a target-specific appearance-based mean-shift tracker, in [201] they do not use any appearance model at all and in [185] they define a novel Aggregated Local Flow Descriptor (ALFD) that encodes the relative motion pattern between a pair of temporally distant detections using long term interest point trajectories (IPTs). Another contribution in [185] is a near on-line tracking approach using data-association between targets and detections in a temporal window, that is performed repeatedly at every frame. Leveraging on the IPTs, the ALFD provides a robust affinity measure for estimating the likelihood of matching detections regardless of the application scenarios. We argue that motion descriptors, albeit effective as an appearance metric, are quite computationally heavy to compute in image data. For some other sensors such as Radar, part of the motion (the radial component) is measured directly, yet and its use in applications where computing power is limited should be reconsidered.

Trackers using Cooperative Fusion

Gruyer et al. [203] propose a fusion method based on single-line lidar and camera. Detected regions of interest in the lidar data are projected on the camera image plane and instigate tracks. The authors make strong assumptions about the object size in the lidar point clusters which helps to reduce false positives, but only of the objects satisfy the assumptions. Tracking based on belief theory therefore continues by evaluating motion vectors within the projected ROIs. Regions that match with content from past time instances get associated with existing tracks. This approach does not perform object classification of any sort and relies on assumptions about the detected regions based on their size and motion.

In [204], a comprehensive cooperative fusion approach using a pair of Stereo Cameras and lidar is presented for the task of robust, accurate and real-time detection of multi-obstacles in the automotive context. The authors take into account the complementary features of both sensors and provide accurate and robust obstacles detection, localization, and characterization. They argue that the final position of the detected obstacles is likely to be the one provided by a laser scanner, which is far more accurate than the stereo-vision. The width and depth will be provided also by laser scanner, whereas the stereo-vision will provide the height of the obstacles, as well as the road lane position. This cooperative system uses a scheme which consists of introducing inter-dependencies: the stereo-vision detection is performed only at the positions corresponding to objects detected by the laser scanner, in order to save computational time. The certainty about the laser track is increased if the stereo-vision detects an obstacle at the corresponding position.

The authors of [205] propose a cooperative radar and infrared sensor fusion technique for the ultimate goal of reduced radar radiation time. They rely on a interacting multiple model (IMM) and an unscented Kalman filter (UKF) to perform tracking whereas the residual of the new information is used to adaptively control the sensor working time. They use the first radar measurement to initialize a track and solve the non-synchronized target detection of the radar and infrared sensors. Furthermore, the probability of switching the Radar on or off is proportional to the residual of the innovation obtained by comparing the filtering result with the estimated measurement. The application of this paper is in aerial target tracking, however, the technique of using the innovation residual for sensor control feedback is directly applicable for automotive systems. In our tracking system, the observation model can switch between the individual sensor modalities based on a control variable whose evolution depends on the evidence likelihood. Our tracker can therefore send a signal to the individual sensor detectors to adjust their detection thresholds in regions of poor detection performance.

In [206] the authors propose a radar and camera sensor fusion approach for the tracking of vehicles. They use a combination of a smart camera and automotive-grade radar in a cooperative fusion scheme. Tracks are initialized from within the narrow Field Of View (FOV) of the radar, but can then be tracked also outside of this FOV as long as they remain visible in the camera image. During an update, each track triggers a raw image search to look for a vehicle in the area where it is predicted to be. The likelihood of the object being a vehicle is calculated using histogram search techniques and evaluating the symmetry of the region. Tests are carried out on highway, rural, and urban scenarios and show a very good detection rate while keeping the number of false positives very low. The paper does not provide details about the cooperative aspect of the image search, which relies on basic edge and symmetry-based object detection. Finally, the paper provides a subjective evaluation of the methods. It does not quantify the effect of fusion on the performance. The general system design is, nonetheless, highly applicable to our perception application.

A three-sensor fusion approach was presented in [207] proposing to match tracking outputs by radar, lidar and camera. The authors propose a Permutation Matrix Track Association (PMTA) which treats the optimal association of tracks from two

sensors as an integer programming problem. They relax the association optimization by treating it as a soft alignment instead of hard decision. Thus each entry in the permutation matrix is a value of an objective cost function consisting of spatial, temporal mismatched cost and entropy terms. Of special interest to our approach is the model of the spatial closeness, which these authors design as the joint-likelihood i.e., the product of Radial Basis Functions (RBF) over the distance, velocity and heading of objects. However, the choice of parameters for these functions is not well motivated in the paper.

Tracking in clutter and handling missing detections

Sensor to sensor mismatches due to occlusion, sensing failure or various other factors will inadvertently occur in the real world. Besides the clutter and false detections, measurements in regions where we expect to find objects can go missing, thus it is of great interest to find an optimal strategy for dealing with missing measurements. The literature on tracking in missing data has been so far overwhelmingly focused on regression and classification analysis in big-data problems, while missing data in object tracking remains understudied. A common practice being the usage of a missing indicator variable and propagation of the past estimate and covariance. However, the authors of [208] demonstrate that in regression and classification problems, methods based on missing-indicator variables are outperformed by ones using statistical imputation. In this context, imputation is the process of replacing missing data with substituted values based on the statistics of available data from the past. Specifically, the study [208] measures the difference between the standard method where missing data is indicated by a missing-indicator variable and various imputation methods on 22 datasets when data is missing completely at random. The authors compared classifier performances after applying mean, median, linear regression, and tree-based regression imputation methods with the corresponding performances yielded by applying the missing-indicator approach. The impact was measured with respect to three different classifiers, namely a tree-based ensemble classifier, radial basis function support vector machine classifier, and k-nearest neighbors classifier. In our work, we extend the analysis to the multi-object tracking problem where we compare the missing indicator Kalman and particle filter with a multiple imputations approach.

In [209] the authors look at a few track-to-track fusion methods comparing whether it is better to estimate the missing information or ignore it. They use two 2-D (single-model) Kalman filter trackers using identical and time-synchronized sensors. Three different target motion behaviors were considered: discrete white noise acceleration, constant velocity, and constant acceleration. Track fusion by three different methods was analyzed: best-of-breed which selects the tracker with minimal covariance at the update, fusion without memory where tracks from individual trackers are combined and the estimate persists only for the current update, and fusion with memory which maintains the combined state estimate and covariance from update to update. This study showed that there is no clear winner meaning that tracking all types of motion depends on the estimation of the process noise and the target motion types. The

authors note that accelerating targets are problematic for all methods.

In tracking situations where the observation is polluted with clutter, the probabilistic data association filter (PDAF) has been studied in [210], where the authors propose a multi-object extension: a joint probabilistic data association filter (JPDAF). They argue that the proposed approach is far superior to standard heuristic tracking approaches such as local and global nearest neighbor standard filters (NNSF). They show that in a simulated space-based surveillance example, the PDAF can track a target in a level of clutter in which the NNSF loses the target with high probability. Additionally, they argue that the PDAF and JPDAF, using on-line, recursive state estimation equations, has far lower complexity than the off-line multiple hypothesis tracker (MHT) in terms of computation time, memory, and code length/debugging. Depending on the scene complexity, mainly the number of VRUs being tracked, even the JPDAF solution is rather complicated since all detection-track pair likelihoods need to be evaluated and updated. To combat this problem, Tchango et al. [211] propose to update the state of multiple tracks by selecting and separately updating groups of targets in interaction. The complexity of the update step is reduced by data association and gating heuristics. Inspired by both these works, we concluded that no single technique provides desirably high tracking precision as well as low computational complexity. Therefore, we propose a dual approach where a high detection threshold is applied and only confident detections are associated to tracked objects for optimizing execution time. When inadvertent assignment ambiguities or missing detections do occur, we revert to a probabilistic association approach by re-using sub-threshold sensor evidence.

Handling of missing detections by multiple imputations particle filter (MIPF) has been successfully applied in astrometry by [212]. Albeit defined as a tracking-before-detection, the same principles have been applied in other domains as well. One notable example is underwater acoustic signal processing in [213] where the low signal-to-noise ratio, random missing measurements, multiple interference scenarios, and merging-splitting contacts in measurement space are found to pose challenges for common target tracking algorithms. The authors of this paper propose a tracking-before-detection particle filter that estimates particle likelihood functions directly using the beam-former output energy and adopts crossover and mutation operators to evolve particles with a small weight. The state estimate is therefore largely independent of the availability of detection and significantly outperforms a track-after-detection method based on a Kalman filter. Due to the differences in domains between this paper and our own, the direct application of the proposed method is not possible. However, we adopt the idea of using the raw sensor evidence values as an estimate of the missing detection likelihood function and draw imputations accordingly when detections are missing.

Finally, a very relevant analysis on dealing with missing data in non-linear state estimation with particle filters is presented in [214, 215]. The authors propose a multiple imputations particle filter formulation that uses randomly drawn values called imputations to provide a replacement for the missing data. Under the assumption that the missingness is conditioned on the available data (missing at random), opposed to

missingness being completely independent of observable data (missing completely at random), imputations can be drawn using a proposal sampling function similar to the estimated posterior distribution in tracking. Then, using the particle filter the tracker can estimate the non-linear state with a complete dataset. The paper also presents a convergence analysis of the proposed filter in a non-linear system where observations are missing at random and shows that it is almost surely convergent. The performance analysis is compares the proposed MIPF with existing techniques such as extended Kalman filter, sigma-point Kalman filter and Expectation-Maximization algorithm. They conclude that the multiple imputations particle filter has superior performance when up to 50% of the data is missing. This method provides us with the theoretical framework for handling missing data and has inspired the work proposed in this chapter. The following text extends the idea to a switching observation model particle filter and a novel imputation proposal function based on the likelihood without association for better conditioning of the sampled imputations on the sensor evidence.

6.3 Proposed method

Multi-object tracking must answer the following critical questions: are there any objects of interest based on sensor evidence throughout time, if so, how many, where are they situated, and what is their predicted location in the future? Within the application of autonomous vehicles, tracking needs also be computationally and memory efficient, and introduce very little time lag. In order to be accurate and keep within the limits of these requirements, we make the following assumptions about the nature of the problem.

Assumptions:

1. We *assume* that, the existence of a road user is independent of the specific location within a small region of the state space $\Omega(\mathbf{x})$. This enables us to split the estimation of the object existence and its exact location into a chain of semi-independent tasks. The estimation of object existence can thus be done using the sensor activations $a(.)$ within this region, and, assuming that an object exists, the estimation of its exact location can be performed using the geometric part of the sensor measurements: $(\mathbf{u}, \mathbf{s})$.

2. We *assume* that every road user has a Markov property i.e., the future state (both their location and existence) depends only upon their present state. This assumption enables us to apply Bayesian filtering for the estimation of both the object existence state as well as the object location state. Bayesian filtering can be implemented on low-power, low-memory hardware and can generally allow real-time tracking.

3. We *assume* that multiple road users cannot occupy the same space $\Omega(\mathbf{x})$, which in most cases results in sensor observations which largely independent from

each-other. This means that every object causes the sensor to generate a unique observation, independent of the presence or position of other objects. It allows us to apply simple, localized detection to track association which greatly reduces the combinatorial space in scenes with many objects and many observations. We will later see that this assumption does not hold in cases of occlusion, causing missing detections which will be handled by a separate mechanism.

4. We *assume* that the motion of road users (especially pedestrians) is a non-linear dynamic system that cannot be modeled by standard linear models such as constant velocity or constant acceleration. We also *assume* that the posterior distribution of the belief in the location of a road users can become skewed or multimodal. Under these assumptions we chose to apply sampling-based Bayesian filtering i.e., Particle Filters, which model beliefs using a set of weighted particles. Moreover, Particle Filters allow us to apply highly non-linear motion to each particle which in turn enables the tracking of unpredictable road user behavior e.g., a pedestrian walking, then turning around and jogging in another direction etc.

5. We *assume* that the operating characteristics of our sensors change not only over time, but also over the field of view. For example, the detection rate of a camera sensor varies locally over the area of the image plane, but it can also quickly deteriorate due to loss of brightness the moment the vehicle enters a tunnel or passes under a bridge. Under these assumptions tracking based on a single sensor model becomes sub-optimal, and we propose to use latent indicator variables that encode various modes of sensor operation to adjust the characteristics of the sensor model at runtime.

6. We *assume* that in cases of intermittent missing sensor observations, the missingness is not completely at random and it can be explained by the other available observations in the scene. Therefore, the proposed tracker uses multiple imputations to reconstruct any missing observation and continue the state estimation.

Despite the fact that this may seem like a lengthy list of assumptions, each one is based on a sound understanding of the characteristics of the objects we are attempting to track. Formally, our goal with tracking is to locate the unique road users in the scene, increase our confidence that they exist or do not over time, and reduce the uncertainty of their positions. We will do this using model-based reasoning and spatiotemporal sensor data. The proposed method extends the probabilistic concepts explained in the book "Probabilistic Robotics" by Thrun et al. [216] to tracking of multiple objects from a moving platform. Before we proceed with explaining the details, it is important to note the slight abuse of notation we will use $p(\mathbf{x}) = p_X(\mathbf{x})$ for the probability density function of the vector stochastic variable X, $p(\mathbf{x}|\mathbf{y}) = p_{X|Y}(\mathbf{x}|\mathbf{y})$ for the conditional probability density function of X conditioned on the vector stochastic variable Y and $p(\mathbf{x}, \mathbf{y}) = p_{X,Y}(\mathbf{x}, \mathbf{y})$ for the joint probability density function of X

and Y. For the same reason, instead of $X\left(t\right)$ the notation X_t will be used to indicate a random variable that changes over time. A single realization at time t will therefore be indicated as $\mathbf{x}_t$ instead of $\mathbf{x}\left(t\right)$.

Following the exposition of concepts in Chapter 2, we will use the tuple $(\mathbf{x}, \mathbf{g})$ to indicate the **state** of a road user at $\mathbf{x}$ with features $\mathbf{g}$; and **measurement** $\mathbf{z}^{(k)}$ from the k^{th}sensor consisting of: the location $\mathbf{u}^{(k)}$, size $\mathbf{s}^{(k)}$ of the detection in a sensor-specific coordinate system, reliability score or activation $a^{(k)}$ and other observational features $\mathbf{f}^{(k)}$. We further use H_1 to model the **state** pertaining to the existence of a road user within $\Omega(\mathbf{x}, \mathbf{g})$ and H_0 to model the absence of road users within this small region, see Section 2.5 for details. The measurement is gathered in the process of perception explained in Chapter 5 by which the system uses its sensors to obtain information about the state.

Another crucial element of tracking is the matching between current measurements and measurements of the from the past. In autonomous driving the ego-vehicle is in motion which causes the measurements $\mathbf{z}_t^{(k)}$ at each time step t to be observed from a different point of view. Since we rely on sensors to estimate the ego-position of the vehicle carrying the sensors, the error in the coordinate transform estimate between two time moments can propagate into the tracker. The proposed system uses coordinate transforms estimated by our own lidar odometry algorithm, which given the accuracy demonstrated in Chapter 3, has negligible error in the short term. Therefore, we will assume that each observation $\mathbf{z}_{l,t}^{(k)}$ can be translated perfectly back into a global coordinate system.

Assuming each road user is tracked independently from the rest, the following single-object tracking analysis applies to every individual track. The sequence of states and measurements time t_0 up until time t, $(t_0 \leq t)$, is summarized with the notation:

$$\begin{aligned} \mathbf{x}_{t_0:t} &= \mathbf{x}_{t_0}, \mathbf{x}_{t_0+1}, \mathbf{x}_{t_0+2}, ..., \mathbf{x}_t, \\ \mathbf{z}_{t_0:t} &= \mathbf{z}_{t_0}, \mathbf{z}_{t_0+1}, \mathbf{z}_{t_0+2}, ..., \mathbf{z}_t, \end{aligned} \tag{6.1}$$

where the evolution of state and measurements is governed by probabilistic laws. The state $\mathbf{x}_t$ is generated stochastically from the state $\mathbf{x}_{t-1}$ under the following law:

$$\mathbf{x}_t = f_t\left(\mathbf{x}_{t-1}\right) + \mathbf{v}_t, \tag{6.2}$$

where we use the generic function notation $f_t\left(.\right)$ to indicate the state transition model which is applied to the previous state $\mathbf{x}_{t-1}$ and $\mathbf{v}_t$ is the process noise (more details on this in Section 6.5). We use the probabilistic law to characterize the evolution of the state: $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ which is called the **state transition probability**. Note that the features $\mathbf{g}$, unique to each object, are absent from this model and assumed to not change over time. This is of course true for rigid objects such as cars whose shape stays the same over time, but for other categories such as pedestrians $\mathbf{g}$ can have slight variations. For example,while walking the 3-D bounding box of a person changes its dimensions due to the moving arms and legs. For simplicity, in our approach we use the same, class-specific shape $\mathbf{g}$ for every object. For the class pedestrian $\mathbf{g}$ is the

maximum spanning 3-D bounding box of an average person which is derived from a statistical model. The process by which measurements are being generated is given by the measurement process:

$$\mathbf{z}_t = h_t^{(k)}\left(\mathbf{x}_t, \mathbf{g}\right) + \mathbf{w}_t^{(k)}, \tag{6.3}$$

where $h_t^{(k)}\left(.\right)$ is the observation model of the k^{th} sensor and $\mathbf{w}_t^{(k)}$ is the observation noise. It is appropriate to think of measurements as noisy projections of the true, unknown state. After a sensor has made a detection, we can use $p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{g})$ to test the support of a state $(\mathbf{x}_t, \mathbf{g})$ given this observational evidence $\mathbf{z}_t$. We will refer to this function as the **measurement likelihood** or **sensor model** which is the joint probability of the observed data viewed as a function of the parameters of the chosen statistical model.

Under the Markov assumption no variables prior to $\mathbf{x}_t$ may influence the stochastic evolution of future states, unless this dependence is mediated through the state $\mathbf{x}_t$. Temporal processes that meet these conditions are commonly known as **Markov chains**. The state at time t is stochastically dependent on the state at time $t-1$ through the model explained in Eq. (6.2). The measurement $\mathbf{z}_t$ depends stochastically on the state at time t through the observation model in Eq. (6.3). Such a temporal generative model is also known as hidden Markov model (HMM) or dynamic Bayes network (DBN).

Belief reflects the system's internal knowledge about the state of the objects it is tracking. We therefore distinguish the true states from the internal belief with regards to those states. Following the probabilistic framework, beliefs are represented through conditional probability distributions. A belief distribution assigns a probability (or density value) to each possible hypothesis with regards to the true state. In object tracking we want to recall true objects by maximizing the belief in hypotheses that such objects exist, and if they do, minimize the uncertainty about their location. Due to the computational difficulties when optimizing the two beliefs simultaneously, as we will see later, the proposed method decouples the belief in the object's existence from the belief in object's location.

Thus, for every object we formulate two sets of hypotheses, first the existence hypotheses: $\{H_0 = 0, H_1 = 1\} : H\left(\mathbf{x}, \mathbf{g}\right) \triangleq \max_{x \in \Omega(\mathbf{x},\mathbf{g})} \mathrm{o}\left(\mathbf{x}'\right)$, which equals 1 if at least one road user with physical features $\mathbf{g}$ is present in $\Omega(\mathbf{x}, \mathbf{g})$ which is a specific region near $\mathbf{x}$, and 0 if no road user is present in that region; and second, the location hypotheses: given that $H_1$is more likely than H_0 we seed the region around $\mathbf{x}$ with weighted location hypotheses $w^{(i)}\delta_{\mathbf{x}^{(i)}}\left(d\mathbf{x}^{(i)}\right)$. The existence hypotheses H_0 and H_1 are maximized using the detector activations $a\left(.\right)$ as evidence evaluated at the hypothesis location $\mathbf{x}$ and shape $\mathbf{g}$. Contrarily, the location hypotheses are maximized using the detector locations $\mathbf{u}^{(k)}$, sizes $\mathbf{s}^{(k)}$ and the features $\mathbf{f}$ at each location $\mathbf{x}^{(i)}$.

Both the existence and location beliefs are distributions modeled as the posterior probabilities over state variables conditioned on the respective observational evidence.

We will denote the belief in object's existence as:

$$\mathrm{bel}\left(H\left(\mathbf{x},\mathbf{g}\right);a\right)=\frac{p\left(H_1|a^{(0)},...,a^{(K-1)}\right)}{p\left(H_0|a^{(0)},...,a^{(K-1)}\right)}, \tag{6.4}$$

where the dividend is the posterior probability that the object with shape $\mathbf{g}$ located at $\mathbf{x}$ does exist, and the divisor is the probability of the null hypothesis for $(\mathbf{x},\mathbf{g})$ respectively. The activation evidence $a^{(k)}$ has been collected from the K sensors at sensor coordinates $\mathbf{u}$ and shapes $\mathbf{s}$ that lie closest to $(\mathbf{x},\mathbf{g})$. Assuming that an object does exists within $\Omega\left(\mathbf{x},\mathbf{g}\right)$, the belief in this object's position is given by the distribution:

$$\mathrm{bel}\left(\mathbf{x},\mathbf{g},H_1;\mathbf{u},\mathbf{s}\right)=p\left(\mathbf{x},H_1|\mathbf{u}^{(0)},...,\mathbf{u}^{(K-1)},\mathbf{s}^{(0)},...,\mathbf{s}^{(K-1)}\right). \tag{6.5}$$

Before we incorporate the measurement $\mathbf{z}_t$, both beliefs go through a mutation step independent of the outcome of the sensor observation (referred to as **state transition** or **motion model**) giving the belief estimates for Eq. (6.4) and Eq. (6.5). The mutation step for the belief of existence is trivial since hypotheses that exist continue to exist and hypotheses that do not exist continue to not exist: the predicted belief before observing the sensor activations is simply transferred from the belief in the previous time step. The belief for the object location is however predicted using the motion model in Eq. (6.2). In the context of probabilistic filtering this probability distribution is often referred to as a **prediction**. These prediction and update steps constitute exactly the Bayes filter which we will explain in more detail in the following. The Bayes filter applies the state transition model to predict the state at time t based on the previous state posterior, before incorporating the measurement at time t. Calculating the current belief from the current observations and the predicted belief in the past is called a correction or a measurement **update**. For optimizing the belief in the object's location we will use a standard Bayes filter where the state is a real valued random vector, while for optimizing the belief in the object's existence we will use a binary Bayes filter with a static state. Our method uses the Particle Filter implementation of the Bayes filter which will be explained in detail in the following.

6.3.1 Object existence estimation with a Binary Bayes filter

When tracking objects in noisy environments, false sensor observations often create hypotheses pertaining to clutter that resembles the object of interest. Tracking exploits the temporal persistence of detections around real objects and the randomness of clutter in order to maximize the belief in the existence of real objects and suppress the belief in the existence of false hypotheses. In this section we propose a probabilistic method which, based on the observational evidence, computes how likely are we tracking a real object as opposed to tracking clutter. We are mainly interested in answering the question whether the observational evidence better supports H_1-the existence of a true object or H_0-a null-hypothesis. Logically, this hypothesis h

is best modeled as a binary (Bernoulli) state variable where only one of the two values (H_0, H_1) can be correct for the life-cycle of a tracked object, and moreover, the two values are mutually exclusive:

$$\Pr(h = H_0) = r = 1 - \Pr(h = H_1). \tag{6.6}$$

We will use a Bayes filter to maximize the belief in this state based on the observations over time, $\mathrm{bel}_t(h; a_{1:t})$. Another example of binary Bayes filters with static state are occupancy grid maps, which we discussed when computing the ego-motion in Chapter 3.

Formally, the belief for the existence of an object at a given position $\mathbf{x}$ with features $\mathbf{g}$: $\mathrm{bel}\left(h; a^{(k)}(\mathbf{u}, \mathbf{s})\right)$, is a function of the detector activations $a^{(k)}(\mathbf{u}, \mathbf{s})$ (in sensor coordinates) pertaining to this object over time. Note that in practical applications we usually limit the analysis only to sensor evidence at positions$(\mathbf{u}, \mathbf{s})$ that closely match the location of the object in sensor coordinates. The belief can be computed from all previous activations of the K sensors belonging to this hypothesis:

$$\mathrm{bel}_t(h; a_{1:t}) = \frac{p\left(H_1 | a_{1:t}^{(0)}(\mathbf{u}, \mathbf{s}), ..., a_{1:t}^{(K-1)}(\mathbf{u}, \mathbf{s})\right)}{p\left(H_0 | a_{1:t}^{(0)}(\mathbf{u}, \mathbf{s}), ..., a_{1:t}^{(K-1)}(\mathbf{u}, \mathbf{s})\right)}, \tag{6.7}$$

where the lack of a time index for the state h reflects the fact that the state does not change for the duration of the life cycle of an object. For numerical stability, this belief is commonly implemented as a log-odds $\Lambda_t(h)$. The log-odds of state h then assumes values in the range $[-\infty, \infty]$. The Bayes filter for updating beliefs in the log-odds representation, see Eq. (2.12), is computationally effective as it avoids truncation problems that arise for probabilities close to 0 or 1. This algorithm uses an inverse measurement model in the form of $p(h|\mathbf{z})$ instead of the familiar forward model $p(\mathbf{z}|h)$ (recall that $\mathbf{z}$ consists of $\mathbf{u}$ and $\mathbf{s}$, Eq. (2.1)). The inverse measurement model specifies a distribution over the (binary) state variable h as a function of the measurement $\mathbf{z}$, or more specifically the activations from all K sensors: $a^{(k)}(\mathbf{u}, \mathbf{s})$.

Inverse models are often used in situations where measurements are more complex than the state. Here the state is an extremely simple binary variable, but the space of all measurements is huge. It is easier to devise a function that calculates a probability of a true object from a camera image, than describing the distribution over all camera images that show a true object. In other words, it is easier to implement an inverse than a forward sensor model. The belief is then simply recovered from the log-odds ratio $\Lambda_t(h)$ as:

$$\mathrm{bel}_t(h; a_{1:t}) = 1 - \frac{1}{1 + \exp(\Lambda_t(h))}. \tag{6.8}$$

In practice, the binary Bayes filter for the positive and negative hypothesis values can be computed recursively [216]. In order to improve the readability of the following analysis and be consistent with notation in object tracking literature, we make a slight

abuse of notations where we substitute $\mathbf{z}_{1:t}$ for $\left\{a_{1:t}^{(0)}\left(\mathbf{u},\mathbf{s}\right),...,a_{1:t}^{(K)}\left(\mathbf{u},\mathbf{s}\right)\right\}$, thus:

$$p\left(H_1|\mathbf{z}_{1:t}\right)=\frac{p\left(H_1|\mathbf{z}_t\right)p\left(\mathbf{z}_t\right)p\left(H_1|\mathbf{z}_{1:t-1}\right)}{p\left(H_1\right)p\left(\mathbf{z}_t|\mathbf{z}_{1:t-1}\right)}, \tag{6.9}$$

and by analogy for the negative hypothesis:

$$p\left(H_0|\mathbf{z}_{1:t}\right)=\frac{p\left(H_0|\mathbf{z}_t\right)p\left(\mathbf{z}_t\right)p\left(H_0|\mathbf{z}_{1:t-1}\right)}{p\left(H_0\right)p\left(\mathbf{z}_t|\mathbf{z}_{1:t-1}\right)}, \tag{6.10}$$

where their ratio can be simplified to the following form:

$$\frac{p\left(H_1|\mathbf{z}_{1:t}\right)}{p\left(H_0|\mathbf{z}_{1:t}\right)}=\frac{p\left(H_1|\mathbf{z}_t\right)}{1-p\left(H_1|\mathbf{z}_t\right)}\frac{p\left(H_1|\mathbf{z}_{1:t-1}\right)}{1-p\left(H_1|\mathbf{z}_{1:t-1}\right)}\frac{1-p\left(H_1\right)}{p\left(H_1\right)}. \tag{6.11}$$

The log-odds ratio of the belief at time t, denoted as $\Lambda_t\left(h\right)$, is then computed using the following recursion:

$$\Lambda_t\left(h\right)=\Lambda_{t-1}\left(h\right)+\ln\frac{p\left(H_1|\mathbf{z}_t\right)}{1-p\left(H_1|\mathbf{z}_t\right)}-\ln\frac{p\left(H_1\right)}{1-p\left(H_1\right)}, \tag{6.12}$$

where $p\left(H_1\right)$ is the prior probability of the state h for the value 1. Each measurement update involves the addition of the prior (in log odds form). The prior also defines the log odds of the initial belief:

$$\Lambda_0\left(h\right)=\ln\frac{p\left(H_1\right)}{1-p\left(H_1\right)}. \tag{6.13}$$

For a more detailed analysis please refer to chapter 4.2 of [216].

The proposed tracker uses this log-odds existence ratio as a measure of the quality of each track during track maintenance. For example, a track can be declared a false positive and removed from further processing if the log-odds of its existence drops below a predefined track removal threshold. Moreover, most tracking benchmarks require a track existence score of some sort to be provided in order to compute tracking accuracy at various sensitivities. The binary Bayes filter therefore is a very useful tool to optimally integrate detection beliefs over time.

A practical implementation of the binary Bayes filter requires the prior object existence probability $p\left(H_1\right)$ and the inverse measurement function $p\left(h|\mathbf{z}\right)$. The proposed system uses a prior which is inferred from labeled training data using the characteristics of the employed object detector. The inverse measurement function $p\left(h|\mathbf{z}\right)$ can be modeled using a labeled training dataset as well. For example, after applying the CenterPoint object detector on the nuScenes [168] we matched all detections to ground truth labels and split them into true positives H_1 and false positives H_0 using a gating function $\Omega\left(\mathbf{x},\mathbf{g}\right)$. The probability $p\left(H_1|\mathbf{z}\right)$ can be represented as a histogram of true positive vs. total number of detections with a certain detection score range. After applying the same analysis for the false positives, we can compute the

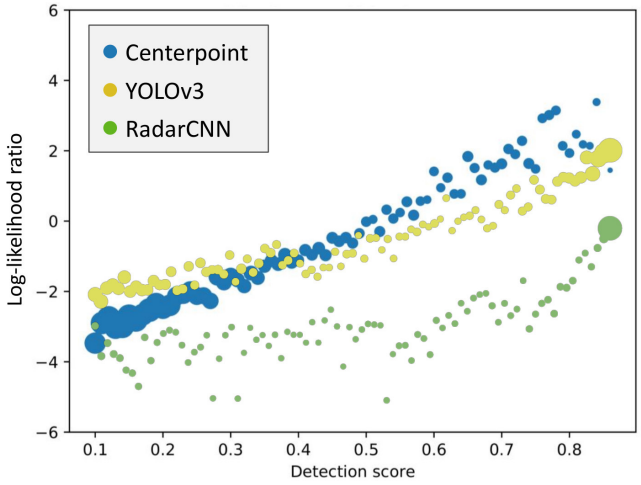


Figure 6.2: Inverse measurement models for the object detectors CenterPoint, YOLOv3 and the proposed RadarCNN. The markers represent the log-likelihood ratios, Eq. (2.11), of the three detectors for various detection scores respectively. Data from the IMEC training dataset.

ratio of true positives and false positives for each detector activation score. The plot shown in Figure 6.2 represents the proportion between true positives and false positives at several detection score bins for the detectors Centerpoint [51], YOLOv3 [24] and RadarCNN [16]. An object detector which achieves a good class separability will have a monotonic log-likelihood ratios relative to its activations, which as can be seen in Figure 6.2 is the case for all considered detectors.

Finally, in multi-sensor systems where the same object is detected by several object detectors, it becomes important to accurately model the joint belief of existence given the detections from different sensors at time t. Assuming the sensors are operating independently, it is reasonable to assume statistical independence of the detections. Therefore, the log-odds ratio at time t Eq. (6.12) in a multi-sensor setup is computed by adding the log odds of the individual sensors:

$$\Lambda_t(h) = \Lambda_{t-1}(h) + \sum_{k=0}^{K-1} \ln \frac{p\left(H_1|\mathbf{z}_t^{(k)}\right)}{1 - p\left(H_1|\mathbf{z}_t^{(k)}\right)} - \ln \frac{p(H_1)}{1 - p(H_1)}, \tag{6.14}$$

where K is the number of sensors that perceive the same object. This means that the evidence of existence about the same object is summed along all sensors to form the current log-likelihood. The more sensors perceive the same object, the higher the belief in its existence and vice versa. In the case when a missing detection, the constant log-likelihood ratio for that sensor is applied. This ratio is derived empirically from the detector recall rate at the lowest detection threshold.

6.3.2 Object location estimation with a Bayes filter

The Bayes filter is a two-step probabilistic optimization method for maximizing the belief of a state variable using observations and a model of the process. Assuming a Markov process, the Bayes filter algorithm *recursively* calculates the belief from a prior, a motion model and a measurement. In this section we show the correctness of the Bayes filter applying it to estimate the location $\mathbf{x}_{1:t}$ of a tracked object from observations over time $\mathbf{z}_{1:t}$, assuming that the object exists. For notational simplicity, in the following we will drop the object shape $\mathbf{g}$ (which in principle are always present), its existence hypothesis H_1 and simplify the observation vectors $\mathbf{u}$ and $\mathbf{s}$ using a unified notation $\mathbf{z}$. We can thus rewrite Eq. (6.5) as:

$$\mathrm{bel}\left(\mathbf{x}_t; \mathbf{z}_{1:t}\right) = p\left(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{z}_{1:t}\right). \tag{6.15}$$

The Bayes filter computes the belief $\mathrm{bel}(\mathbf{x}_t; \mathbf{z}_{1:t})$ at time t using the belief $\mathrm{bel}(\mathbf{x}_{t-1}; \mathbf{z}_{1:t-1})$ at time $t-1$ as a prior. The algorithm then applies a motion model to predict $\widehat{\mathrm{bel}}(\mathbf{x}_t; \mathbf{z}_{1:t-1})$ and finally, integrates the most recent measurement $\mathbf{z}_t$. The prediction $\widehat{bel}(\mathbf{x}_t; \mathbf{z}_{1:t-1})$ that the tracked object is at location $\mathbf{x}_t$ is obtained by the integral (sum) of the product of two distributions: the prior assigned to $\mathbf{x}_{t-1}$, and the probability that the state induces a transition from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$. This update step is called the control update, or prediction. In the measurement update step, the Bayes filter algorithm multiplies the prediction by the likelihood that the measurement $\mathbf{z}_t$ may have been observed. The resulting product is generally not a probability as it may not integrate to 1. Hence, the result is normalized, by virtue of the normalization constant η. This leads to the final belief estimate which is a probability distribution whose mode and statistical moments (mean, variance, etc.) will constitute the output of our algorithm and reported to the later steps in the autonomous vehicle control. In order to compute the posterior belief recursively, the algorithm requires an initial belief $\mathrm{bel}(\mathbf{x}_0)$ at time $t = 0$ as boundary condition. In Section 6.3.1 we will show hot to use a simplification of the Bayes filter to maximize the belief in the object existence using a static state.

Assuming that the object exists, the correctness of the Bayes filter algorithm can be shown by induction, showing that it correctly calculates the posterior distribution $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ from the corresponding posterior one time step earlier, $p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})$, under the assumption that we correctly initialized the prior belief $\mathrm{bel}(\mathbf{x}_0)$ at time $t = 0$. We can apply the Bayes' rule to the current belief i.e., the conditional probability defined in Eq. (6.4):

$$\mathrm{bel}(\mathbf{x}_t; \mathbf{z}_{1:t}) = p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \eta p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{z}_{1:t-1}) p(\mathbf{x}_t|\mathbf{z}_{1:t-1}), \tag{6.16}$$

where η is the probability of the evidence which in the analysis of the belief acts as a normalization constant. If we now analyze the first term in Eq. (6.16), assuming the state has a Markov property and $(\mathbf{x}_0, ..., \mathbf{x}_t)$ forms a Markov chain, we can ignore all

the past observations and controls:

$$\mathrm{bel}(\mathbf{x}_t; \mathbf{z}_{1:t}) = \eta p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{z}_{1:t-1}), \tag{6.17}$$

where $p(\mathbf{z}_t|\mathbf{x}_t)$ is the observation model. Then, by applying the law of total probability, we use the variable $\mathbf{x}_{t-1}$ to decompose the second factor, by integrating over all possible $\mathbf{x}_{t-1}$:

$$\mathrm{bel}(\mathbf{x}_t; \mathbf{z}_{1:t}) = \eta p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}) p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}) d\mathbf{x}_{t-1}. \tag{6.18}$$

The first factor of the integral in Eq. (6.18) represents the posterior probability of the current state given the previous state, all observations up until the previous time and all state transitions up to the present. Under the Markov chain assumption, the second factor can be further simplified as:

$$\mathrm{bel}(\mathbf{x}_t; \mathbf{z}_{1:t}) = \eta p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}) d\mathbf{x}_{t-1}, \tag{6.19}$$

where the resulting factor $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is simply the state transition probability discussed in the introduction and the second factor inside the integral is exactly the belief at time $t-1$:

$$\mathrm{bel}(\mathbf{x}_t; \mathbf{z}_{1:t}) = \eta p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) \mathrm{bel}(\mathbf{x}_{t-1}; \mathbf{z}_{1:t-1}) d\mathbf{x}_{t-1}. \tag{6.20}$$

The last factor in the recursion in Eq. (6.20) is the state transition likelihood, while the first factor is the update step:

$$\begin{array}{ll} \text{prediction:} & \widehat{\mathrm{bel}}(\mathbf{x}_t; \mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) \mathrm{bel}(\mathbf{x}_{t-1}; \mathbf{z}_{1:t-1}) d\mathbf{x}_{t-1}, \\ \text{update:} & \mathrm{bel}(\mathbf{x}_t; \mathbf{z}_{1:t}) = \eta p(\mathbf{z}_t|\mathbf{x}_t) \widehat{\mathrm{bel}}(\mathbf{x}_t; \mathbf{z}_{1:t-1}). \end{array} \tag{6.21}$$

Any implementation of the Bayes filter requires the following three probability distributions: the initial belief $p(\mathbf{x}_0)$, the measurement probability $p(\mathbf{z}_t|\mathbf{x}_t)$, and the state transition probability $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. Depending on how we model these probability distributions, the Bayes filter can be implemented using various algorithms such as the Kalman Filter (KF), Extended Kalman Filter (EKF), Particle Filter (PF), mixture models, etc. KF and EKF use Gaussian models for all probability distributions and a linear or piecewise linear motion models respectively. Particle filters, on the other hand, have the advantage to represent arbitrary probability distributions and motion models using sampling and pointwise evaluation of the observation model.

Although computationally efficient, Kalman Filters and their derivations produce inaccurate estimates when tracking the position of objects with unpredictable motion in the presence of occlusion, clutter, ambiguity and possible sensing failure. To illustrate why this is so, imagine a person that suddenly decides to change their direction of motion or gets occluded behind a parked vehicle. The belief of where this person

is expected to appear in the next time instances strongly depends on the motion and the interactions between the person and the environment. It is not difficult to imagine that in such situations the person is likely to be expected at multiple locations. Therefore, it is best to use multiple hypotheses about the expected position of the person in the future, each defining a corresponding density in the hypothesis space. The shape of this predicted belief can even become multi-modal if enough time passes without observing the person. Filters which assume uni-modality in the belief in the location of objects will simply be unable to accurately estimate the state in such scenarios. The Particle Filter, however, allows for modeling of the belief distribution as a density function with, potentially, arbitrary number of modes. Furthermore, as long as we can perform sampling from it, we can also use an arbitrarily complex motion model to represent the state transition. Due to the complex nature of the dynamics of the tracked objects, we choose to use Particle Filter as an implementation of the Bayes filter, and trade the increased computational load of Monte Carlo simulations in order to be able to accurately estimate the position of road users under difficult circumstances. Note that Particle Filters can be implemented to vary their sample size over time and adapt to the ambiguity of the observations.

6.4 Object tracking with a Particle Filter

The proposed method uses particle filters to compute the recursion in Eq. (6.21) in order to estimate the locations of road users from sensor observations over time. The particle filter is a sampling-based, non-parametric Bayes filter which we apply to track the positions of objects. This filter does not rely on a fixed functional form of the posterior, such as a Gaussian. Instead, it approximates the posterior distribution using a finite number of weights $w^{(m)}$ and particles $\mathbf{x}^{(m)}$, each roughly corresponding to a region in state space. The number of particles used to approximate the posterior can be varied and it influences the quality of the approximation. PF is well-suited to represent complex multi-modal beliefs and it is for this reason that is the method of choice for tracking the position of people when facing hard data association problems that yield separate, distinct hypotheses.

In particle filters, the samples of a posterior distribution are called particles and are denoted as the set of:

$$\begin{array}{c} \text{state samples: } \left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(N_{pts})}\right\}, \\ \text{sample weights: } \left\{w^{(1)}, w^{(2)}, ..., w^{(N_{pts})}\right\}, \end{array} \tag{6.22}$$

where each sample $\mathbf{x}_t^{(m)}$(with $1 \leq m \leq N_{pts}$) is a concrete instantiation of the state at time t, and $w_t^{(m)}$ is its corresponding weight. In order to make this into a probability distribution, we use a Dirac delta function positioned at each particle, $\delta\left(\mathbf{x} - \mathbf{x}_t^{(m)}\right)$, and constrain all weights to sum to one: $\sum_{m=1}^{N_{pts}} w^{(m)} \triangleq 1$. The particle filter then approximates the belief in Eq. (6.16) using the set of particles and weights as a weighted

sum:

$$p(\mathbf{x}_t|\mathbf{z}_t) \approx \sum_{m=1}^{N_{pts}} w_t^{(m)} \delta\left(\mathbf{x} - \mathbf{x}_t^{(m)}\right), \tag{6.23}$$

of N_{pts} Dirac deltas $\delta\left(\mathbf{x} - \mathbf{x}_t^{(m)}\right)$ positioned at the particle locations $\mathbf{x}_t^{(m)}$, and $w_t^{(m)}$ are the corresponding particle weights. At time t, each particle is distributed according to this posterior:

$$\mathbf{x}_t^{(m)} \sim p\left(\mathbf{x}_t|\mathbf{z}_{1:t}\right), \tag{6.24}$$

The complete set of particles represents a probability mass function, and the maximum a posteriori state estimate $\mathbf{x}_t^{(MAP)}$ can be obtained by searching for the mode of this discrete (potentially multi-modal) distribution, usually using clustering density estimation algorithms. The denser a sub-region of the state space is populated by samples, the more likely it is that the true state falls into this region. In practice, the number of particles N_{pts} is often a large number e.g., $N_{pt} = 10^3$ and it can be a function of t or other quantities such as the spread/skewness of the posterior distribution.

Being a Bayesian filter, the PF algorithm constructs the posterior recursively from the posterior one time step earlier. For a given observation $\mathbf{z}_t$, the observation model $p(\mathbf{z}_t|\mathbf{x}_t)$ can, usually, be obtained experimentally, details in Section 6.6. Additionally, the state transition or motion model characterizing the evolution of the state: $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is generally known or assumed using domain knowledge, details in Section 6.5. The only remaining challenge is how to compute the particle weights $w_t^{(m)}$. Since at time t the posterior distribution can not be sampled directly, but its likelihood can easily be evaluated, an approximation of this distribution can be obtained by means of importance sampling. Instead of sampling the posterior distribution, samples are drawn from any other distribution, called the proposal distribution $q\left(.\right)$, the support of which must include the support of the true posterior, [217]. The weights $w_t^{(m)}$ of the samples from this proposal distribution are then obtained by evaluating these samples using the observation model, the state transition model and the proposal function, such that the weighted set of samples approximates the true posterior distribution.

Let the proposal density, also called importance density, be $q\left(.\right)$. Recall Eq. (6.21), according to Bayes' rule, we can write the posterior as $p(\mathbf{x}_t|\mathbf{z}_t) = \eta p(\mathbf{z}_t|\mathbf{x}_t)\widehat{\text{bel}}(\mathbf{x}_t; \mathbf{z}_{1:t-1})$, where η is a normalization factor that is equal for all the samples drawn from $q\left(.\right)$. Then the importance weight of each particle can be calculated as:

$$w_t^{(m)} = \eta \frac{p\left(\mathbf{z}_t|\mathbf{x}_t^{(m)}\right) p(\mathbf{x}_t^{(m)}|\mathbf{x}_{t-1}^{(m)})}{q(\mathbf{x}_t^{(m)}|\mathbf{x}_{t-1}^{(m)}, \mathbf{z}_t)}, \tag{6.25}$$

where the enumerator is the product of the observation model, evaluated for the newest observation $\mathbf{z}_t$ at each particle $\mathbf{x}_t^{(m)}$, and the state transition model $p(\mathbf{x}_t^{(m)}|\mathbf{x}_{t-1}^{(m)})$

evaluated at each hypothesis location $\mathbf{x}_t^{(m)}$. Note that for brevity we are simplifying the observations $\mathbf{z}^{(k)} : \left(\mathbf{u}^{(k)}, \mathbf{s}^{(k)}, a^{(k)}, \mathbf{f}^{(k)}\right)$ and omitting the object features $\mathbf{g}$. The process of estimating the position of an object is generally reliant only on the positional part of the observation $\mathbf{u}^{(k)}$ and not so much on the shape, activations and the features. We also simplify the notation for detections from the multiple sensors with a single variable $\mathbf{z}_t$ where the joint sensor model indicated with $p\left(\mathbf{z}_t|\mathbf{x}_t^{(m)}\right)$ consists of the individual sensor models evaluated at each particle position: $p\left(\mathbf{z}_t|\mathbf{x}_t^{(m)}\right) = \prod_{k=0}^{K-1} p_{U_t^{(0)},\ldots,U_t^{(K-1)}|X_t}\left(\mathbf{u}_t^{(0)}, \ldots, \mathbf{u}_t^{(K-1)}|\mathbf{x}_t^{(m)}\right)$. We will give the details of two implementations: a **standard particle filter** using an optimal proposal function $q(.)$, and a **bootstrap particle filter** which approximates $q\left(.\right)$ resulting in a much lower algorithmic complexity. It is clear that the design of the proposal function $q\left(.\right)$ is also of great importance as its density serves for sampling, but the values also are used as a divisor for the particle weights in Eq. (6.25). A typical problem arising from a poorly chosen proposal function are the effects of particle *depletion* and particle *impoverishment*. Particle depletion refers to a situation where most of the particles have zero-weights and do not contribute towards the posterior distribution. In order to solve this problem particle filters apply Sequential Importance Sampling (SIS) where only particles with high weights are propagated to the next time step. This procedure solves the depletion problem, but causes particles to concentrate towards the peaks of the posterior distribution, called particle impoverishment. Next, we will present the general concepts of the two particle filter implementations and state their advantages and disadvantages as well as give a solution to the particle depletion and impoverishment problem. The general working principles of a particle filter are explained in Algorithm 6.1.

Standard Particle Filter

The optimal proposal distribution has been shown to be the one that minimizes the variance of the particle weights [218]. Not only do we need an analytic expression of the proposal distribution such that it can be sampled, we also need to calculate the actual state transition probability during weight assignment. An optimal proposal distribution which satisfies these requirements is the probability density output of the standard Kalman filter applied on each particle [217]. This approach has been successfully applied for the problem of hand tracking in video by researchers of our department [219]. Due to the Kalman filter's properties, the resulting proposal distribution represents an optimally weighted average of the motion model and the observed measurement. The standard Particle Filter using a proposal function mediated by Kalman Filter treats each particle as an independent Kalman Filter and runs the standard Kalman prediction/update equations with each new observation. For the sake of brevity, this analysis assumes that the reader is familiar with the details of the Kalman filter design as they are hereby performed in the standard way, see [219] for details. During importance sampling, the particle is drawn from a normal distribution

Initialize

- sample: $\mathbf{x}_{t_0}^{(m)} \sim p(\mathbf{z}_{t_0}|\mathbf{x}_{t_0}^{(m)})$
- set weights: $w_{t_0}^{(m)} := \frac{1}{N_{pts}}$;

Predict $\widehat{\text{bel}}(\mathbf{x}_t; \mathbf{z}_{1:t-1})$

- sample with replacement: $\mathbf{x}_t^{(m)}$ from the state transition distribution $p(\mathbf{x}_t^{(m)}|\mathbf{x}_{t-1}^{(m)})$

Measure $\mathbf{z}_t$

Update (standard PF) $\text{bel}(\mathbf{x}_t; \mathbf{z}_{1:t})$

- sample with replacement: $\mathbf{x}_t^{(m)}$ from the proposal distribution $q(\mathbf{x}_t^{(m)}|\mathbf{x}_{t-1}^{(m)}, \mathbf{z}_t)$
- compute weights $w_t^{(m)} = \eta \frac{p\left(\mathbf{z}_t|\mathbf{x}_t^{(m)}\right) p(\mathbf{x}_t^{(m)}|\mathbf{x}_{t-1}^{(m)})}{q(\mathbf{x}_t^{(m)}|\mathbf{x}_{t-1}^{(m)}, \mathbf{z}_t)}$

Update (bootstrap PF) $\text{bel}(\mathbf{x}_t; \mathbf{z}_{1:t})$

- compute weights $w_t^{(m)} = \eta p\left(\mathbf{z}_t|\mathbf{x}_t^{(m)}\right)$

MAP estimate

- compute $\mathbf{x}_t^{(MAP)} = \text{mode}\left[\sum_{m=1}^{N_{pts}} w_t^{(m)} \delta\left(\mathbf{x} - \mathbf{x}_t^{(m)}\right)\right]$

Re-sample

- Sample with replacement: $\mathbf{x}_t^{(m)} \sim \text{bel}(\mathbf{x}_t; \mathbf{z}_{1:t})$

Algorithm 6.1: Particle filter algorithm.

that serves as the proposal distribution:

$$q\left(\mathbf{x}_t^{(m)}|\mathbf{x}_{t-1}^{(m)}, \mathbf{z}_t\right) \sim \mathcal{N}\left(\bar{\mathbf{x}}_t^{(m)}, P_t^{(m)}\right), \tag{6.26}$$

where $\bar{\mathbf{x}}_t^{(m)}$ represents the Kalman filter's state estimate for the m^{th} particle and $P_t^{(m)}$ represents its covariance matrix. In the update step, Algorithm 6.1, the particle filter uses the current observation $\mathbf{z}_t$ to update the KF of each particle and draws N_{pts} new particles, each from their corresponding proposal distribution. Let $\dot{\mathbf{x}}_t^{(m)}$ be the state of the particle that is sampled from this proposal distribution of particle m at time and let $\ddot{\mathbf{x}}_t^{(m)}$ be the state estimate of the same particle, obtained by the state transition model.

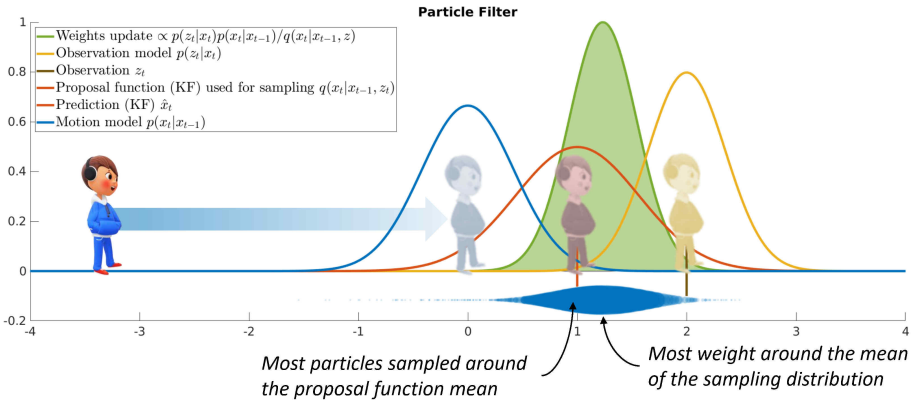


Figure 6.3: A one-dimensional example demonstrating the sampling and weight updates of a standard particle filter using the Kalman Filter estimate as a proposal function.

In the case of a Gaussian state transition model centered at the state that was predicted by a constant velocity model, then the state transition is:

$$p(\dot{\mathbf{x}}_t^{(m)}|\dot{\mathbf{x}}_{t-1}^{(m)}) \sim \exp(-\frac{1}{2}(\dot{\mathbf{x}}_t^{(m)} - \ddot{\mathbf{x}}_t^{(m)})^{\mathrm{T}} Q^{-1} (\dot{\mathbf{x}}_t^{(m)} - \ddot{\mathbf{x}}_t^{(m)})), \tag{6.27}$$

where Q is the KF process noise covariance matrix. The proposal function from which samples $\dot{\mathbf{x}}_t^{(m)}$ are drawn is defined as a Gaussian distribution:

$$q(\dot{\mathbf{x}}_t^{(m)}|\dot{\mathbf{x}}_{t-1}^{(m)}) \sim \frac{1}{\sqrt{\left|P_t^{(m)}\right|}} \exp(-\frac{1}{2}(\dot{\mathbf{x}}_t^{(m)} - \bar{\mathbf{x}}_t^{(m)})^{\mathrm{T}} \left(P_t^{(m)}\right)^{-1} (\dot{\mathbf{x}}_t^{(m)} - \bar{\mathbf{x}}_t^{(m)})), \tag{6.28}$$

computed as the distance between the motion model estimates $\dot{\mathbf{x}}_t^{(m)}$ and the KF estimates $\bar{\mathbf{x}}_t^{(m)}$, Eq. (6.26). The particle weights are then updated according to:

$$w_t^{(m)} = w_{t-1}^{(m)} \frac{p\left(\mathbf{z}_t|\dot{\mathbf{x}}_t^{(m)}\right) p(\dot{\mathbf{x}}_t^{(m)}|\dot{\mathbf{x}}_{t-1}^{(m)})}{q(\dot{\mathbf{x}}_t^{(m)}|\dot{\mathbf{x}}_{t-1}^{(m)}, \mathbf{z}_t)}. \tag{6.29}$$

The example shown in Figure 6.3 illustrates a one-dimensional example of a standard particle filter prediction and update on the horizontal axis. In this example the observation (yellow) greatly overshoots the expected position from the motion model (blue). The KF estimate after updating all particles is shown in red, while the proposal distribution is shown in green. The newly sampled particles, according to Eq. (6.28), are scattered bellow the plot with their size relative to their updated weights according to Eq. (6.25).

It is clear the in order to update the weights of particles, we need to compute the

positions of three different sets of particles: one using the state transition model, one by updating each particle as a KF and one by sampling from the proposal function. Then, the weight of each particle sampled from the proposal function can be computed using Eq. (6.29) which requires the evaluation of tree Gaussians. The combined computational load of these steps reduces its potential for practical applications where we have limited processing and memory resources. If we, however, model the proposal function to ignore the latest observation, and instead approximate it using the estimate from the previous step $q(\dot{\mathbf{x}}_t^{(m)}|\dot{\mathbf{x}}_{t-1}^{(m)}, \mathbf{z}_t) \approx p(\dot{\mathbf{x}}_t^{(m)}|\dot{\mathbf{x}}_{t-1}^{(m)})$ we come up with the Bootstrap Particle Filter which has much less complicated particle sampling and update steps.

Bootstrap Particle Filter

In practice it is often the case that the optimal proposal distribution $q(.)$ cannot be formulated analytically or too complex to compute in real-time given limited hardware resource. Most current particle filter implementations, such as the well known CONDENSATION algorithm [220], simply predict the new state using the state transition model as the proposal distribution, ignoring the latest observation $\mathbf{z}_t$:

$$q(\mathbf{x}_t^{(m)}|\mathbf{x}_{t-1}^{(m)}, \mathbf{z}_t) \approx p(\mathbf{x}_t^{(m)}|\mathbf{x}_{t-1}^{(m)}). \tag{6.30}$$

Particle filter implementations that employ this strategy are called bootstrap filters and are the most widely used particle filter variant. Bootstrap filters ignore the fact that the proposal distribution is conditioned on the latest observation. Instead, in the prediction step they assume that the current state is only a function of the previous state and the motion while the latest observation has no influence. In other words, the posterior is assumed to change smoothly over time thereby closely resembling the transition probability at each time step t.

Logically, using the state transition model as a proposal would yield two sets of particles $\ddot{\mathbf{x}}_t^{(m)}$ and $\dot{\mathbf{x}}_t^{(m)}$ that are equally distributed and we need only sample once according to Eq. (6.30). Then, plugging the state transition as a proposal function Eq. (6.25) makes for a greatly simplified weight update step $\left(q(\mathbf{x}_t^{(m)}|\mathbf{x}_{t-1}^{(m)}, \mathbf{z}_t) = p(\mathbf{x}_t^{(m)}|\mathbf{x}_{t-1}^{(m)})\right)$:

$$w_t^{(m)} = w_{t-1}^{(m)} p\left(\mathbf{z}_t|\mathbf{x}_t^{(m)}\right), \tag{6.31}$$

where the proposal distribution is canceled out, and as a result only a prediction using the state transition model $p(\mathbf{x}_t^{(m)}|\mathbf{x}_{t-1}^{(m)})$ and an update using the likelihood model $p\left(\mathbf{z}_t|\mathbf{x}_t^{(m)}\right)$ needs to be performed, see Algorithm 6.1.

Although bootstrap filters are the most widely used type of particle filter, failing to introduce the latest observation into the proposal distribution can cause problems. The worst case scenario for the bootstrap particle filter is a badly modeled outlier observation i.e., one which adheres poorly to the observation model. The example in

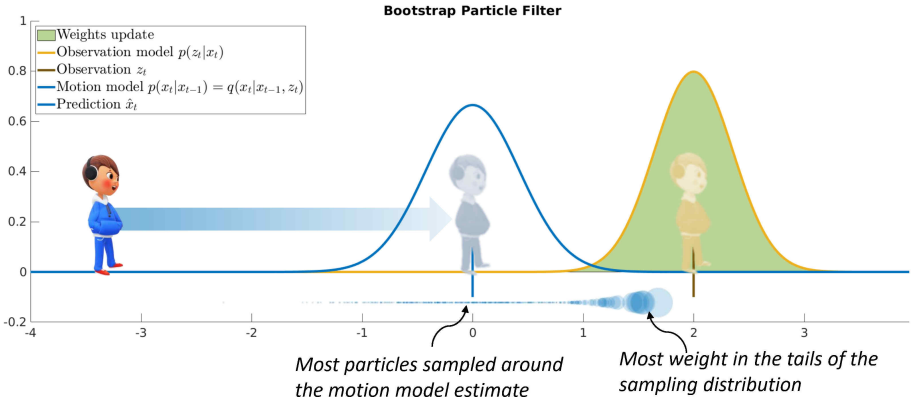


Figure 6.4: A one-dimensional example demonstrating the sampling and weight updates of a bootstrap particle filter using the state transition distribution as a proposal function. The person is expected at position: $\text{mode}\left[\widehat{\text{bel}}(x_t; z_{1:t-1})\right]$ (around 0), while an observation is measured at $z_t = 2$.

Figure 6.4 illustrates this issue through the same one-dimensional tracking scenario as we previously analyzed. The bootstrap particle filter uses the motion model to sample new particles which are scattered around the expected value, 0 in this example. Due to the limited number of samples there are only a few newly sampled particles which support the new evidence $z_t = 2$. Therefore, almost all of the particles will have near-zero weights with most of the weight concentrated around a few particles close to the observation. The problem becomes worse at the next time instance, $t + 1$, when particles with a low weight will probably be assigned an even lower weight, since the new estimate is based on the bad state approximation of the previous time instance t. Due to the limited computer precision, after a while, only few particles with a corresponding high weight will model the mode of the posterior, whereas most of the particle weights will corresponding to the tails of the posterior distribution will have zero weights.

The problem where many of the particles have zero weights, and are thus uninformative, is called particle depletion. Sparse sampling of the mode of the posterior distribution causes the filter to get easily distracted by noise towards the tails of the distribution (outliers). To solve this problem, resampling techniques are often used to replicate particles with high weights, while removing particles with a low weight. SIS particle filters that employ resampling are called Sequential Importance Re-sampling (SIR) filters. A widely used resampling scheme is multinomial resampling. In multinomial resampling, the set of new particles is sampled with replacement from the set of old particles. The probability that a particle with index m is picked at each sampling round is defined by its weight $w_t^{(m)}$. This way zero-weight particles are removed from the model and after re-sampling, the weights of all remaining particles are re-set to $\frac{1}{N_{pts}}$ such that the new sample set represents a probability distribution.

Although re-sampling solves the particle depletion problem, it introduces a new problem called particle impoverishment. After resampling, most samples are located on or around the peaks of the distribution, while few samples are available to represent the tails. As a result, the tracker fails to explore the complete search space and easily gets stuck on local maxima. As a compromise, most SIR filters only perform re-sampling when needed, instead of re-sampling at every time instance t. In order to determine when re-sampling should be performed, an estimate is needed for the performance of the particle filter state estimation at each time step. A common strategy is to use the Effective Sample Size (ESS) of the posterior:

$$ESS_t = \frac{1}{\sum_{m=1}^{N_{pts}} \left(w_t^{(m)}\right)^2},$$

and re-sample only when the ESS satisfies a certain fraction of the number of particles. If all particles are assigned the same weight, the effective sample size equals the real sample size. If a single particle has a weight of 1 and all other particles have a weight of 0, then the effective sample size is 1. In order to avoid impoverishment, most SIR filters therefore only re-sample if $ESS < \alpha N_{pts}$ with $\alpha \in [0, 1]$. In general, we want to re-sample as little as possible, while still being able to model the mode of the posterior density accurately. Hence, a proposal distribution $q(.)$ that closely matches the posterior distribution is extremely important as it greatly reduces the need for re-sampling. For a detailed mathematical derivation of the Particle Filter, see section 4.3 in [216] and chapter 3 in [217].

Application-specific considerations

The particle filter is designed to approximate the posterior probability distribution of a state variable (the position of a road user in our application) from sensor observations over time. Therefore, we need to devise a strategy on how to apply particle filters when we want to track several objects of interest in the scene. As discussed in the introduction, Section 2.7, we found that one tractable approach for real-time applications is to disentangle the states of every object into independent states tracked by independent particle filters. This bottom-up principle relies on association between detections and individual hypotheses and applies local sensor models that are confined to a small gating region $\Omega(\mathbf{x}, \mathbf{g})$ around each hypothesis. Although this simplification of the state-space allows for accurate tracking of multiple objects in most circumstances, it also causes problems under border cases of ambiguous observations where we deem the bootstrap PF is the preferred algorithm over the standard PF.

Whenever the objects of interest are positioned close to each other, it is difficult to estimate the correct association of measurements to hypotheses. Faulty association results in outliers (a track is assigned an observation from another object) which compared to noisy observations reduce the statistical power of the sensor model. In this situation, conditioning the PF proposal function on the latest (potentially outlying) observation as does the standard PF algorithm has the potential to incorrectly lead the

the posterior distribution estimate away from the true state. In object tracking this effect results in divergence of the track and identity switches between nearby objects. Therefore, besides it being less computationally complex, we are more motivated to use the simpler bootstrap PF. To further address the problems created by confusing data association, we shall implement a switching observation model method, details follow in Section 6.6.4.

Over time, the object motion of people is affected by many factors and is difficult to predict without accurate observations. Even predicting the position of moving people in the short-term is challenging around occluding objects. The problem becomes especially difficult when, due to the occluding objects, the person is impossible to detect causing intermittent loss of detection data. In such situations the tracked object can be expected to re-appear in the sensor view at multiple places with a low probability density. This means that the posterior density becomes less peaked and we need to use more particles and more computations which would not be the case if the motion of the object is predictable. Moreover, due to the imperfect recall of object detectors, intermittent loss of detection is to be expected even in normal viewing conditions. Whenever there are no detections, the proposal Eq. (6.28) is meaningless because there is no $\mathbf{z}_t$ to update the individual particle KFs. Therefore, both the standard PF and the bootstrap PF predict based on the motion model and have the same performance.

Finally, the quality of associating detections to tracks is dependent on how often the system receives data updates. Since we are tracking multiple objects that move in unpredictable ways, the longer the system waits for new data the less motion models can predict the object motion. Using the latest observation to condition the proposal function for sampling, the standard PF is theoretically better suited than the bootstrap PF. In order for these PF characteristics to translate into significant performance differences, the data update frequency has to be lower than the rate of change of the object motion i.e. its motion is highly predictable within the time frame between two detections. Typical camera/lidar/radar sensors produce tens to hundred of samples per second. The period between data updates is therefore relatively short and not enough for an object to change its motion before a new detection is made. This motivates us to suggest using the bootstrap PF over the standard PF, however, we propose the use of a stochastic, behavioral motion model which will be explained in the following section.

6.5 Motion model

In this analysis we will largely restrict to object tracking in planar environments. The pose (location and orientation) of objects in such environments is summarized by three variables: two-dimensional planar coordinates and an angular orientation. In the literature, this reference system is often referred to as a 2.5-D, defining the location using x and y, and the orientation using θ, forming the following vector: $(x, y, \theta)^T$. The orientation is referred to as heading direction and, for convention, follows the compass heading: $\theta = 0$ points into the direction of the y-axis. A pose without

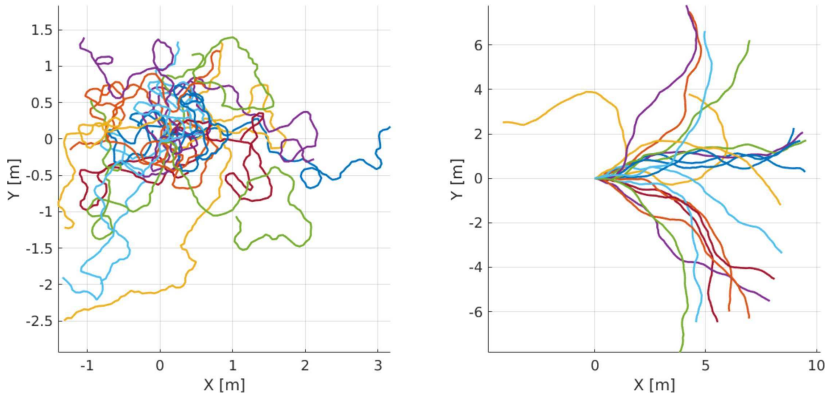


Figure 6.5: Visualization of 20 Monte Carlo realizations of a pedestrian moving according to the proposed behavioral motion model. Initial parameters: pose $\mathbf{x}_{t_0}^{(m)} = (0, 0, \pi/2)$ and velocity of $2Km/h$ (left) and $5Km/h$ (right).

orientation will be called location.

The probabilistic kinematic model, or motion model plays the role of the state transition model. Assuming that the object does exist at $\mathbf{x}_t$ and has a shape $\mathbf{g}$, $H(\mathbf{x}_t, \mathbf{g}) = H_1; \forall t$, and its shape doesn't change over time, the motion model is the conditional density $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ where $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$ are both 2.5-D object poses. This model describes the posterior distribution over kinematic states that an object assumes when exercising motion at $\mathbf{x}_{t-1}$. Since our perception system does not provide direct measurement of the object velocity, we will use a velocity motion model. The true nature of $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ explaining the motion of road users on the ground plane is dependent on both scene geometry as well as high-level reasoning which is difficult to model completely. Oftentimes in high frame-rate applications, the motion of road users (especially pedestrians and cyclists) is commonly modeled using a constant velocity motion model. This, however, is a gross underestimation of the unpredictable human motion which we will address with a novel, behavioral motion model.

The proposed transition probability density explores the state space through mutating the rotational velocity ω and a radial velocity v using a behavioral model. The radial velocity at time t is denoted as v_t while the rotational/tangential velocity is ω_t making the motion vector $(v_t, \omega_t)^T$. Positive rotational velocities ω_t induce a clockwise rotation (right turns) and positive radial velocities v_t correspond to forward motion.

Since we will be using particle filters, computing the conditional probability $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ needs to be performed by sampling $\mathbf{x}_t^{(m)}$ given the previous set of particles $\mathbf{x}_{t-1}^{(m)}$. The proposed sampling algorithm perturbs the motion vector parameters by noise, drawn from error parameters that we train from labeled road users motion in traffic. The noise parameters are then used to generate the sample's new pose consisting of a new location and a new orientation. Note that $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ does not apply on the shape of

For each particle $\mathbf{x}_{t-1}^{(m)} = \left(x_{t-1}^{(m)}, y_{t-1}^{(m)}, \theta_{t-1}^{(m)}\right)$ with velocity (v_{t-1}, ω_{t-1}):

- **sample:** $\hat{v}_t \sim \mathcal{N}\left(v_{t-1}, \alpha_1 v_{t-1}^2 + \alpha_2 \omega_{t-1}^2\right)$
- **sample:** $\hat{\omega}_t \sim \mathcal{N}\left(\omega_{t-1}, \alpha_2 v_{t-1}^2 + \alpha_3 \omega_{t-1}^2\right)$
- **compute:** $x_t^{(m)} = x_{t-1} - \frac{\hat{v}_t}{\hat{\omega}_t} \sin\theta_{t-1} + \frac{\hat{v}_t}{\hat{\omega}_t} \sin\left(\theta_{t-1} + \hat{\omega}_t \Delta t\right)$
- **compute:** $y_t^{(m)} = y_{t-1} - \frac{\hat{v}_t}{\hat{\omega}_t} \cos\theta_{t-1} + \frac{\hat{v}_t}{\hat{\omega}_t} \cos\left(\theta_{t-1} + \hat{\omega}_t \Delta t\right)$
- **compute:** $\theta_t^{(m)} = \theta_{t-1} + \hat{\omega}_t \Delta t$

return: $\mathbf{x}_t^{(m)} : \left(x_t^{(m)}, y_t^{(m)}, \theta_t^{(m)}\right)^T$

Algorithm 6.2: Algorithm for sampling poses $\mathbf{x}_t^{(m)}$ from an initial pose $\mathbf{x}_{t-1} = (x_{t-1}, y_{t-1}, \theta_{t-1})^T$.

the object which remains the same throughout time. Thus, the sampling procedure implements the motion model that incorporates control noise in its prediction in the most straightforward way, summarized in Algorithm 6.2.

In order to train the parameters motion model parameters we used the labeled KITTI pedestrian dataset and analyzed the motion of road users through typical urban environments. In [7] we confirmed the hypothesis that pedestrians exert a motion behavior which is highly non-uniform. Moreover, we measured a distinct dependency between the person's walking speed v and the likelihood for change in their orientation $\Delta\omega$. Specifically, a static person is more likely to start moving in any direction, while a person in motion is likely to continue to walk in the same direction. The faster the walking pace, the less likely it is the person will change their heading.

The proposed behavioral motion model is a Gaussian random-walk where the longitudinal and lateral acceleration components (Δv and $\Delta\omega$) are sampled from zero-mean normal distribution. The variance of the longitudinal acceleration distribution is constant while for the lateral acceleration we use a variance factored on the current velocity magnitude, details in [7]. This model can be sampled by taking the steps explained in Algorithm 6.2. In Figure 6.5 we present the realization of several random-walk chains by sampling new states using $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ for a slowly moving (left) and a fast moving (right) pedestrian motion model. In this simulation we are tracking a single object purely on the motion model without applying a correction step with an observation. We let the simulation run until the object has traversed $10m$ and show the means of 20 estimated trajectories with unique colors. The distribution of the newly sampled poses varies greatly depending on the initial motion vector. A slowly moving pedestrian is more likely to change direction than a fast moving one.

In order to better illustrate the characteristics of the motion model, consider the example depicted in Figure 6.6. This experiments draws the new states for 10^5 parti-

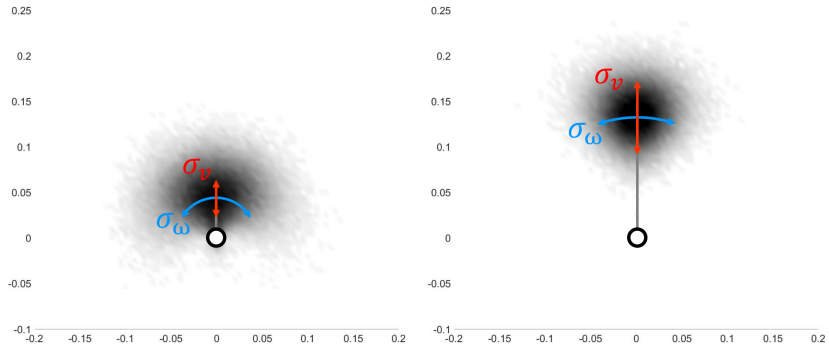


Figure 6.6: Visualization of the proposed behavioral motion model. The plots show the outcome of drawing 10^5 new samples from an initial state $\mathbf{x}_{t_0} = \left(0, 0, \frac{\pi}{2}\right)^T$ and a motion vector for a slowly moving pedestrian (left) and a pedestrian with a normal walking speed (right).

cles initialized at $\mathbf{x}_{t_0} = \left(0, 0, \frac{\pi}{2}\right)^T$ and shows the density of these new states. At each time step, the object in the left plot is assumed to be moving with an initial radial and angular velocity of $(2Km/h, 0\text{rad}/s)^T$, while the object in the right plot is moving with an initial velocity of $(5Km/h, 0\text{rad}/s)^T$ respectively. The left plot demonstrates the probability density $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ for a pedestrian walking at a slow pace of $2Km/h$ while the right plot demonstrates the same density for a person walking at a faster pace of $5Km/h$, respective to the single time step motion models shown in Figure 6.6.

The added benefit when using the proposed behavioral motion model is that the system can explore the state space much more effectively than using constant velocity or even constant acceleration models. Before the Particle Filter update with observational data, the location and orientation of each particle is mutated according to the motion of the tracked object. This becomes especially important in tracking with missing detections where the loss of sensor information for a longer period of time causes loss of tracking due to track divergence. In the experimental evaluation section, 6.9.3, we use simulations to measure the objective tracking performance of the proposed model implemented in a Particle Filter as opposed to a Kalman Filter with a linear motion model.

6.6 Observation models

In Section 6.3.1 we showed how to model the belief in object existence from multiple detectors using the observation model for detection confidence based on the K detector activations $\sum_{k=0}^{K-1} \text{llr}^{(k)}(a^{(k)}(\mathbf{x}))$. In this section we assume an object is always present $(H(\mathbf{x}, \mathbf{g}) = H_1)$ and analyze the observation models that update the belief in the location of a tracked object in Eq. (6.21). Decoupled from the belief in existence, these observation models describe the spatial measurement formation process i.e., the laws by which an object in physical space is projected into sensor space.

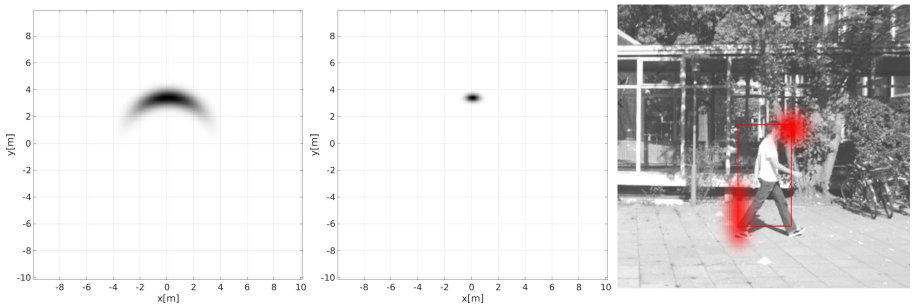


Figure 6.7: Example with a single pedestrian at $3.5m$ distance visualizing the observation models for radar, lidar and camera object detectors.

In the following we use the general observation model mapping an object $(\mathbf{x}, \mathbf{g})$ into a sensor measurements $\mathbf{z}$ that we defined earlier in Eq. (2.3). Note that the observation function is also time-varying and in this analysis we will assume that it can change between multiple known shapes which we will analyze in detail in Section 6.6.4. Since autonomous vehicles use a variety of different sensors, such as lidar, radar, or cameras, the specifics of the observation model depends on greatly the sensor: Imaging sensors are best modeled by 3-D to 2-D projective geometry, whereas radar and lidar sensors are best modeled by describing the propagation of the electromagnetic wave and its reflection on surfaces in the environment. An observation model, in this context, explains the uncertainty in the perceived location of an object. For a more detailed theoretical formulation the reader is referred to chapter 6 in [216].

In a multi-sensor perception system running object tracking, each track update needs to compute the K-sensor product: $\prod_{k=0}^{K-1} p_{U_t^{(0)},...,U_t^{(K-1)}|H,X_t}\left(\mathbf{u}_t^{(0)}, ..., \mathbf{u}_t^{(K-1)}|\mathbf{x}_t\right)$. We hereby assume that the track does exist and drop H_1 as a conditioning value. In a Particle Filter, this update evaluates the said product of observation models for each particle $\mathbf{x}_t^{(m)}$ and each detection location $\mathbf{u}_t^{(k)}$, recall Eq. (6.25). When the scene contains a single object, computing the joint observation model is straight forward and yields accurate state estimation, however, when tracking multiple objects the association between multiple detections and multiple tracks can sometimes be ambiguous. In these situations it is not always guaranteed that the multi-sensor evidence matched to a hypothesis within $\Omega(\mathbf{x}, \mathbf{g})$ belongs to the correct track. Depending on the size of $\Omega(\mathbf{x}, \mathbf{g})$ the joint-measurement model might not be entirely accurate i.e., it is quite possible that a detection belongs to another object (a false positive). This motivates us to propose an adaptive observation model that, instead of using a fixed observation model, allows the tracker to adapt the observation model to the spatial distribution of observations over time.

6.6.1 Uncertainty of the location of radar observations

Object detection by radar yields observations $\mathbf{z}^{(radar)} = (\mathbf{u}, \mathbf{s}, a, \mathbf{f})$ consisting of the object center in polar coordinates $\mathbf{u}^{(radar)} : (\rho, \theta)$, the object shape $\mathbf{s}$, an activation a and features $\mathbf{f}$. When modeling the positional uncertainty of a radar detection for object tracking we are generally interested in the relationship $p(\mathbf{u}_t|\mathbf{x}_t)$ where the shape and features of a target are of secondary importance. Due to its intrinsic ability to measure range, the measured object range is fairly accurate and independent on the distance to the sensor. However, the measured object azimuth is dependent on the number of antennas in the radar array as well as external factors such as multi-path fading. Since the radar uncertainty differs greatly along the range and azimuth we will define the radar observation model separately along these two dimensions.

Our proposed radar detector is designed specifically to reduce this uncertainty by using a wide receptive field along the azimuth dimension and feedback information from other sensors. In Section 5.7.1 we showed that radar detection greatly benefits from camera feedback, however, when the feedback information is unavailable the radar detector has reduced azimuth accuracy. The uncertainty of detections from the proposed object detector is therefore bi-modal. Depending on the quality of the camera feedback, the uncertainty along the azimuth can be either high (if no feedback is present) or greatly reduced if the camera detects the same object. The radar observation models for the position of a detection are formally described as:

$$\rho = h_{\rho}^{(radar)}\left(\mathbf{x}_t, \mathbf{g}\right) + w_{\rho,t}^{(radar)}, \tag{6.32}$$

$$\theta = \begin{cases} \arctan\left(\frac{y_t}{x_t}\right) + \dot{w}_{\theta,t}^{(radar)}, & \text{Radar CNN with feedback,} \\ \arctan\left(\frac{y_t}{x_t}\right) + \ddot{w}_{\theta,t}^{(radar)}, & \text{Radar CNN without feedback,} \end{cases} \tag{6.33}$$

where the range mapping is simply the Euclidean distance: $h_{\rho}^{(radar)}\left(\mathbf{x}_t, \mathbf{g}\right) = \left\|\mathbf{x}_t\right\|_2$, and the scalars $w_{\rho,t}^{(radar)}$, $\dot{w}_{\theta,t}^{(radar)}$ and $\ddot{w}_{\theta,t}^{(radar)}$ are Gaussians whose parameters are learned offline using labeled data which is unseen during training. The shape of these models was confirmed in controlled experiments with a single target (a person) moving across the Radar field of view where its true position (measured with lidar) was compared to the detection data from the Radar. Usually the variance along the azimuth is much smaller in situations where the radar CNN has access to feedback information. The example shown on the left plot in 6.7 illustrates the radar CNN observation model $p(\mathbf{u}_t|\mathbf{x}_t)$ for a target at 3.5m in front of the sensor. In this example the radar doesn't have any additional feedback information which makes the uncertainty along the azimuth much greater than the range uncertainty.

6.6.2 Uncertainty of the location of lidar observations

A lidar detection $\mathbf{z}^{(lidar)} = (\mathbf{u}, \mathbf{s}, a, \mathbf{f})$ consists of a 3-D bounding box defined by its center $\mathbf{u} : (x, y, z)$, its size $\mathbf{s} : (w, d, h)$, and an activation a and features $\mathbf{f}$.

Depending on how the lidar object detector is implemented, the bounding box can also contain a 3-D orientation vector or an orientation scalar indicating its compass heading. When modeling the positional uncertainty of detected objects we will focus on the measurement of the object center $\mathbf{u}$, while the orientation, size, activation and features are of secondary importance. Assuming the lidar detection is a true positive, the positional uncertainty of the observed vector $\mathbf{u}$ is a function of the true position $\mathbf{x}$, largely influenced by the lidar analysis algorithm, non-maximum suppression of the raw activations as well as the quality of the raw lidar data points. Due to the properties of propagation of the laser beams, objects at distance are expected to be sparsely scanned resulting in much less data for object detection. Contemporary lidar sensors such as the Velodyne HDL series have a nominal range error of 3cm-5cm, however, most state of the art lidar object detectors such as [51] yield detections with an order of magnitude lower positional accuracy.

Even though lidar object detectors operate on extremely accurate range data and are extensively trained to minimize positional errors, the accuracy of detections varies across the detection field. For example, the lidar can only perceive the front-facing surface of objects which makes estimating the object center dependent on trained geometrical models and contextual information. Given a large enough dataset, object detectors perform this task fairly accurately for close-by objects, however, further away objects are scanned more sparsely making the fitting of geometrical shapes less accurate. Positional uncertainty in the context of object detection refers to the inability for a range sensor to reliably measure the center of objects, and is represented as noise. This noise has predictable characteristics that can be learned off-line from training data. For the detector Centerpoint [51] we have:

$$\mathbf{u}_t^{(lidar)} = h^{(lidar)}\left(\mathbf{x}_t, \mathbf{g}\right) + \mathbf{w}_t^{(lidar)}, \tag{6.34}$$

where the mapping is a linear function: $h^{(lidar)}\left(\mathbf{x}_t, \mathbf{g}\right) = \mathbf{x}_t$ and the noise is zero-mean with range dependent covariance matrix: $\mathbf{w}_t \sim \mathcal{N}_2\left(\mathbf{0}, \mathbf{\Sigma}; \mathbf{x}_t\right)$. This model was confirmed using the nuScenes dataset where we compared lidar detections to true object positions of various classes that were manually labeled by human annotators.

Since we are using a rotating lidar which has the same sensing properties in all direction, the sampled data is isotropic along the azimuth and decays with range. Thus the covariance matrix $\mathbf{\Sigma}$ is a diagonal matrix defined by the range of the object and a correcting coefficient r: $\mathrm{diag}\left(r\left\|\mathbf{x}_t\right\|, r\left\|\mathbf{x}_t\right\|\right)$. The characteristics of this model are illustrated with the example shown on the middle plot in Figure 6.7. In this example there is a single pedestrian at 3.5m from the sensor. The plot shows the density of the conditional probability $p(\mathbf{u}_t|\mathbf{x}_t)$ using the model in Eq. (6.34).

6.6.3 Uncertainty of the location of camera observations

Depending on the implementation of the camera detector, camera observations can be defined in image coordinate space or in 3-D if we apply object ranging using depth images as we saw in Section 4.3. Camera object detections which are ranged

using depth images exert the positional uncertainty of the ranging sensor which was used to generate the depth maps. In this case we can generally use the positional uncertainty model defined by Eq. (6.34). When no ranging information is available to the camera, the observations $\mathbf{z}^{(cam)} = (\mathbf{u}, \mathbf{s}, a, \mathbf{f})$ are completely defined in image pixel coordinates. The position vector consists of the 2-D image bounding box edge points containing the image object boundaries. The observation model $p(\mathbf{u}_t|\mathbf{x}_t)$ in this case will be defined in terms of the image pixel uncertainties of a 2-D bounding box $\mathbf{u}^{(cam)} = (u_0, u_1, v_0, v_1)$ given an object position and shape in 3-D:

$$\mathbf{u}_t^{(cam)} = h^{(cam)}(\mathbf{x}_t, \mathbf{g}) + \mathbf{w}_t^{(cam)}, \tag{6.35}$$

where $h^{(cam)}(.)$ applies the pinhole camera model and perspective transformation, explained in Appendix (A), to project a 3-D bounding box defined by $(\mathbf{x}, \mathbf{g})$ into four corners defining a 2-D bounding box in image coordinates $\mathbf{u}^{(cam)}$. The uncertainty of the 2-D bounding box coordinates depends on several factors. Firstly, the accuracy of the camera extrinsic and intrinsic matrices used for the projection $h^{(cam)}(.)$ can cause small systemic skew of observed image bounding boxes. Second, occlusion causes parts of the object to be hidden from camera view resulting in a bounding box covering only the visible portion of the object. Lastly, difficult image textures and poor viewing conditions can cause the object detector to misinterpret the position/shape of the object bounding box. We model all of these effects as a 4-D noise vector $\mathbf{w}_t^{(cam)}$ which applies horizontal and vertical perturbations to the observed image bounding box corners $\mathbf{u}_t^{(cam)}$. In practice, $\mathbf{w}_t^{(cam)}$ consists of two 2-D Gaussians referring to the bottom left (u_0, v_0) and top-right (u_1, v_1) corners of an image bounding box. This model is in line with the probabilistic bounding boxes concept [221]. The parameters of these Gaussians are learned offline using the KITTI object detection dataset labeled for both the 3-D and 2-D position of objects in the scene and in the camera image. The right image shown in Figure 6.7 illustrates the camera model $p(\mathbf{z}_t|\mathbf{x}_t)$ for the YOLOv3 object detector. In this example, there is a single pedestrian at 3.5m which is visible to the camera. The positional uncertainties of an expected camera object detection are illustrated as two probability densities with red color and the most likely detection with a red bounding box.

6.6.4 Switching observation models

The spatial distribution of detections fused using K sensors follows a generalized, fused measurement model $\prod_{k=0}^{K-1} p_{U^{(0)},...,U^{(K-1)}|H,X}\left(\mathbf{u}^{(0)}, ..., \mathbf{u}^{(K-1)}|H_1, \mathbf{x}\right)$ which models the uncertainty of the measurement attributes of all sensors. Using a single model per sensor has the advantage that the model will fit most situations and will be optimal under the general case of conditions. However, in border cases when the characteristics of the observations change, the general sensor models are no longer optimal. At run-time, observation conditions can change dramatically due to changing light levels, occlusions, transmission channel errors, battery power level. A camera will react to such changes by adjusting its integration time, aperture, sensitiv-

ity, white balance etc. which inadvertently results in detection characteristics different from the nominal ones. Any sensor, in general, can stop working altogether in cases of mechanical failure caused by vibrations, overheating, dust contamination, etc. Additionally, manufacturing defects, overheating, physical or cyber-attacks can alter the characteristics of measured data. Lastly, real-world sensor configurations employed to measure a wide area of interest often have "blind spots" where information is missing by design.

Another factor which can influence the current observation likelihood to be different from a nominal one is the in-between step of ranging camera detections using depth images. When fusing these ranged detections with other sensor data we use the positional uncertainty model of the range sensor. However, due to depth image sparsity and errors in its reconstruction, camera detections will not always get the correct range. For example, an object in the distance with a small image bounding box may fall between two lidar depth measurements. The ranging of this object depends entirely on the depth image reconstruction algorithm that can sometimes introduce wrong 3-D position.

One possible solution for modeling the changing characteristics of sensor observations is to use heavy-tailed or multi-modal distributions that, on average, explain the sensor observations under all conditions. Such distributions, on the other hand, are sub-optimal under normal circumstances. Moreover, a completely missing detection from one sensor will require a special case of the location likelihood indicating an unknown/constant contribution of that specific sensor. We can see that modeling all scenarios of sensing failures as a uni-modal function can quickly lead to a measurement model which is less informative in the general case. In this section we propose the switching observation model (SOM) concept which allows the system to automatically determine the optimal parameters of the measurement model based on the fit of the evidence to the hypothesis. The proposed method uses a hidden categorical variable which we refer to as a *local context variable.* The local context variable models the type of uncertainty of the current viewing conditions and the scene complexity at a given region in the scene. Based on the state of this context variable, a different sensor likelihood function can be applied locally, when introducing the current measurement to the system. The SOM principle is especially suitable in sampling based filters where each sample/particle can use a different measurement model.

Formally, the SOM represents the context of a local region in the environment and the characteristics of the sensor operation in this region as a latent random variable c that can switch between categorical values relating to different sensing modes of operation. Note that c is not part of the tracked object state $\mathbf{x}$, but it explains the context in which the observation is made and is related to the position in the scene where the object is located. For example, the sensor observing an object in a well lit area of the scene can be considered to be in a nominal state $c = 1$ and the detections can be explained through the nominal measurement model. A degenerate mode of operation $c = 0$ means that the sensor has failed to produce any measurement and refers to a sensor model which is independent of the state. For tracking multiple objects we will use separate contextual variables which are specific to the unique regions where

we hypothesize the presence of objects, but for simplicity in the following we will present the details for a single object tracker and a single contextual variable referring to a single location in the scene. The concept can be easily extended to multiple objects since the contextual variables do not interact with each other. In a multi-sensor system we will denote $c^{(k)}$ as the contextual state for the k^{th} sensor. The collection of K contextual variables forms the contextual vector $\mathbf{c}$. If the individual sensors use unique sensing principles (like is the case with camera, radar and lidar), then the individual $c^{(k)}$ can be thought to be conditionally independent. For each sensor, we keep a collection of $n_c^{(k)}$ contextual states that it can cycle through: $c^{(k)} \in \left\{0, ..., n_c^{(k)}\right\}$, making the general observation model for the joint evidence:

$$p\left(\mathbf{z}|\mathbf{x}, \mathbf{c}\right) = \prod_{k=1}^{K} p\left(\mathbf{z}^{(k)}|\mathbf{x}, c^{(k)}\right), \tag{6.36}$$

and the individual sensor models:

$$p\left(\mathbf{z}^{(k)}|\mathbf{x}, c^{(k)}\right) = \delta_{0,c^{(k)}} p_0\left(\mathbf{z}^{(k)}\right) + \sum_{j=1}^{n_c^{(k)}} \delta_{j,c^{(k)}} p_j\left(\mathbf{z}^{(k)}|\mathbf{x}\right), \tag{6.37}$$

where $\delta_{j,c^{(k)}}$ is the Kronecker delta.

Currently, most sensors on the market do not have the capability to estimate the quality of their own performance at an arbitrary location in the scene. Therefore, for every sensor k, we will use an observation model $p\left(\mathbf{z}^{(k)}|\mathbf{x}, c^{(k)}\right)$ where the optimization is performed without directly observing the contextual variable $c^{(k)}$, but rather using its most likely estimate given past observations of objects in a small region $\Omega\left(\mathbf{x}\right)$ around the hypothesis $\mathbf{x}$. In a general case, for every constituent sensor the following modes of operation are possible:

$$c_j^{(k)} = \begin{cases} 0 & \text{if the detection is independent of object presence at } \mathbf{x}, \\ 1 & \text{if the sensor is in its nominal state of work,} \\ j,\ j \in \left\{2, ..., n_c^{(k)} - 1\right\} & \text{if the sensor is in its } j\text{-th state of work.} \end{cases} \tag{6.38}$$

In a nominal state of work $c^{(k)} = 1$, the sensor k is assumed to be producing detections under general operating conditions, while in the other j states of work the sensor is producing various (degraded) levels of service. It is important to note that the sensor mode of operation also varies across the field of view and over time. This means that the detection quality will change in different regions of the field of view due to transient occlusions, atmospheric conditions, light changes due to shadows or multipath reflections which can cause an object detection score to briefly drop below the detection threshold. Generally, the positional accuracy of detections degrades with distance, switching between more degraded modes of operation as the range of targets

increases.

Formally, the local context variable c follows a categorical probability distribution whose sample space is the set of n_c individually identified items (sensor states). For convenience, we will define the sample space as the finite sequence of integers as presented in Eq. (6.38). The probability mass function defining the categorical distribution is given by:

$$\Pr\left(c = j\right) = \alpha_j,\ j \in \left\{0, ..., n_c^{(k)} - 1\right\}, \tag{6.39}$$

which is parameterized by the real-valued numbers α_j that sum to 1:

$$\sum_{j=0}^{n_c-1} \alpha_j = 1, \tag{6.40}$$

forming the n_c-dimensional probabilistic simplex $\boldsymbol{\alpha}$. This simplex defines the probabilities that our sensor is in any of its n_c states which, as we previously explained, are conditioned on the location of the scene. Since the local context can dramatically change throughout the scene and over time, using a single set of parameters α_j for the categorical distribution defining our context variable c is insufficient. Therefore, our method parameterizes the categorical distribution of c by treating $\boldsymbol{\alpha}$ as a random vector which is subject to estimation over time. In a K-sensor system, the categorical probability distribution of each $c^{(k)}$ is then parameterized by the sensor-specific random vector $\boldsymbol{\alpha}^{(k)} = \left(\alpha_0^{(k)}, ..., \alpha_{n_c}^{(k)}\right)$ defining the probability mass that the k^{th} sensor is in a certain sensor mode. Finally, at time t, we use the notation $c_t^{(k)}$ to indicate a sensor-specific local context variable, and $\boldsymbol{\alpha}_t^{(k)}$ the probabilistic vector with the parameters of its distribution at this time.

The visualization in Figure 6.8 shows an example perception system consisting of two sensors which can be in one of 3 different sensor states based on the location and the configuration of the scene. The person in blue is outside of the field of view of the camera and occluded by a large vehicle in the radar field, therefore both sensors are most likely in a degenerate or blind sensor state in this region $\Omega\left(\mathbf{x}\right)$ (shown as a shadow behind the truck in Figure 6.8). The person in black is visible by both sensors, however the radar signal in that region is compromised due to the presence of a large vehicle making the radar to be likely in a degraded state. Similarly, the person in red is occluded in the camera view making the camera to likely be in a degraded state for that location. Lastly, the person in green is clearly visible by both camera and radar, thus the two sensors are likely in a nominal state.

Before we give the details on how to estimate the vector $\boldsymbol{\alpha}$, we define the posterior PDF for the state $\mathbf{x}$ over the time interval $[0, t]$ given the switching observation model formulation for a system with K sensors, which now expands to:

$$p\left(\mathbf{x}_t, \mathbf{c}_t, \boldsymbol{\alpha}_t^{(k)} | \mathbf{z}_{1:t}^{(k)}, \boldsymbol{\alpha}_{t-1}^{(k)}, \boldsymbol{\sigma}_t^{(k)}\right); k = 1, ..., K. \tag{6.41}$$

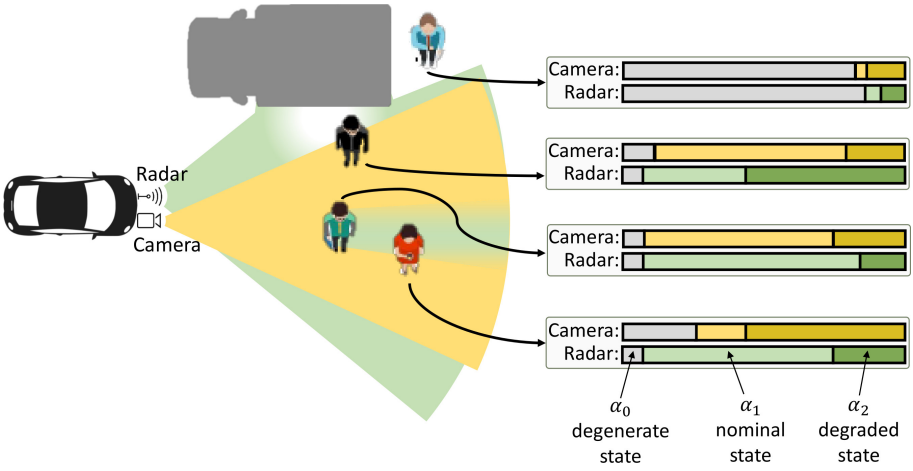


Figure 6.8: Example demonstrating the switching observation model for a scene with several VRUs. The tracker relying on measurements from two sensors: a camera (yellow) and a radar (green). Depending on the VRU position and scene complexity each sensor can be in one of 3 sensor modes: degenerate (gray), nominal (light shade) and degraded (darker shade).

For estimation of the state vector $\mathbf{x}_t$ we first need to estimate the state of the K-sensor local context vector $\mathbf{c}_t$. This, in turn, requires the estimation of the K-sensor parameter vectors $\boldsymbol{\alpha}_t^{(k)}$, for which, assuming they both have Markov property, Bayesian tracking can be employed. Caron et al. [222] have also found out that these PDFs of $c^{(k)}$ are difficult to know a priori due to the possibility of rapid changes of external conditions and propose to tune them adaptively using a Markov evolution model. Thus the transition of sensor state probabilities of the k^{th} sensor, $\alpha_j^{(k)}$, over time is $p\left(\boldsymbol{\alpha}_t^{(k)}|\boldsymbol{\alpha}_{t-1}^{(k)}, \boldsymbol{\sigma}_{t-1}^{(k)}\right)$ with $\boldsymbol{\sigma}_t^{(k)}$ being a vector of hyper-parameters that control this state transition model. We use the following state evolution model structure, as proposed by [222]:

$$\begin{aligned} \mathbf{x}_t &\sim p\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right), \\ \boldsymbol{\alpha}_t^{(k)} &\sim p\left(\boldsymbol{\alpha}_t^{(k)}|\boldsymbol{\alpha}_{t-1}^{(k)}, \mathbf{x}_t, \boldsymbol{\sigma}_t^{(k)}\right), \\ c_t^{(k)} &\sim Pr\left(c_t^{(k)}|\boldsymbol{\alpha}_t^{(k)}\right), \\ \boldsymbol{\sigma}_t^{(k)} &\sim p\left(\boldsymbol{\sigma}_t^{(k)}|\boldsymbol{\sigma}_{t-1}^{(k)}\right). \end{aligned} \tag{6.42}$$

In order to effectively explore the state space of parameters $\boldsymbol{\alpha}$ for the probability mass function of our local context variable c, we use treat $\boldsymbol{\alpha}$ as a random vector following the n_c-dimensional Dirichlet distribution. The Dirichlet distribution is parameterized using only a vector $\mathbf{a} = (a_0, ..., a_{n_c})$ of positive real-valued numbers, and the Dirichlet random variables are often used to parameterize the probabilities $\boldsymbol{\alpha}$ in categorical random variables (such as our c). That is, the vector of probabilities $\boldsymbol{\alpha}$

for a categorical random variable is itself a Dirichlet random variable. Recall that the probability density function of the n-variate Dirichlet distribution is given by:

$$\mathrm{Dir}\left(x_0, ..., x_n; a_0, ..., a_n\right) = \frac{\Gamma\left(\sum_{j=0}^{n} a_i\right)}{\Pi_{i=0}^{n}\Gamma\left(a_i\right)}\Pi_{i=0}^{n} x_i^{a_i - 1}, \tag{6.43}$$

where $\Gamma\left(.\right)$ is the Gamma function.

Our proposed method uses the parameters $\boldsymbol{\alpha}_{t-1}$ from the previous time step $t-1$ (referring to location in the scene $\mathbf{x}$) and a set of hyper-parameters $\boldsymbol{\sigma}_t$ to parameterize a Dirichlet distribution and generate new parameters $\boldsymbol{\alpha}_t$. This procedure forms the state transition model $\boldsymbol{\alpha}_t^{(k)} \sim p\left(\boldsymbol{\alpha}_t^{(k)} | \boldsymbol{\alpha}_{t-1}^{(k)}, \mathbf{x}_t, \boldsymbol{\sigma}_t^{(k)}\right)$ in Eq. (6.42). Formally, we use the Hadamard product of $\boldsymbol{\sigma}_t \circ \boldsymbol{\alpha}_{t-1}$ to parameterize the Dirichlet distribution at time t making:

$$p\left(\boldsymbol{\alpha}_t | \boldsymbol{\alpha}_{t-1}, \mathbf{x}_t, \boldsymbol{\sigma}_t\right) = \mathrm{Dir}\left(\sigma_{t,1}\alpha_{t-1,1}, ..., \sigma_{t,n_c}\alpha_{t-1,n_c}\right), \tag{6.44}$$

where the vector $\boldsymbol{\sigma}_t$ is a sensor-specific coefficient that adjusts the spread of the distribution. The intuition behind this approach lies in the interpretation of the parameter vector as a measure of how concentrated the probability of sensor state will be. For example, if $\sigma_i\alpha_i < 1$ the sample is very likely to fall in the i-th component i.e., the sensor to be in that mode of operation. If $\sigma_i\alpha_i > 1$ then the uncertainty of the sensor state will be dispersed among all components. Since $|\boldsymbol{\alpha}| = 1$, it becomes clear that the variation is mainly controlled by$|\boldsymbol{\sigma}|$. The larger the values in $\boldsymbol{\sigma}$, there will be less category variance in our Dirichlet samples, practically meaning that each new sample will explore less of the contextual state space and vice versa.

To better illustrate the properties of applying samples from a Dirichlet distribution as parameters for our categorical random variable c we show the experiment in Figure 6.9. In this experiment we perform three sampling series of 20 samples for $\boldsymbol{\alpha}_t; n_c = 5$, parameterizing its PDF using $\boldsymbol{\alpha}_{t-1} = (0.2, 0.2, 0.2, 0.2, 0.2)$ and three different sets of hyper-parameter vectors $\boldsymbol{\sigma}_t$ with equal components. The visualization clearly demonstrates the impact of the parameter $\boldsymbol{\sigma}$ in the uncertainty of the states. The experiment shown in the left plot has the highest variance of the newly sampled values meaning that the contextual variable c will likely switch to a different state regardless of the PDF of $\boldsymbol{\alpha}$ in the previous step. Contrarily, the plot on the right shows newly sampled $\boldsymbol{\alpha}_t$ that differ very slightly from the parameters in the past, meaning that the contextual variable c will likely remain in the same state.

Caron et al. [222] propose to model $p\left(\boldsymbol{\sigma}_t | \boldsymbol{\sigma}_{t-1}\right)$ using a Gaussian noise model with variances σ_t^v that are also estimated. To reduce the complexity of the estimation process, in our approach we use the following transition equation:

$$\log\left(\sigma_t\right) = \log\left(\sigma_{t-1}\right) + \lambda, \tag{6.45}$$

where λ is a zero-mean white Gaussian noise with known constant variance and the logarithm is used to ensure that the variances remain positive. As long as the individual

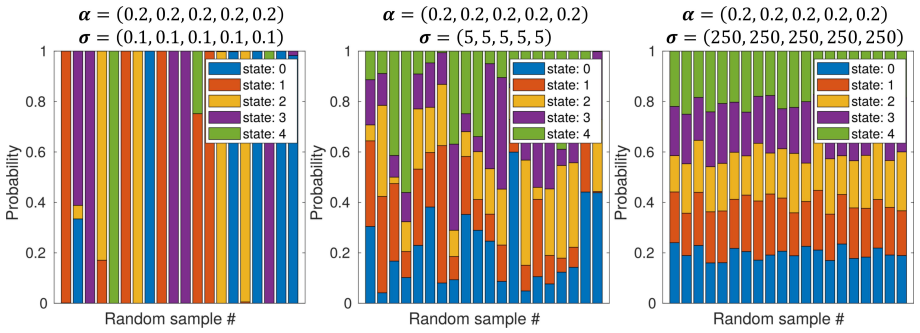


Figure 6.9: Realization of samples from a Dirichlet distribution using various parameter sets. Left: $\sigma_i < 1$ resulting in a high-variance state largely independent of α_i; middle: $\sigma_i > 1$ resulting in a state with lower variance depending on both α_i and σ_i; right: $\sigma_i \gg 1$ resulting in state with very low variance.

measurement models defined by the sensor states c_t in Eq. (6.38) can be sampled, we can apply the standard or bootstrap Particle Filter to estimate Eq. (6.41). Thus, the posterior (for a single sensor) can be rewritten as a weighted sample average of particle locations:

$$p\left(\mathbf{x}_t, \mathbf{c}_t, \boldsymbol{\alpha}_t | \mathbf{z}_{1:t}, \boldsymbol{\alpha}_{t-1}, \boldsymbol{\sigma}_t\right) \approx \sum_{m=1}^{N_{pts}} w_t^{(m)} \delta\left(\mathbf{x} - \mathbf{x}_t^{(m)}\right), \tag{6.46}$$

where the particle weights $w_t^{(m)}$ are computed using the appropriate context as:

$$w_t^{(m)} = \eta p\left(\mathbf{z}_{1:t} | \mathbf{x}_t, c_t^{(m)}\right), \tag{6.47}$$

where we use a context variable $c_t^{(m)}$ that is specific for the location of each particle $\mathbf{x}_t^{(m)}$. This posterior can be easily extended for a K-sensor setup where we will use the following weight update function instead:

$$w_t^{(m)} = \eta \prod_{k=0}^{K-1} p\left(\mathbf{z}_{1:t}^{(k)} | \mathbf{x}_t^{(m)}, c_t^{(k,m)}\right).$$

The proposed switching observation model particle filter can be interpreted as a single observation model whose PDF is a mixture of the various context-specific PDFs whose contribution distributes according to the fit of the data to the hypothesis in the past and varies locally throughout the scene. Under nominal tracking conditions the mixture context model trains itself to track optimally given the nominal sensor input and can handle transient losses of detection. When a sensor switches to a failure mode then the mixture model learns (at runtime) from the faulty (or missing) data that the current model is a poor fit and adjusts the context parameter c_t to retain optimal

tracking under the newly formed sub-optimal conditions. Its application has thus far been limited to non-linear systems in simulations and to our best knowledge has not been used for tracking of moving road users. In our experience we found that the SOMPF can be an excellent road user tracker in difficult real-world scenarios. These findings are supported by the experimental results in Section 6.9.3 and Section 6.9.4 where we compare the SOMPF to standard PF using both simulated and physical multi-sensor perception systems.

6.7 Handling Missing Detections

Bayesian filtering defines the mechanism for performing state estimation based on priors and updates given sensor evidence. Monte-Carlo implementations of the Bayes filter, such as PF, offer a sampling-based solution to compute the posterior distributions. Sampling is performed by means of a proposal function conditioned on the new observations and sample weights are computed relative to the fit of the model to the data. In the previous sub-section we saw how this framework can be extended to handle transient changes in the statistics of the observations and soft sensing failures using an adaptive mixture-measurement model which conforms to the available observations.

However, even in multi-sensor setups, it is possible that at time t a tracked object is not confidently detected by any of the sensors. As detections become missing, observation-mediated sampling in the standard Particle Filter is impossible and sample weights can no longer be updated in an informative way. This is mainly because the proposal functions are conditioned on the new detection. When using the bootstrap Particle Filter, sampling of new particles can still be performed, but the update of particle weights is impossible without an observation. In situations where a detection is completely missing, Bayesian filters cannot make a state estimate. This is problematic for the tasks in autonomous driving because disappearing object estimates can drastically affect the actions taken by collision avoidance and path planning.

To combat the problem of missing detections, trackers in the literature often part from the Bayesian theory, skipping the update step and still providing an object estimate which in reality is only a prediction based on the state transition model. Handling missing detections without leaving the Bayesian framework can be achieved by imputation i.e., replacing missing data with substituted values. In the following, we propose two solutions for multi-sensor, multi-object tracking that deal with missed detections. The first is a particle filter that reconstructs a missed detection using statistical modeling and multiple imputations, while the second is a particle filter with a two-step update cycle that only switches to tracking-before-detection in ambiguous circumstances. In the field of missing data reconstruction, the first approach is regarded as the state-of-the-art methodology, but as we shall show in the experimental assessment, it yields poor results since it offers little gains in accuracy at a substantial increase in computational complexity. However, compared to the first one, the second method, which only uses tracking-before-detection when necessary, is able to attain

greater accuracy at a lower computational cost.

Multiple Imputations Particle Filter

Originally introduced in the book [223] and later used in the papers [212] and [214], the Multiple-Imputations Particle Filter (MIPF) extends the PF algorithm for cases of missing detections. The main statistical assumption in this approach is that the sensor detections are missing-at-random (MAR). This means that the predisposition for a detection to be missing can be related to the observed ones. It is a solid assumption in tracking-by-detection systems where the detectors use a high precision working point and some true positive detections are missing due to a low detection score. A low detection score is usually the result of poor viewing conditions or occlusion. The missing detection is indicated whenever there is a presence of an occluder, so, good techniques for imputing MAR data need to incorporate variables that are related to the "missingness". In the following we will lay out the standard MIPF formulation as a baseline method for handling missing observations.

The general principle of operation of MIPF is the following: first the filter's particles are propagated using the state transition model, then the missing detection is substituted by multiple, random samples (imputations) sampling from the the sensor model centered at the hypothesized positions of each particle. Finally, the particle weights are updated by taking a weighted sum of the likelihood of all imputed samples. In the paper [215] authors provide more details about the MIPF and prove the *almost sure* convergence of this filter. Formally, the MIPF [212, 215] uses a partitioned vector $\mathbf{v} = (\mathbf{y}, \mathbf{z})$ to model a detection which consists of a missing part and an observed part. The missing part is modeled using the auxiliary variable $\mathbf{y}$. This formulation allows the particle filter to apply updates as defined in Eq. (6.29) and Eq. (6.31) even when a detection $\mathbf{z}_t$ is missing. In the following we will lay out the method for sample the imputations and update the particle weights.

To make the following analysis compatible with the SOM model proposed in the previous chapter, a detection $\mathbf{v}_t$ can be explained by the switching observation model:

$$\mathbf{v}_t = \begin{cases} \mathbf{y}_t & \text{if } c_t = 0, \text{ (missing obseration)} \\ h_1\left(\mathbf{x}_t, \mathbf{g}\right) + \boldsymbol{w}_{1,t}; & \text{if } c_t = 1, \text{ (nominal state)} \\ h_j\left(\mathbf{x}_t, \mathbf{g}\right) + \boldsymbol{w}_{j,t}; & \text{if } c_t = j, \text{ (j-th degraded stete)} \end{cases} \tag{6.48}$$

where $h_j\left(.\right)$ are nonlinear observation functions and $\mathbf{w}_{j,t}$ is an observation noise for the j-th sensor state at time t, and c_t is the local context variable estimated at time t using the transition models in Eq. (6.42). The posterior PDF for the location of a tracked object $\mathbf{x}_t$, Eq. (6.41) can be written as:

$$p\left(\mathbf{x}_t \left|\mathbf{z}_t, c_t\right.\right) = \int p\left(\mathbf{x}_t | \mathbf{z}_t, \mathbf{y}_t, c_t\right) p\left(\mathbf{y}_t \left|\mathbf{z}_t, c_t\right.\right) d\mathbf{y}_t, \tag{6.49}$$

integrating over all possible $\mathbf{y}_t$, where for clarity, we will omit the parameterization variables $\boldsymbol{\alpha}_t$ and $\boldsymbol{\sigma}_t$ assuming they have been estimated by applying Eq. (6.42)

through Eq. (6.45). Consequently, let's assume that the method by which a detection was missed depends only on the detected ones [223], we will rewrite the posterior as:

$$p\left(\mathbf{x}_t \left| \mathbf{z}_t, c_t \right.\right) = \int p\left(\mathbf{x}_t \left| \mathbf{z}_t, \mathbf{y}_t, c_t \right.\right) p\left(\mathbf{y}_t \left| \mathbf{z}_t \right.\right) d\mathbf{y}_t, \tag{6.50}$$

which means that the statistical model of the missing information is not necessary. In this special case, the posterior distribution, can be computed using n_{imp} number of imputed particles:

$$p\left(\mathbf{x}_t \left| \mathbf{z}_t, c_t \right.\right) = \lim_{n_{imp}\to\infty} \frac{1}{n_{imp}} \sum_{i=1}^{n_{imp}} P\left(\mathbf{x}_t \left| \mathbf{z}_t, \mathbf{y}_t^{(i)}, c_t \right.\right), \tag{6.51}$$

where the imputations $\mathbf{y}_t^{(i)} \sim p\left(\mathbf{y}_t \left| \mathbf{z}_t \right.\right)$ are not conditioned on past detections and the state transition model. We adopt the proposed solution devised in [215] to resolve this deficiency by drawing imputations from the missing data probability density which is unknown, but can be approximated from the posterior assuming no detections went missing prior to t:

$$p\left(\mathbf{y}_t \left| \mathbf{z}_{0:t} \right.\right) = \int p\left(\mathbf{y}_t, c_t \left| \mathbf{x}_t \right.\right) p\left(\mathbf{x}_t \left| \mathbf{z}_{0:t}, c_t \right.\right) d\mathbf{x}_t, \tag{6.52}$$

This assumption is true for brief periods of time when there are few missed and much more seen detections. In order to get a good estimate of the posterior it is required that detections were present in the time instances leading up to the missing detection. When a detection is missing, we cannot update the posterior in the standard way which means that we can not sample directly from it, so we use an approximation by applying the state transition model. This means that the particles $\mathbf{x}_{t-1}^{(m)}$ are propagated using the state transition model to obtain an estimated PDF, formed by $\widetilde{\mathbf{x}}_t^{(m)}$.

In practice, a missing detection will almost certainly be caused by a localized change of sensor mode c_t due to the loss of signal strength, occlusion, ambiguous association or noise. An imputed detection can therefore be simulated using the last known sensor model and the expected position from the motion model. At the moment when a detection went missing, we can let the sensor evolution model Eq. (6.42) choose the most likely course of evolution of c_t. It is safe to assume that the missing detection PDF is the same as that of the observed data, $p\left(\mathbf{y}_t, c_t^{(m)} \left| \widetilde{\mathbf{x}}_t^{(m)} \right.\right) = p\left(\mathbf{z}_t, c_t^{(m)} \left| \widetilde{\mathbf{x}}_t^{(m)} \right.\right)$, so we can use the imputation proposal function $\phi\left(\mathbf{y}_t \left|, \mathbf{z}_{0:t} \right.\right)$:

$$p\left(\boldsymbol{y}_t \left| \boldsymbol{z}_{0:t} \right.\right) \approx \phi_{Y|Z}\left(\mathbf{y}_t \left|, \boldsymbol{z}_{0:t} \right.\right) = \sum_{m=1}^{n_{pts}} w_t^{(m)} p\left(\mathbf{y}_t, c_t \left| \widetilde{\mathbf{x}}_t^{(m)} \right.\right), \tag{6.53}$$

from which we draw imputations $\mathbf{y}_t^{(j)} \sim \phi\left(\mathbf{y}_t \left|, \boldsymbol{z}_{0:t} \right.\right)$. Practically, this equation stipulates that the set of simulated detections (imputations) will be generated using the

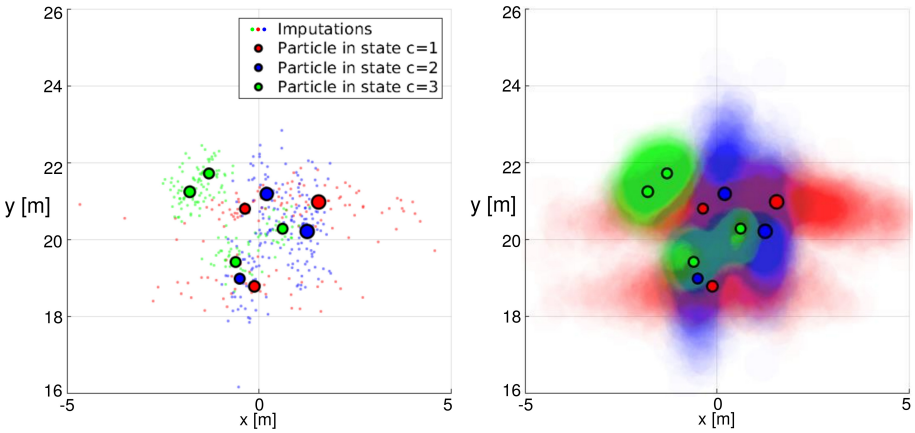


Figure 6.10: Visualization of the set of 10 particles and 500 imputations drawn by a standard MIPF for a single VRU $\mathbf{x}_t = (0, 20)$. The left plot shows one realization of sampling imputations from particles whose colors indicate their respective sensor state variable $c_t^{(m)}$. The right plot illustrates the underlying imputation proposal function $\phi\left(\mathbf{y}_t \,|\mathbf{z}_t\right)$.

most likely observation models applied at the location of each predicted particle. This procedure is illustrated on the two plots on Figure 6.10.

According to [224] we can use these complete data sets, $\mathbf{v}_t = \left(\mathbf{y}_t^{(j)}, \boldsymbol{z}_t\right); j = 0, ..., n_{imp}$, to compute an approximation of the posterior PDF. Substituting $\mathbf{v}_t$ in Eq. (6.50) yields:

$$p\left(\mathbf{x}_t|\mathbf{z}_t, c_t\right) = \int \left(\mathbf{x}_t \,|\mathbf{v}_t, \mathbf{z}_t, c_t\right) p\left(\mathbf{y}_t \,|\mathbf{z}_t\right) d\mathbf{y}_t, \tag{6.54}$$

where the approximate PDF is computed as the Monte Carlo simulation using n_{imp} imputations. The posterior computed using a particle filter then becomes:

$$p\left(\mathbf{x}_t, |\mathbf{z}_{0:t}, c_{0:t}\right) = \lim_{n_{imp}\to\infty} \frac{1}{n_{imp}} \sum_{j=1}^{n_{imp}} \sum_{m=1}^{n_{pts}} w_t^{(j,m)} \delta\left(\mathbf{x}_t - \widetilde{\mathbf{x}}_t^{(j,m)}\right), \tag{6.55}$$

which is computed by performing particle filtering treating each imputation $\mathbf{v}_t^{(j)}$ as a detection, where $\widetilde{\mathbf{x}}_t^{(j,m)}$is the m-th particle for the j-th imputation at time t and $w_t^{(j,m)}$ is the respective weight estimated from the most likely observation model from $c_t^{(m)}$.

Two problems arise when applying the multiple imputations PF for real-time application. Firstly, its computation is prohibitively expensive because each time a detection is missing, the particle filter needs to perform $n_{imp} \gg 1$ updates treating each imputation $\mathbf{v}_t^{(j)}$ as a simulated detection and then average the results (double sum in Eq. (6.55)). The complexity lies mainly in sampling the measurement model when

computation of the weights which requires $n_{pts} \times n_{imp}$ evaluations. Second, since we are dealing with a switching observation model, the accuracy of imputed particles relies on the accuracy of the estimates $\left(\widetilde{\mathbf{x}}_t^{(m)}, c_t^{(m)}\right)$ which are in turn driven by available detections from the past. In cases when detections are missing in short bursts, updating the model with imputations yields an accurate estimate of the posterior PDF. However, when detections are missing over an extended time interval e.g., more than a few update cycles, the context variable state transition model can quickly lead to an uninformative vector $\boldsymbol{\alpha}_t^{(m)}$, meaning that the states of all context particles $c_t^{(m)}$ become uniformly distributed. This results in diminished informativeness of the imputations and tracking becomes no better than using motion prediction alone, which we experimentally show to be true using realistic tracking simulations in Section 6.9.3. To overcome these issues, in the next section we propose a two-stage tracking mechanism based on the same variable threshold principle proposed in Section 5.3. Specifically, the particle filter can update its state in locations where a detection is expected but not seen, by using sub-threshold observations without association. We will experimentally show that even without association, uncertain sensor data helps the tracker to overcome the problem of missing detections better than using imputations.

Proposed method

Detections missing at random can mostly be attributed to two main factors. Firstly, due to poor viewing conditions, the detection confidence of individual objects can sometimes dip bellow the detection threshold set by the working point of the detector. In such cases, the fact that a detection is missing (referred to as missingness) is generally not dependent on the scene geometry or the presence of other objects in the scene. In this case, the imputation filter can offer little advantage over tracking based on motion prediction. Second, in situations where the scene contains multiple objects, a missing detection can be caused by ambiguous matching of evidence to hypotheses. This means that a detection that is associated to the wrong hypothesis may cause a detection to be missed by the correct hypothesis. In the context of multi-object tracking, a missing detection caused by ambiguous association can be considered to be weakly conditioned on the nearby observations. The imputation filter can here be effectively applied, but at a great computational cost. In this section we propose a novel, alternative method for tracking with missing detections by using a two-step update cycle with a locally varying detection threshold.

The proposed method updates the tracking hypotheses in two steps by controlling the operating points of the object detectors locally: first, hypotheses (both their existence and location) are updated in the standard way i.e., using confident detections obtained at a high precision point, and second, the location of the remaining hypotheses are updated with detection evidence obtained at a high recall point. Running the detectors at a high recall point generates lots of false positives and clutter. It is important to note that the proposed tracker employs this data in particular areas of the scene where there is suspicion of object presence based on previous observations. The

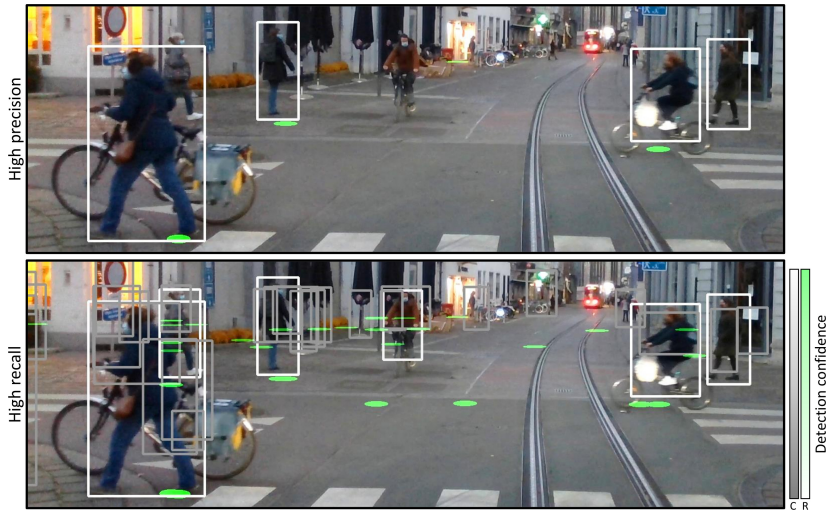


Figure 6.11: Visualization of objects detected by camera (bounding boxes) and radar (ellipses) at different operating points. Top: objects detected at high precision, bottom: objects detected at high recall.

idea behind this method is that rather of relying just on motion estimates, the position of hypotheses for which we do not have detections may be better inferred from the low confidence evidence (even if it may be a false positive). However, with time, the belief of existence of such hypotheses will decrease. Before we delve into the details of the method, let's give an example of how the observational evidence look like at the two separate working points of an image and radar object detector. On the top image in Figure 6.11, we illustrate detection at high precision, consistent with the tracking-by-detection principle, while on the bottom image the same detectors operate at maximum recall. It is apparent that in the first case some difficult-to-classify objects are not detected while in the second case all objects are detected, although at a large cost of false positives.

In order to overcome the issues of missing detections when tracking at high-precision (track by detection) and the high computational load of tracking at high-recall (track before detection) we will devise the following strategy which combines the useful properties of the two principles and avoids their downsides. Strong (high precision) *detections* are used to update confident hypotheses in the standard track-by-detection way, while weak (high recall) *evidence* updates uncertain hypotheses in a track-before-detection way. Therefore, the proposed algorithm achieves both high tracking accuracy while retaining low algorithmic complexity.

The **first step** is to select only the confident detections and match them to the tracking hypotheses to which they correspond the most. To that end we make use of a heterogeneous distance measure $d(.)$, that mixes distance, shape and appearance. Depending on the specific sensor in question, the distance measure operates either on

the ground plane or on the image plane. For measuring the similarity in the position and shape we use the Euclidean distance, while for measuring the appearance similarity we use the Kullback-Leibler divergence of the color histograms. The set of N_{det} confident detections and N_{obj} hypotheses gives $N_{det} \times N_{obj}$ combinations of possible matches. The optimal association of detections and hypotheses is achieved by optimizing an association cost matrix $\left[\mathrm{d}\left(\mathbf{x}_{i,t}, \mathbf{z}_{j,t}\right)\right]_{N_{det} \times N_{obj}}$ using the Hungarian algorithm [28]. The result of the matching is a set of association pairs: tuples $\left(\mathbf{x}_{i,t}, \mathbf{z}_{j,t}\right)$ where the indices i and j refer to a detection and a hypothesis with high similarity. Each confident detection $\mathbf{z}_{j,t}$ will update the belief in existence and location of each hypothesis with location $\mathbf{x}_{i,t}$.

It is not unlikely that some confident detections cannot match to a hypothesis, and also, some of the hypotheses might not match with the confident detections. These leftover confident detections will spawn new hypotheses which we add to the pool of tracked objects, while the unmatched hypotheses are processed in a **second step** where we use sub-threshold evidence and matching based purely on position. Additionally, because these hypotheses were not matched to confident detections in the first step, the belief in their existence $\left({}^{p(H_1|\mathbf{z})}/_{p(H_0|\mathbf{z})}\right)$ will decrease. The motivation to use a weaker matching function in the second step is due to the abundance of sub-threshold evidence which makes the number of detection-hypothesis combinations very large. Computing appearance similarity in a high-dimensional feature space is much more computationally intensive than computing the Euclidean distance in $\mathbb{R}^3$. Formally, for each unassociated hypothesis from the first step (indicated by index i), we compute the likelihood of the position $\mathbf{x}_{i,t}$ (in polar coordinates) using the sub-threshold evidence based purely on their positions in the respective sensor coordinate system:

$$p\left(\mathbf{u}_{j,t}^{(0)}, ..., \mathbf{u}_{j,t}^{(K-1)} | \mathbf{x}_{i,t}\right) = \prod_{k=0}^{K-1} p_{U_t^{(k)}|X_t}\left(\mathbf{u}_{j,t}^{(k)} | \mathbf{x}_{i,t}\right), \tag{6.56}$$

where $\mathbf{u}_{j,t}^{(k)}$ is the location of the sub-threshold detection from sensor k that is closest to *the location* $\mathbf{x}_{i,t}$. Since we are using particle filters to track objects, Eq. (6.56) needs to be evaluated many times in order to update all particle weights for all unassociated hypotheses. Due to the large number of sub-threshold detections, this procedure can become intractable even for small number of hypotheses. We propose an approximation of Eq. (6.56) where at each time step we pre-compute the likelihood values over a grid of positions $\mathbf{x}$ with a finite size and resolution. This grid of likelihoods can be used to quickly update particle weights $w_{i,t}^{(m)}$ by approximating the likelihood at particle position $\mathbf{x}_{i,t}^{(m)}$ with the likelihood value computed at the nearest grid cell. In Figure 6.12 we show an example of the weak camera and radar evidence aggregated in a 2-D grid for the same scene shown in Figure 6.11. The assumption in our approach is that the number of grid cells is far lower than the number of sampling operations needed to update the weights of N_{pts} particles of the N'_{obj} unassociated hypotheses using N'_{det} sub-threshold detections. This method becomes especially effective as the number of tracked objects in the scene increases.

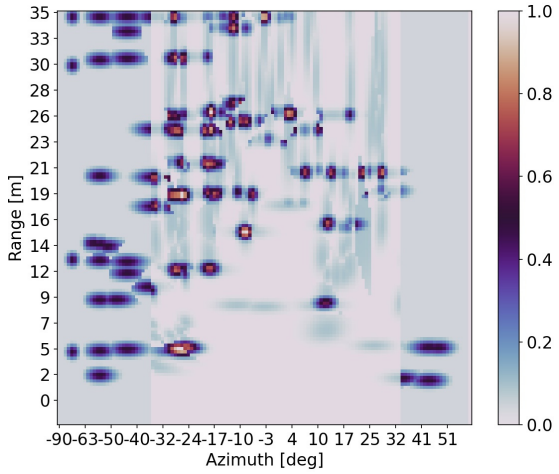


Figure 6.12: Visualization of the proposed measurement model computed over a 2-D grid of size 128×128 for the scene in Figure 6.11.

It is important to note that our weak evidence grid contains detection information which falls below the high precision detection threshold. As such, this evidence is used to update only the belief in the location of each hypothesis, while the belief in the existence and the sensor state contextual variables $c^{(k)}$ are only predicted by transition probabilities. We found that updating the locations of particles with this weak evidence provides additional information over using motion models alone or the multiple imputations filter in 6.7. Still, the confidence for the existence of such tracks decreases, as explained in Section 6.3.1.

Using the proposed second step to update particle positions of unassociated objects causes the particle position to conform to any weak observational evidence which improves tracking accuracy compared to using motion models or imputations. The weight of the posterior in this missing detection case is independent of appearance association as it depends only on the closest weak detection. Compared to MIPF the proposed approach has a reduced computational load for updating the particle weights at the increased cost of pre-computing the likelihood grid. For most of our experiments in the following section we used a grid of size 128×128 which spans the range of $\pm 90^{\circ}$ in the azimuth and $[0m, 40m]$ in polar coordinates which resulted in measurable improvements of tracking performance in cases of missing detections over methods such as Kalman Filter, Particle Filter and Multiple Imputations Particle filter. The improvements are most significant in complex traffic scenes where failures in the object detection cause most problems for standard object trackers.

The downside to this approach is that we allow for the rasterized likelihood information to update any particle of any track which happens to be closest to the particular grid cell. In rare cases, the positions of particles of multiple nearby tracks can be updated using the same information which can inadvertently lead to hypotheses spatially

converging to each other and merging. However, one can argue that in such situations the limited evidence does not support the existence of more than one hypothesis and they should be merged anyway.

6.8 Track management

There are extra difficulties when tracking a potentially high number of objects from a moving vehicle with several sensors since the perception system must keep track of the number, location, and identity of objects over time. An object of interest may move into the sensors' field of view, leave it, or stay there. Track management refers to the process of organizing the existence, identity and the number of tracked objects. In a well performing multi-object tracker the belief in existence of true hypotheses should be maximized while minimizing the belief in existence of clutter. Moreover, the number of tracking hypotheses should reflect the number of true objects in the scene. Since this thesis does not bring significant contributions to track management, this section only briefly explains the employed management algorithm.

To illustrate the necessity of track management, consider the following example. A perception system based on a forward-looking sensor array performs object detection and tracking. New and unseen objects will most likely appear near the borders of a camera frame or near occlusion zones. When such an event does happen, it creates evidence which does not explain any existing hypotheses. The track manager needs to treat this as a creation event where a new object enters the scene, give it a unique identity and increase the number of hypotheses. Similarly, when the vehicle drives past objects that exit the sensor field of view, it is likely that such objects will not be re-acquired in the near future. In this case, a hypothesis is not expected to be supported by evidence for a longer period of time and the track manager needs to remove it from the state space. Finally, in situations of object occlusion such as groups of people, we can observe ambiguous evidence which supports some, but not all of the hypotheses. When this type of evidence persists and some of the hypotheses become similar, the tracker needs to decide which one to keep and which one to remove. Lastly, the opposite event where the tracker starts to persistently get multiple detections around a single hypothesis is considered as a track splitting scenario.

In the proposed track management method object hypotheses can switch between several logical categories of existence, referred to as *states* in this section. These existence states are in direct correlation with the belief in existence explained that we explained in Section 6.3.1. Switching between logical tracking states is defined by taking *actions* that are unique to the specific state. Our track manager is a finite state machine computing the states of hypotheses by taking actions defined by the state diagram shown in Figure 6.13. Before initialization, we have a pool *inactive* hypotheses which consists only of a unique track identifier (ID) but have no spatial or appearance attributes. An inactive track becomes *active* by spawning from novel observational evidence and is defined by its position, features and track score. The track manager uses the maximum a posteriori (MAP) value of the random variable: $\mathbf{x}_t^{(MAP)}$ as its

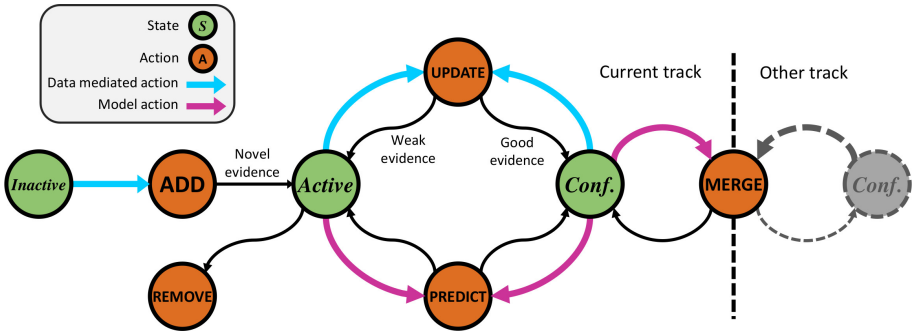


Figure 6.13: State diagram of the proposed track management finite state machine.

track location, while the track score is the log-odds ratio of the track existence defined in Section 6.3.1. Active hypotheses that are well supported by observational evidence over time become *confirmed* tracks. The confirmation happens when the track score, Eq. (2.11), reaches a certain threshold which defines the tracker operating point. Contrarily, active hypotheses that over time do not adhere to any observational evidence get removed from the system.

The set of actions consists of *adding* a new hypothesis to the solution (add/spawn), *predicting* the state of an existing hypothesis, *updating* an existing hypothesis, *merger* of two hypotheses and, finally, *removal*. Add and update actions are mediated by observational evidence or data while predict, merge and update are conducted by transition models. The outcome of each action is a transition to a state as a function of the track score. For example, the track manager changes the logical state of an inactive hypothesis to active by executing the add action given confident novel detection. We consider a detection to be novel if it does not associate with existing hypotheses with respect to the association metric. The decision to downgrade a confirmed hypothesis to "active" is based on a set threshold on the log-odds existence ratio. During merger between two tracks, the one with a lower track score gets removed from the system and switches to a lost state, not shown on the state diagram in Figure 6.13.

Track management systems such as the one we describe here can be thought of a Markov Decision Process (MDP) where the transitions between states are partly random and partly under the control of a decision maker. A MDP randomly transitions through the states defined by the finite state diagram of the system driven by appropriate reward functions. In our case the reward computes how well the newly computed track score matches the state the track is in. To illustrate this, imagine an "active" hypothesis that gets updated, and its track score reaches the track confirmation threshold. The reward for moving this hypothesis from "active" to "confirmed" is greater than the reward for staying in the "active" state. In other words, based on the evidence we are more confident that this hypothesis is a real object than not. A MDP (stochastic) track manager should, in this case, randomly switch the state of the hypothesis from "active" to "confirmed" with a higher probability than keeping in the state unchanged.

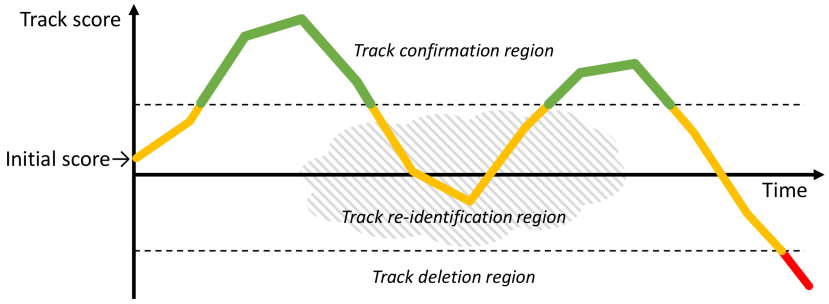


Figure 6.14: Example evolution of the track score for one tracked object over its life cycle. This track gets confirmed, then briefly lost and re-identified, and, finally, removed due to lacking support by the observational evidence.

MDP solvers discover such policies through iterative traversing the graph of possible states and actions and computing the total reward. However, this means that the solver needs access to track states in the future and is therefore not applicable for on-line tracking. Thus, we did not investigate more complex track management methods and reduced the MDP to a deterministic, sequential decision driven process by fixed rules based on the current track scores which we optimize empirically.

The diagram in Figure 6.14 illustrates an example of the evolution of a track score for one hypothesis over its life cycle. This example track moves through the following chain of states: inactive→active→confirmed→active→confirmed→active→lost. After spawning, the track gets associated with good observational evidence and becomes a confirmed track when its track score reaches a confirmation threshold. This threshold defines the tracker operating point. After a while, the same track is lost from sensor view and its track score decreases, then it's reacquired and its track score reaches the confirmation threshold again. Finally, the track gets out of view without the chance of re-acquiring and its track score drops below the track deletion threshold when the track manager ultimately removes this hypothesis from the system.

6.9 Experimental evaluation and results

This section presents the methodology, datasets and results from experiments conducted for assessing the performance of both the individual components and the complete tracking system. We will compare the proposed tracker to state-of-the-art tracking methods in the literature using various combinations of camera, lidar and radar sensors and evaluate over multiple datasets using various tracking performance criteria. We will also employ accurate simulations to identify and quantify the distinct advantages of the various contributions over baseline trackers. Real-world tracking experiments focused on camera and lidar fusion are performed using the highly cited KITTI [32] and nuScenes [168] datasets. The IPI research group at Ghent University has been also internally capturing and labeling unique datasets for use in tracking studies that combine camera, lidar, and radar. We will apply our tracker on two such internal datasets on which we will compare against trackers from the literature for which the full implementation was a publicly available at the time of experimentation. It is important to note that this part also offers a fine-grained evaluation of the proposed method's performance against controls on each sub-set individually under varied traffic and weather situations.

6.9.1 Datasets

As of the time of writing this thesis the availability of quality datasets tailored for autonomous driving which include a combination of cameras, lidars and radars is limited. This scarcity is a consequence of to the novelty of automotive radar technology which many companies consider a closely guarded secret. One of the few public datasets containing, among other sensor modalities, automotive radar data is the recently introduced nuScenes dataset [168]. However, this dataset contains radar data with sparsely populated 2-D radar targets. The very recent public dataset (OLIMP [225]) tailored for VRU detection and tracking incorporates measurements from both camera and ultra wide-band radar sensors. The captured radar data, however, is incomplete as it lacks dense azimuth information which makes it incompatible to our method. The Astyx HIRES2019 dataset [226] contains semi-dense radar data in the form of up to 1000 processed radar targets per frame. Even though this dataset might seem like a solid benchmark for our system, upon closer inspection we found that the majority of the objects are of vehicle targets while people and other vulnerable road users are underrepresented. The information provided in these datasets is always at the target level with raw radar readouts being lost. Our radar-based cooperative fusion detector requires a complete range-Doppler-azimuth radar cube to perform accurate, high recall, radar detection making available datasets inadequate. Lastly, the dataset most applicable to our work was captured by the Intelligent Vehicles at TU Delft [136] and offers synchronized camera and raw radar 3-D cubes. Unfortunately, due to a non-disclosure agreement issues the complete data is not available at the time of writing and simulations are provided as placeholders. We were therefore motivated to either experiment on public datasets lacking radar, and capture, annotate and run

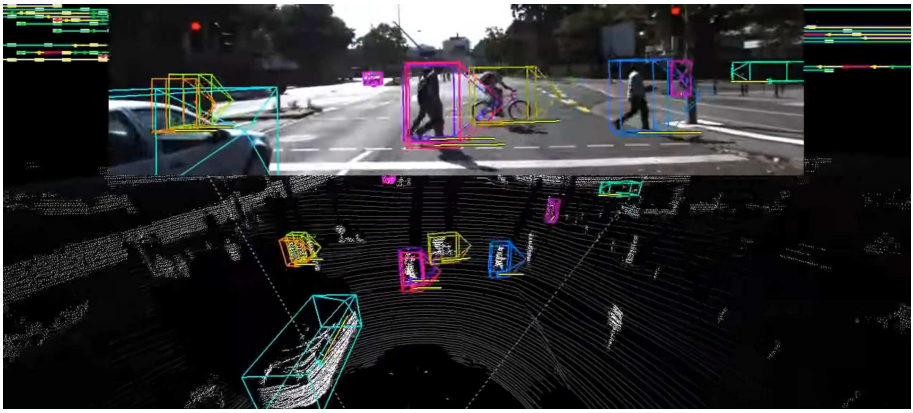


Figure 6.15: Example visualizing the camera, lidar and annotations available in the KITTI dataset. All objects are labeled in 3-D (bottom view) and their projections onto the camera frame (top view) can be computed using the provided calibration data.

experiments on our own dataset which includes radar measurements.

KITTI tracking dataset

The KITTI [20,32] public dataset has been recorded from a moving platform while driving in and around Karlsruhe, Germany. It includes camera images, laser scans, high-precision GPS measurements and IMU accelerations from a combined GPS/IMU system. The main purpose of this dataset is to enable the training and evaluation of perception systems and push forward the development of computer vision and robotic algorithms targeted to autonomous driving. The benchmarks are captured by driving around a mid-size city, in rural areas and on highways. The KITTI recording platform is equipped with two high resolution stereo camera systems (grayscale and color), a Velodyne HDL-64E laser scanner that produces more than one million 3-D points per second and a state-of-the-art OXTS RT 3003 localization system which combines GPS, GLONASS, an IMU and RTK correction signals. The cameras, laser scanner and localization system are calibrated and synchronized, providing us with accurate ground truth.

Most relevant to the task of multi-object tracking is the set of 40 sequences which are labeled with dynamic objects: their position, category, ID and size. For each dynamic object within the reference camera's field of view, this benchmark provides annotations in the form of 3-D bounding box tracklets, represented in local coordinates. Objects are categorized into the following classes 'Car', 'Van', 'Truck', 'Pedestrian', 'Person (sitting)', 'Cyclist', 'Tram' and 'Misc' (e.g., Trailers, Segways). The KITTI dataset also provides a tracking benchmarking platform which shares its methodologies with MOT16 [227] where performance is evaluated primarily for tracking using camera data, with lidar data synchronized to the camera image. Therefore, a lidar

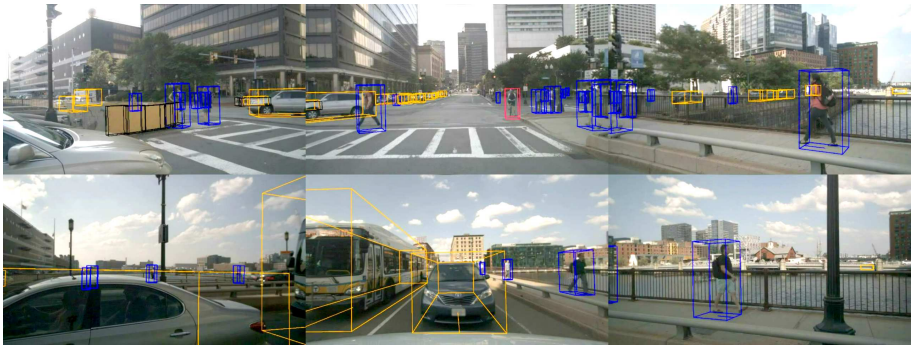


Figure 6.16: Example visualizing the 3 forward looking cameras (top) and the three rear looking cameras (bottom) of the nuScenes dataset for scene-0716. Labeled ground truth is visualized as colored cuboids: blue-person, yellow-vehicle, black-infrastructure, red-cyclist.

object is only annotated if and only if the object is within camera frame. The dataset is split into two disjoint sets of video sequences, 29 of which are used for independent evaluation at the server side and 21 sequences have ground truth available for training. A total of 172 unique pedestrians are labeled with 11470 instances across the 8008 training frames, see Figure 6.15 for an example 2-D / 3-D data frame.

nuScenes tracking dataset

The NuScenes dataset [168] addresses the limitations of current autonomous vehicle (AV) benchmarking datasets by providing a large-scale multi-modal dataset with coverage across all vision and range sensors collected from diverse situations alongside map information. It represents a large leap forward in terms of data volumes and complexities, and is the first dataset to provide 360-degree sensor coverage from the entire sensor suite. It is also the first AV dataset to include radar data and captured using an AV approved for public roads. It is further the first multi-modal dataset that contains data from nighttime and rainy conditions, and with object attributes and scene descriptions in addition to object class and location. It enables research on multiple tasks such as object detection, tracking and behavior modeling in a range of conditions.

NuScenes was captured in Boston (Seaport and South Boston) and Singapore (One North, Holland Village and Queenstown), two cities that are known for their dense traffic and highly challenging driving situations. It emphasizes the diversity across locations in terms of vegetation, buildings, vehicles, road markings and right versus left-hand traffic. The capture vehicle was equipped with a sensor array comprising of 6 RGB cameras, a 3-D lidar, 5 FMCW radars and a GPS and IMU sensors. The dataset comprises of 1000 short sequences (20s each) selected from 84 logs with 15h of driving data (242km traveled).

Most relevant to this thesis are the nuScenes detection and tracking benchmarks

which contain ground truth labels for objects similarly to the KITTI dataset. Labels are provided for every "key-frame" (image, lidar, radar) at 2Hz. Each object is annotated into one of the 23 object classes with a semantic category, attributes (visibility, activity, and pose) and a cuboid modeled as x, y, z, width, length, height and yaw angle, see Figure 6.16 for an example of the data provided. There are an average of 7 pedestrians and 20 vehicles per key frame with a total of 40K keyframes organized in 700 training sequences, 150 validation sequences and 150 test sequences. Around 10% of the sequences were captured at night or in bad weather conditions. Generally speaking, the more recent nuSenes dataset is an order of magnitude larger than KITTI and provides a more realistic benchmark for environmental perception systems in autonomous vehicles.

Unfortunately, the provided radar data in the nuScenes dataset is already processed by the internal object detector of the employed radar sensor and comprises only of target positions and velocity vectors. The lack of raw range-azimuth-Doppler radar data makes it impossible for us to deploy our radar detection CNN explained in Section 5.5. Therefore, we were unable to fully utilize the provided radar data in nuScenes and instead focused on experimenting with the camera-lidar measurements.

IMEC tracking dataset v1 (Gent Zuid)

In order to close the gap in availability of raw radar data in AV benchmarks, we used the equipment and expertise within the IPI research group and recorded and annotated a dataset captured in real-world traffic over two campaigns. The initial recording campaign was done in the city center of Ghent, around the Zuid region while the second recording campaign covered multiple locations in Ghent, Bruges and Maldegem; cities in Belgium. For the first capture campaign we used a prototype electric tricycle equipped with a sensor array and recording hardware. The sensor array consists of a forward-looking, wide field of view RGB camera, (GoPro Hero4 black), providing high quality, time-stamped video information, a 77GHz FMCW radar (Texas Instruments AWR1243) and a 3D lidar (Velodyne VLP-16).

The camera was set to record frames in a FullHD resolution at $30FPS$. Its horizontal field of view after accounting for the fish-eye image deformation is approximately $90°$. The Velodyne lidar contains 16 laser diodes that measure distance in a 360-degree circle up to $100m$. The beams are uniformly arranged over the $\pm 15°$ elevation, while the azimuth resolution is $\sim 0.2°$. The accuracy of the measured range of each beam is$\pm 3cm$. The AWR1243 radar, on the other hand, contains only 3 transmit and 4 receive antennas capturing raw data organized in 3-D (range-azimuth-Doppler cubes). We used a configuration of the AWR1243 radar that allows for a range resolution of $36.5cm$ with maximal range of $46.72m$, and Doppler resolution of $0.21ms^{-1}$ with a maximal unambiguous velocity of $50kmh^{-1}$. Due to the low number of antennas which are arranged in a horizontal pattern, the radar can sense only in one elevation plane with an angular resolution of $11.25°$. Vehicle odometry is computed at run-time using the lidar data and the algorithm from [9] allowing perception in global coordinates.

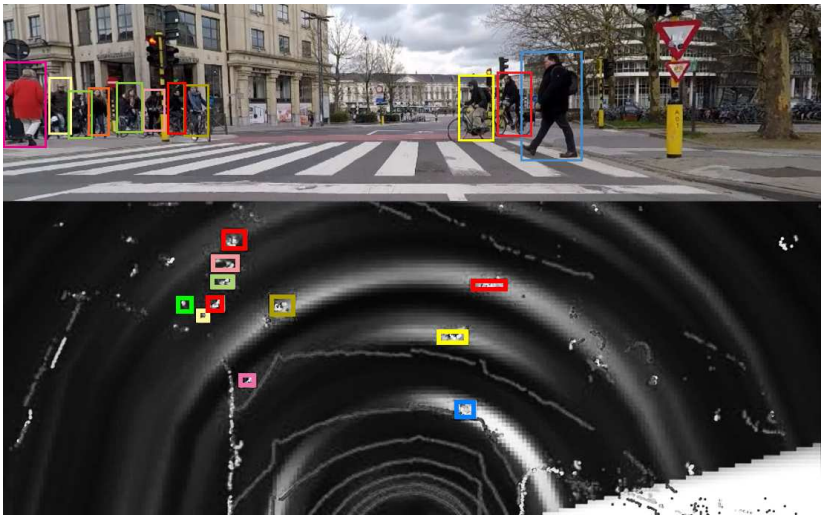


Figure 6.17: Example visualizing the camera (top) and radar, lidar and annotations (bottom) available in the IMEC v1 dataset. Colored cuboids in the image and top-view represent the ground truth positions and shapes of individual objects.

We selected 16 short and highly representative sequences in which we manually labeled the positions of all VRUs. We used the camera and lidar data to guide the annotation of ground truth VRUs positions. Objects are labeled as 2.5-D (ground plane) bounding boxes defined by their position, size and identity, see Figure 6.17. Four annotators were trained and asked to match people visible in the RGB image to objects in the lidar point cloud. The matched objects were ranged, forming labels containing the bounding boxes of each person relative to the sensor array. This manually labeled dataset consists of 1840 annotated frames with 4988 VRU instances. The data covers situations from poorly lit environments (20% of the data) to well-lit sequences captured in daylight. A total of 148 unique VRUs, pedestrians, and cyclists were labeled with each test sequence containing at least one person. On average, there are 9 unique VRUs, with 34 occurrences per sequence. Due to the limited resolution of the available VLP-16 lidar, we were able to only confirm the presence of VRUs within a range of 20m. Thus all performance numbers are computed for detected and tracked objects within this range while the areas beyond 20m are considered as "don't care" regions where we ignore detections. The raw radar measurements and labels are freely available[1] and can be downloaded upon request.

[1] radar- fusion.ipids.ugent.be

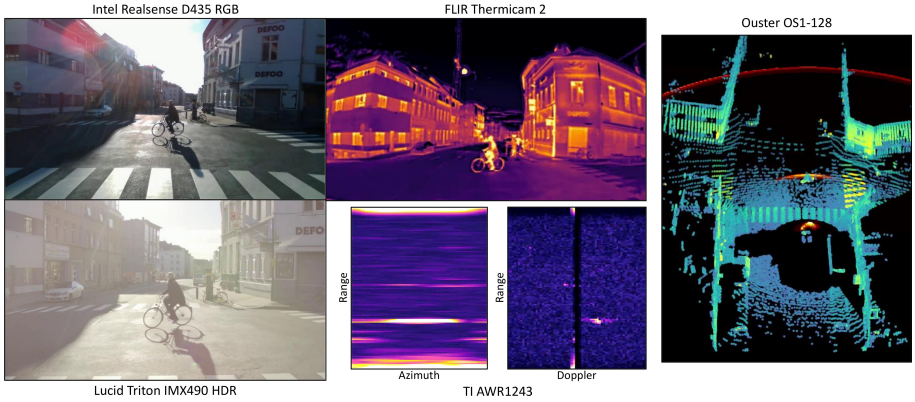


Figure 6.18: Example visualizing the data captured by the sensor array in the IMEC v2 dataset.

IMEC tracking dataset v2 (Gent, Bruges, Maldegem)

For the second campaign of data captures, we have upgraded our sensor array and capture vehicle which enabled us to record an order of magnitude more scenes than in Section 6.9.1. The goal of this data capture was, nonetheless, to offer raw radar measurements accompanied by camera and lidar scans in an AV setting. We used the 77GHz FMCW radar (Texas Instruments AWR1243) together with a full HD RGB camera module (Intel Realsense D435 in rgb mode), a Lucid Triton HDR camera with a Sony IMX490 sensor, a FLIR ThermiCam2 infrared camera and an Ouster OS1-128 3-D lidar, see example data in Figure 6.18. Each sensor was set up to capture at an optimal, constant frame rate and data frames were recorded and timestamped on an onboard PC. Note, similar to KITTI, our sensor array is forward looking with overlapping fields of view of the cameras, radar and lidar.

Our dataset consists of a total of 316 sequences of duration between 10s and 20s, captured throughout the year 2020 in several cities across Belgium. These sequences are representative of the difficult driving scenarios encountered in typical large cities in Western Europe: parked vehicles obscuring the view, mixing of motorized traffic with cyclists and pedestrians, occasionally narrow streets and an abundance of poorly thought out road infrastructure. Data labeling was performed in a semi-automated manner using an internally developed tool, [17]. An initial, automated pre-labeling, solution was generated by running object detection in the camera and lidar scans, then this solution was visually inspected and manually corrected by several annotators. The fully automatic labels contain 1936 unique VRUs across 179422 instances, while the manually corrected labels contain 1318 unique VRUs across 173095 instances. Ground truth is available in the form of 2.5-D bounding box annotations on the ground plane for the category vulnerable road users (pedestrians, bicyclists, mopeds, etc.). Due to the limited confidence in labeling distant objects, the ground truth only contains labels for objects within the range of [0m, 20m] and an azimuths of $\pm 35^\circ$ (the field of

view of the Realsense D435 camera) while the area beyond these ranges is ignored. We recommend using a spatial gate of 2m for accepting true positives.

This dataset was further labeled with various weather and traffic attributes: ambient light level, level of occlusion, VRU sub-category, ego-motion, object height and group forming behavior. Depending on the time of day of the capture, we split the data into three categories: "daytime" (160 sequences), "twilight" (98 sequences), and "nighttime" (58 sequences). Depending on the level of occlusion in the scene, we split the dataset into three categories: "visible" (where less than 10% of VRUs are occluded; 136 sequences), "partly occluded" (where 10% to 33% of VRUs are occluded; 136 sequences), and "occluded" (more than 33% of VRUs are occluded; 44 sequences). The level of occlusion of a VRU instance is an automatically computed categorical value. For each labeled VRU, we compare their ground truth distance to the median distance of the corresponding depth image patch. If the discrepancy of the labeled depth and the visible depth is more than 5m, then the VRU instance is marked as occluded. Depending of the object's velocity, we split the data into three categories: "mostly pedestrians" (112 sequences), "mixed" (60 sequences) and "mostly cyclists" (149 sequences). We consider VRUs moving faster than 6Km/h as cyclists. Manual inspection of the resulting split on a small random sample confirms that speed is a good indicator for VRU subcategory. Based on the object's height, we split the data into three categories: "mostly short" (97 sequences), "mixed" (200 sequences) and "mostly tall" (19 sequences). A sequence consists of mostly short or mostly tall people if more than 50% of the instances have height bellow 1.5m or above 1.75m, respectively. Depending on the ego-velocity, we split the data into two categories: "static" (54 sequences) and "moving" (262 sequences). A static scene is one where the average ego-velocity is below 10Km/h. Depending on whether there is group forming behavior in the sequences, we split the dataset into two categories: "no groups" (279 sequences) and "contains groups" (37 sequences). A group is formed when two VRUs are within 0.745m distance of each other [228]. We consider a sequence to contain significant group behavior if groups have been identified in at least 30% of the frames. Our benchmark tool performs automated tracking evaluation over the entire dataset as well as along all of the above mentioned weather and traffic qualities. The results of the experiments using this dataset are presented in Section 6.9.4.

6.9.2 Metrics

The proposed perception system is intended to serve as an input to automated collision avoidance and predictive path planning and therefore, it is of critical importance that the system recalls the most amount of VRUs at the lowest false alarm rate. The position estimates of detected VRUs should also be within an acceptable range of their true position in the real world. At the same time, the system should run on-line, confirming tracks with as little delay as possible. These contradicting requirements pose a non-trivial parameter optimization problem. In order to count an estimate as a true positive, its position has to be close to the ground truth. However, a state estimate needs to be made as quickly as possible, reducing the latency for emergency brak-

ing. Therefore we evaluate the performance of our tracker using the MOT16 benchmark methodology proposed in [227]. This methodology combines both quantitative, CLEAR metric [229], and qualitative, Track Quality Measure [230], tests of tracker performance. Since our system builds upon a standard object detection algorithm we will not benchmark the actual detection performance which was already benchmarked previously. We will herein shortly discuss the evaluation metrics, however, the reader is advised to refer to the work in [227] for more comprehensive elaboration and underlying reasoning.

1. Accuracy at a specific working point: refers to the classification performance expressed by the amount of true and false objects within the output of the perception system at a certain working point. The working point is defined by a specific confidence threshold which selects which hypotheses to report as detected. The metrics in this category correlate to the real-world accuracy of the Multi-Object Tracker by measuring the number of True Positives (TP), False Positives (FP) and False Negatives (FN) per and across all frames which is summarized as the F-score:

$$F_\beta = \frac{\left(1+\beta^2\right) TP}{(1+\beta^2)\, TP + \beta^2 FN + FP}, \tag{6.57}$$

where we will use $\beta = 2$ and compute the F_2 score. The intuition behind the F_2 score is that it weights recall $\left(\frac{TP}{TP+FN}\right)$ higher than precision $\left(\frac{TP}{TP+FP}\right)$. This makes the F_2 score more suitable in applications where it's more important to classify correctly as many positive samples as possible, rather than maximizing the number of correct classifications. The highest possible value of an F-score is 1.0, indicating perfect precision and recall, and the lowest possible value is 0, if either the precision or the recall is zero. For an object to be considered as a true positive, the respective algorithm output has to fall within a spatial gate of $1.5m$ of the ground truth. Multiple detections falling within this spatial gate are being counted as false positives.

Tracker robustness at the set working point is also tested by measuring the number of fragmentations (i.e., losing the track and starting a new one) for the same tracked object. This is indicated by the number of tracker ID switch (IDS) across all frames. An ideal tracker would improve upon the FP and FN rates of the object detector while at the same time have a fragmentation or ID switch score of zero. All of these metrics are combined into a single value called Multiple Object Tracking Accuracy (MOTA) which indicates the overall performance of the tracker at the specific working point. Formally MOTA is the ratio:

$$MOTA = 1 - \frac{FN + FP + IDS}{TP + FN}, \tag{6.58}$$

where $TP + FN$ is simply the number of Ground Truth objects. The highest possible value of MOTA is 1.0, indicating perfect tracking accuracy, and lowest possible value is unbounded indicating that the number of errors can greatly exceed the number of ground truth objects. Most trackers in the literature are compared primarily using

these metrics since they represent a good balance between tracking precision, recall, and temporal stability at an application-specific operating point.

2. Accuracy at multiple working points: refers to the classification performance of the tracker averaged over multiple working points. To that end we measure the Average Precision (AP) and Average MOTA (AMOTA) of the objects perceived by the system after tracking. AP is an excellent measure for the quality of the tracker, judging the capability to extract objects regardless of their identities, while minimizing false alarms. Effects such as reduced object recall and increase in false positives, signs of a low signal to noise ratio, will both reduce the AP score. Average MOTA additionally measures the temporal quality of the tracked objects with respect to the consistency of their identities. AP is computed by adjusting the tracking operating point to query the precision Q_{PR} at Q_{REC} uniformly spaced recall points:

$$AP = \frac{\sum_i^{Q_{REC}} Q_{PR,i}}{Q_{REC}}, \tag{6.59}$$

where precision defines the ratio detected relevant objects, true positives (TP), and all tracked objects including false positives (FP):

$$Q_{PR} = \frac{TP}{TP + FP}. \tag{6.60}$$

Recall defines the ratio of relevant objects being tracked and all relevant objects:Q

$$Q_{REC} = \frac{TP}{TP + FN}, \tag{6.61}$$

where FN are all relevant objects missed by the tracker. By analogy, AMOTA is computed as the average MOTA over the same uniformly spaced recall points:

$$AMOTA = \frac{\sum_i^{Q_{REC}} MOTA_i}{Q_{REC}}. \tag{6.62}$$

The highest possible values of both AP and AMOTA is 1.0 indicating perfect detection and tracking at all recall values. The lowest possible value for AP is 0, while the lowest possible value of AMOTA is unbounded.

3. Localization accuracy: Multiple Object Tracking Precision (MOTP) is the total position error for TPs over all frames, averaged by the total number of matches made. It shows the ability of the tracker to estimate the precise object position, independent of its skill at recognizing the object class, keeping consistent trajectories, etc. Essentially, it indicates the local accuracy of the tracker. MOTP is computed as:

$$MOTP = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t} \tag{6.63}$$

where c_t here denotes the amount of tracker-target matches in frame t and $d_{t,i}$ is the

spatial distance between tracked target i with the real object. For evaluation in the image plane the overlap is measured by means of intersection over union (IOU) while on the ground plane we compute the Euclidean distance and report in terms of average mean squared error (MSE) for matched object-hypothesis. The highest possible value for MOTP is equal to that of the gating radius for accepting true positives (1.5m in our experiments), while the lowest possible value is 0 indicating perfect localization.

4. Temporal track quality: These metrics measure the temporal consistency of tracking hypotheses by the amount of trajectories of the GT it covers. Mostly Tracked (MT) correspond to at least 80% coverage, and Mostly Lost (ML) means that track is only covered for less than 20%. Good tracking performance is generally indicated by a MT ratio closer to 1.0 and ML ratio close to 0. Another indicator is Fragmentation Number (FRAG), which is the number of a track interruption before it resumes the previously lost trajectory. The highest possible value for FRAG is the number of tracker output instances, indicating that each instance has a wrong identity and thus refers to a separate track fragment. The lowest possible FRAG value is 0 which indicates perfect tracking of the identities of the objects over time.

5. Tracking delay: is the time it takes for a track to become confident i.e., to reach the working point of the tracker. The delay is measured by the Track Initialization Delay (TID) metric, expressed in milliseconds, which averages the time from the first occurrence of the object in the ground truth labels until it reaches the confidence at which F-2, MOTA, MOTP, MT, IDS and FRAG scores are computed. The highest possible value for TID is unbounded (but generally equal to the length of the dataset) indicating that not a single object is detected or tracked for the entire duration of the dataset. Oppositely, the lowest possible value of TID is 0, which indicates perfect tracking with respect to its latency.

6.9.3 Experiments in simulation

In this sub-section, we assess the performance benefits of the switching observation model and multiple imputation strategy in a series of simulated experiments. For measuring the SOM performance, we track a single simulated object which crosses areas of the scene that are covered by different sensor fields of view. The hypothesis is that the SOM filter will switch to the correct sensor model when the tracked object crosses the boundary of an area covered by different sensors. We will measure how quickly the SOMPF converges to the correct observation model state. Using simulation for this experiment we can isolate the effects of the switching model mechanism and control for other effects such as multi-object association failures.

For assessing effectiveness of the proposed methods for handling missing data we compare the multiple imputation particle filter to several baseline trackers which use motion prediction. In this experiment we perform destructive testing by increasing the rate of missing observations. This simulation test is designed to measure the tracker's capability of handling missing observations in a single target tracking scenario. We generate noisy detections using a single sensor model which helps to reduce the effects of the live model estimation method tested in the first experiment. For this

experiment we report the positional accuracy of the estimate for trackers that do not handle missing detections compared to the MIPF. The hypothesis is that as the proportion of missing detections increases, tracking with multiple imputations is more effective than relying only on motion models.

Tracking with changing sensor modalities

In the first experiment, we simulate a traffic scenario where we track a single road user using a camera-radar sensor array. The sensors' fields of view do not entirely overlap, thus only a portion of the scene is visible to both, with the other parts being covered solely by the camera or radar, respectively. We allowed a computer-generated road user to follow a path that resembled a pedestrian crossing. Moving from a radar-covered region to a camera-covered area, the road user is briefly visible to both sensors as it goes left to right. Observations are generated at every time step following the respective sensor model and fed to a SOMPF tracker. The main motivation behind this simplified experimental scene is that we can eliminate external factors such as occlusion, loss of detection and ambiguous association from the tracking. The goal is then to analyze if the SOM helps the PF to converge to the correct sensor modality c, and if so, how quickly it converges.

The simulated pedestrian starts moving perpendicular to the ego vehicle starting from $\rho_0 = 20m$, $\theta_0 = -60^\circ$ with initial velocity magnitude of $1.38ms^{-1}$ and velocity orientation of $\theta = 90^\circ$. The motion is a stochastic process governed by the state evolution model explained in Section 6.5 characterized by a change in velocity defined in Algorithm 6.2. The array of artificial sensors produces simulated observations at regular intervals of $\triangle t = 100ms$. Since freshly observed sensor evidence does not always appear where it is expected, and since the tracker requires precise motion and sensor models, such lengthy time intervals reflect the worst case situation in tracking. In this manner, we produce a challenging case that stresses the switching observation model.

Both sensors are oriented in such a way that their combined field of view spans over $\pm 90^\circ$ with an overlap over the circular sector of $\pm 15^\circ$ in front of the vehicle. The radar also sense the area to the left of the overlapping region $[-90^\circ, 15^\circ]$, while the camera can also sense objects to the right $[-15^\circ, 90^\circ]$, see Figure 6.19. For generating the radar and camera observations on the ground plane, we use the following covariance matrices in polar coordinates:

$$R\left(\rho\right)^{(cam)} = \begin{bmatrix} 0.339\rho + 0.096 & 0 \\ 0 & 0.014^2 \end{bmatrix}, \quad R^{(radar)} = \begin{bmatrix} 0.170 & 0 \\ 0 & 0.344^2 \end{bmatrix}, \tag{6.64}$$

where ρ is the current range of the object.

Since we are using a particle filter, each particle has also a local contextual variable $c^{(m)}$ that allows the filter to update the weight of each particle using a different observation model. In this experiment we use four observation models: 0-degenerate mode where none of the sensors report an observation, 1-nominal radar-only mode

where the camera observation is missing, 2-nomnal camera-only mode where radar observations are missing, and 3-nominal camera and radar observations. For simplicity, in this experiment we do not model degraded states of individual sensor modalities or soft sensor failures (such as compromised precision due to occlusion or lower accuracy at long range).

The observation model estimate is computed as the mode of the set of the set of context particles: $\hat{c}_t = \text{mode}\left(\left\{c_t^{(m)}\right\}\right)$. For the evolution of each $\boldsymbol{\alpha}_t^{(m)}$ we use the model Eq. (6.44) specified as a Dirichlet process using the spread parameter $\sigma_{t_0} = 100$.

We performed multiple runs of the same experiment and observed that the SOM was able to converge to the correct sensor state each time. This finding confirms that the method is effective at estimating the local variations of the sensor characteristics from the observed data even when the detections arrive at a low frequency of $10Hz$. However, keep in mind that this is a simulated experiment with several complex issues removed, such as occlusion, multi-object data association, and soft sensor failures.

In Figure 6.19 we present qualitative results obtained from one simulation run. The plots on the top show a birds eye view (BEV) of the scene at three different moments in time. Correspondingly, the plots on the bottom show the evolution of our belief of the observation model against the ground truth. Each BEV plot visualizes the the sensor fields of view, the true trajectory of the object (red line), the estimated trajectory (blue line), the observations (marked with symbol " $\times$ ") and the posterior distribution of the current position as a particle cloud (colored circles). In order to better visualize the behavior of the estimate $\hat{c}_t$, Figure 6.19 shows the set average $\hat{c}_t \approx \sum_m^{N_{pts}} c_t^{[m]}$ (blue line) instead of the mode, and it compares this average against the true state (red line).

When the simulated pedestrian crosses a boundary of sensor coverage ($t_{c_1 \to c_3} = 7.0s$ and $t_{c_3 \to c_2} = 14.7s$) perturbations caused by the evolution model of $\boldsymbol{\alpha}_t$ allow for quick exploration of alternative sensor model solutions. On the plot in the middle of Figure 6.19 we can observe that the posterior takes a concentrated shape because most $c_t^{(m)}$ converge to the observation model of a fused camera-radar sensor. In this single-target scenario, our tracker shows quick convergence to the correct observation mode, needing an average of 5 updates. We expect that the convergence time in the real world, however, to be also affected by factors such as missing observations and, in the case of MOT, noise from faulty observation to track assignments. However, the results from this simulation show that the proposed SOM adequately selects the correct model without explicit information about the sensor layout.

Tracking with missing detections

In this set of experiments we analyze several methods for handling of missing detections using through a simulation of tracking road users, comparing the MIPF to standard trackers such as Kalman and particle filter. To that end, we simulate Missing At Random (MAR) detections which we feed to each tracker and measure the posi-

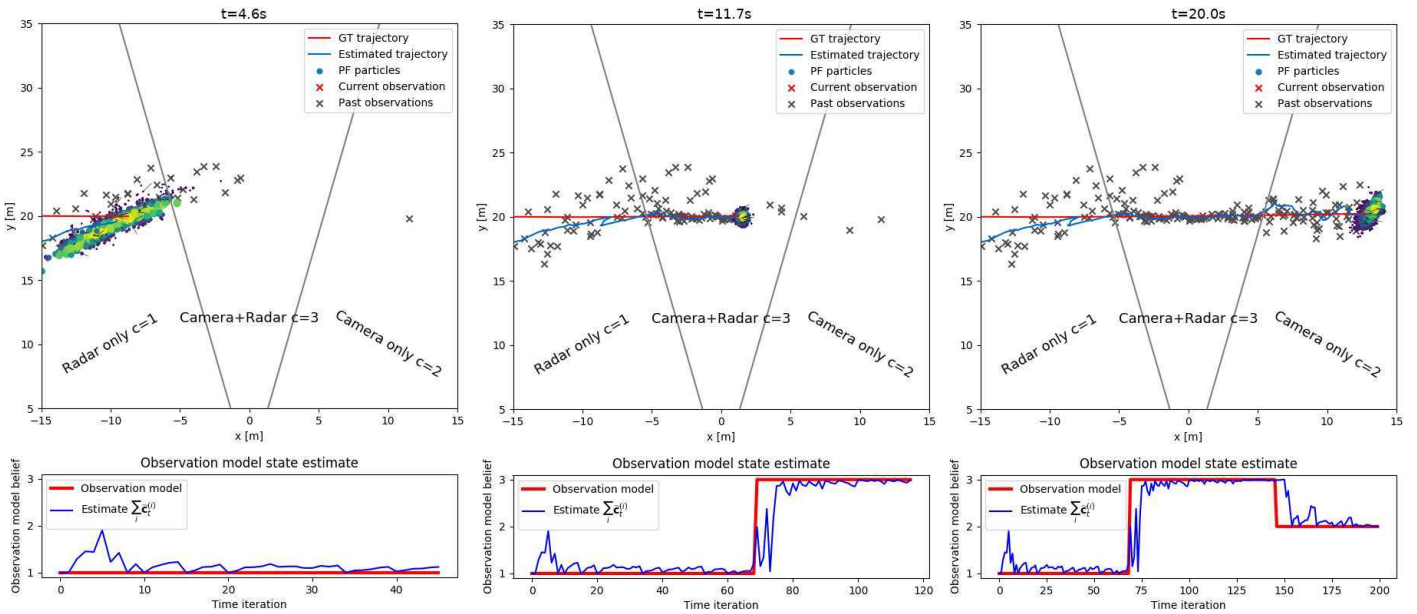


Figure 6.19: Tracking a simulated target with the switching observation model PF at 100ms time intervals. On the top we present a situational layout, while on the bottom we show the evolution of the indicator variable $\sum_i c_t^{(i)}$. Switching of the underlying sensor model happens at $t_{c_1 \to c_3} = 7.0s$ and $t_{c_3 \to c_2} = 14.7s$ while the model estimate switches to the correct state at times $\hat{t}_{c_1 \to c_3} = 7.6$ and $\hat{t}_{c_1 \to c_3} = 15.5s$.

tional error in the estimate with respect to the proportion of missing detections. By definition, the KF and PF do not have a mechanism for handling missing observations, however, in practice it is common to estimate the state using only the state evolution model. Using the appropriate motion model, we can expect that such trackers remain accurate when the ratio of observed vs. missing observations is relatively high. Since the performance of the MIPF is expected to depend on the imputation sample size n_{imp}, in this experiment we compare the tracking performance using various sizes of the imputation sample size.

In order to control for all other factors, for this experiment we use a simplified single object, single sensor setup. This allows us to isolate the impact of missing data on tracking performance. The same observation model parameters, which are utilized to create the observation data, are used by all trackers. PF and MIPF use the same amount of particles n_{imp} and the same behavioral motion model while the Kalman filter uses a constant acceleration motion model. We present our findings on the two plots in Figure 6.20 where on the left plot we show the positional errors of the estimates (in terms of MSE) of various trackers against an increasing proportion of missing detections, while on the right we show the respective average time needed for a single track cycle.

The baseline algorithms, whose results are shown with solid markers (blue: KF and red: PF) on the left plot on Figure 6.20 are computed using complete set of detections. The lines with square markers show the performance of Kalman filter, while lines with circle markers indicate Monte-Carlo trackers. As can be expected, tracking an object with non-linear kinematics is more accurately performed by a Particle Filter than a Kalman Filter. Increasing the proportion of missing detections raises the error in the state estimate for all trackers. In shades of orange we show the performance of the MIPF using $n_{imp} = 5$, $n_{imp} = 50$, $n_{imp} = 500$ imputations which relates to 1%, 10%, 100% of the particle size n_{pts}. Each data point represents a mean value from running 1000 simulations with perturbed object trajectories. From the left plot we can conclude with a degree of certainty the following findings for our experiments:

- Without missing detection, Particle Filters outperform Kalman Filter by around 20% in terms of lower MSE of the estimated position on the ground plane.
- When detections are missing, both Kalman Filter and standard Particle Filter have a marked decrease in the accuracy of their estimates. PF outperforming KF by around 20% regardless of the amount of missing detections.
- Using very few imputations (1% of the number of particles) makes the MIPF estimate worse than predicting the position solely on the state transition model (standard PF). This trend is consistent for all evaluated rates of missing observations.
- When using a larger number of imputations (more than 10% of the number of particles) the MIPF estimate becomes 7% more accurate than the standard PF in terms of MSE. This trend is also consistent for every rate of missing observations.

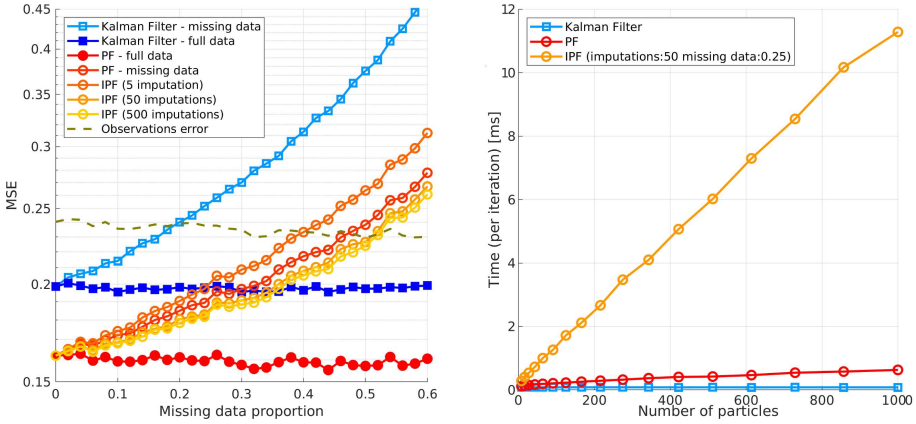


Figure 6.20: Performance evaluation for tracking a simulated VRU. On the left plot, we show the MSE for track estimates of several trackers at various missing data rates. On the right plot we show the average execution time for a single prediction-update cycle.

We measured that the MSE of the KF estimate becomes worse than the original detections when more than 20% of the data is missing. For the standard PF filter this point is reached at 47% data loss, while for the MIPF at 52%. The former finding confirms the limited capacity for handling non-linear motion of KF, while the latter indicates the possible benefits of using imputation theory in road user tracking. Even though we empirically optimized the Kalman filter parameters, the measured MSE of the estimate is consistently worse than the PF. We suspect that this is due to the highly non-linear motion model used to simulate the object motion i.e., the simulated object can sometimes make sudden random changes in their direction, accelerate or decelerate. Regardless, when no data is missing, all algorithms produce an estimate which is more accurate than the measurement, dashed line Figure 6.20. Thus the choice of an appropriate tracker can be made by considering other factors such as algorithmic complexity and memory requirements.

Finally, we measured the computational load of the analyzed trackers by measuring the average time required for a track prediction and update. On the right plot on Figure 6.20 we see the execution time of the three considered trackers. These graphs were generated by fixing the missing detection rate to 25% and the imputation size of $n_{imp} = 50$, a setting which represents a typical operational situation for road user tracking. The Kalman Filter uses a single state variable, while for the PF and MIPF we vary the sampling set size n_{pts} from 0 to 1000 particles. On the vertical axis, we report the average execution time of a single prediction-update cycle in miliseconds. All three trackers were implemented in MATLAB using optimized built-in functions and vectorization. The code runs in a single-thread on an Intel Core i7-4930K with 64GB memory. As expected, the time needed for a PF prediction and update increases with the increase of n_{pts}. This trend is also even more evident for the MIPF, where

the rate of increase is proportional to the imputation set size n_{imp}. We can see from the plot that all three algorithms can be employed in real-time tracking for a single object, however, design considerations need to be taken when tracking multiple targets. For example, our prototype system which we deploy in the following real-world experiments uses a sensor readout of $10Hz$ which limits the maximum number of targets at 1230 for KF, and 237 for PF ($n_{pts} = 500$) and 17 for MIPF ($n_{pts} = 500$ and $n_{imp} = 50$). In systems where the sensor measurements come at higher frame rates, the amount of objects that can be tracked by MIPF decreases which makes this algorithm inapplicable.

6.9.4 Real-world experiments

In this sub-section, we present the methodology and results of the tracking experiments conducted using pre-recorded data captured with real sensors in urban traffic environments. These experiments are designed to investigate the proposed tracker's real-world performance by assessing tracking accuracy under various combinations of poor sensor performance, tough traffic and weather conditions, as well as unexpected motion and interaction of road users. In the first experiment, the proposed tracker's tracking performance is compared against a Kalman filter, a PF, a PF with switching observation models (SOMPF), and a multiple imputations SOMPF. For this experiment we will use our own camera-radar dataset. Then, to further investigate the effects of the cooperative detector-detector and tracker-detector feedback on tracking, we performed experiments using our extended camera-radar-lidar dataset. Finally, we compared the performance of the proposed tracker to the state-of-the-art using two publicly available benchmarks: the KITTI tracking benchmark and the nuScenes tracking benchmark. We will hereby evaluate the validity of following hypotheses:

- **Tracking with missing detections:** when one or more of the detectors fail to detect objects, temporal tracking using the proposed method outperforms optimal techniques such as KF and PF(SOM), and multiple imputations filtering.
- **Tracking with cooperative feedback:** adding a detector-detector and tracker-detector feedback improves not only detection, but also most tracking metrics.
- **Performance with respect to the state of the art:** operating on the same set of detections, the proposed tracker outperforms state-of-the-art trackers on public tracking benchmarks.

IMEC tracking dataset v1 (Gent Zuid)

For testing the effectiveness of tracking with missing detections, we evaluated the performance of the proposed tracker to several optimal trackers: Kalman filter (KF), standard particle filter (PF), switching observation model particle filter [222] (SOMPF) and a multiple imputations SOMPF with 150 imputations. Since the IMEC v1 dataset was designed for the purpose of training and evaluating object detectors, it

lacks the annotation for object identities over time, and thus we cannot compute the metrics such as identity switches and track fragmentation rate or track initialization delay. However, we can still measure the detection performance of various trackers in terms precision and recall and compare these results against the instantaneous object detection methods from Chapter 5.

Each tracker was extended to multiple object tracking using the same track maintenance logic explained in Section 6.8. The KF uses a constant acceleration motion model empirically optimized to best approximate the non-linear models employed by the Monte-Carlo trackers. The number of particles in all experiments was kept the same at $n_{pts} = 1000$. The KF and PF use the single observation model while the SOM PF, the multiple imputations PF, and the proposed tracker switch between the same four observation models as in the simulated experiments. Particle re-sampling for the Monte-Carlo trackers is performed whenever the effective sample size of a track becomes lower than 20% of the number of particles: $n_{eff} \leq \frac{1}{5} n_{pts}$. For repeatability of the experiments, we also took the necessary steps to reset pseudo-random generator seed to the same initial value between all experiments.

We performed two experiments: firstly, by feeding the trackers with detections from the cooperative camera-radar detector from Section 5.5 and secondly, by feeding the trackers with ground truth detections using a simulated MAR mechanism. The *first experiment* is designed to measure the expected tracking performance in a deployment-ready setup where we cannot control for external factors and the missingness of detections is dictated by the detection neural networks, while the *second experiment* performs destructive testing simulating missing detections from the ground truth labels designed to measure the theoretical capacity for handling missing detections using a controlled missingness process. With these tests we want to investigate the degradation of tracking performance with the worsening of input detections, firstly by using realistic detections with MAR mechanism built-in by design of the detection CNNs, and secondly by controlling the MAR mechanism using an increasing rate of missing detections. We expect that the proposed tracker will be more immune to missing observations as it uses sub-threshold detection information which is better conditioned on the missing data. To that end, for the second experiment, we form a sub-set of camera observations that simulates detections missing at random a probability in the range of $p_{FN} \in [0, 1)$.

We present the results of our real-world people tracking experiment in terms of Average Precision computed over multiple recall values, see the bar plot on Figure 6.21. The height of the gray bars indicates the average precision of the detection input, while the height of the colored bars indicates the average precision of the tracking output for the evaluated trackers. Late fusion refers to the camera-radar detector with no-feedback, while "Cooperative fusion" additionally employs radar→camera cooperative feedback. KF, PF, SOM PF, MI-SOM PF are the Kalman Filter, Particle Filter, Switching Observation Model Particle Filter and Multiple Imputation Switching Observation Model Particle Filter, respectively. Additionally, we split our test set into sequences with simple and complex tracking situations based on the amount of objects present in the scene and the degree of tracking ambiguity caused by occlusion.

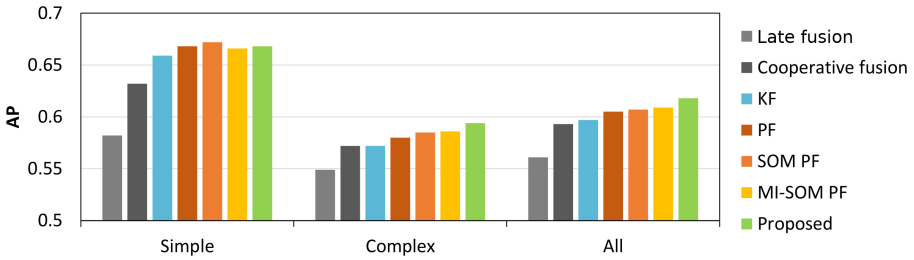


Figure 6.21: People detection (gray bars) and tracking (colored bars) performance expressed as average precision.

The "simple" sub-set consists of the sequences (1, 2, 3, 7, 8, 9, 10, 11, 12, 13) with 1150 frames and 1479 people, while the "complex" sub-set consists of the remaining 6 sequences with 690 frames and 3509 people.

From the first cluster of results visualized on Figure 6.21 we can conclude that all trackers perform significantly better in simple tracking situations than in complex ones which is also indicated by better object detection at the input. In simple situations, the best performing algorithm is the SOM-PF which uses state prediction based on motion model alone when detections are missing. In complex scenarios (middle bar cluster on Figure 6.21) however, the proposed tracker outperforms all other tracking algorithms. The same conclusion can be made when if we average the results for all sequences of the dataset (last bar cluster on Figure 6.21), where the proposed algorithm performs best, followed by the multiple imputations PF, then the other Monte-Carlo trackers and finally KF. This result confirms our hypothesis that the proposed tracker outperforms the optimal trackers under realistic, less than ideal, object detection.

We present the findings of our destructive testing using ground truth detections with simulated MAR mechanism in Figure 6.22. In each trial we increase the proportion of detections missing at random by 10% and measure the tracking performance for the above-mentioned trackers in both simple and complex sequences. In this second set of experiments we observed the same pattern as in the real world tests: tracking in simple situations achieves higher AP than in complex situations. This trend is true for all levels of missing detections. Additionally, we can conclude that all evaluated trackers demonstrate relatively good performance as long as more than half of the data is not missing. However, the Monte-Carlo trackers outperform KF in almost all but the trivial tests which shows their increased capacity for tracking the unpredictable motion of VRUs. As can be seen in Figure 6.22 the proposed tracker outperforms all other trackers when the amount of missing detections is between 20% and 80% in simple tracking situations and between 30% and 90% in complex tracking situations. We note that the proposed cooperative radar-camera fusion detector achieves an AP of 63.2% in simple and 57.2% in complex situations which sits in the middle of the simulated MAR rate ranges of this experiment.

These findings show that the given the same detections, the proposed tracker is capable of tracking people more confidently than existing methods, creating a better

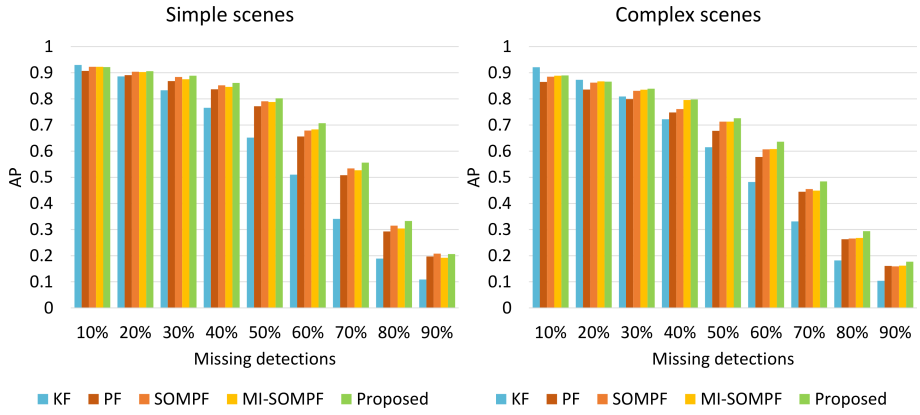


Figure 6.22: People tracking performance using simulated missing detections from the ground truth.

basis for safer collision avoidance. In all experiments we observed the same percentual improvements in F_2 score as in AP, and chose not to report both for the sake of brevity. We note that in real-world tracking situations the main benefit of temporal tracking is not only an increase in object recall, but also better positional estimate accuracy, reduced clutter and higher stability of object identities over time. These effects were impossible to measure without object identity labels which are unavailable in the IMEC v1 dataset, which is why in the following we experiment with a greatly extended version of the dataset which also includes object identity labels.

IMEC tracking dataset v2 (Gent, Bruges, Maldegem)

In this expanded set of experiments we perform fine-grained tracking evaluation of the proposed tracker on a dataset which spans a much larger set of traffic and weather conditions and also contains object identity labels, see example in Figure 6.23. The main goal here is to evaluate what are the effects of the cooperative feedback loops which we proposed in Chapter 4 on object tracking. Furthermore, we will investigate the tracking performance of various sensor combinations. The volume and variability of the dataset is an order of magnitude larger than the previous experiments which reflects in an increased significance of the findings herein.

The experiments are designed in a way that showcases the impact that various detection fusion methods have on the proposed tracker. In that regard, we will use multiple different schemes of lidar-camera-radar detection fusion and feed them to the same tracking algorithm, while measuring the accuracy of the tracked objects. For tracking objects with each sensor combination we will use the switching observation model tracker with the proposed sub-threshold grid method for handling missing detections.

For the camera detector we use the YOLOv3 object detector with default settings

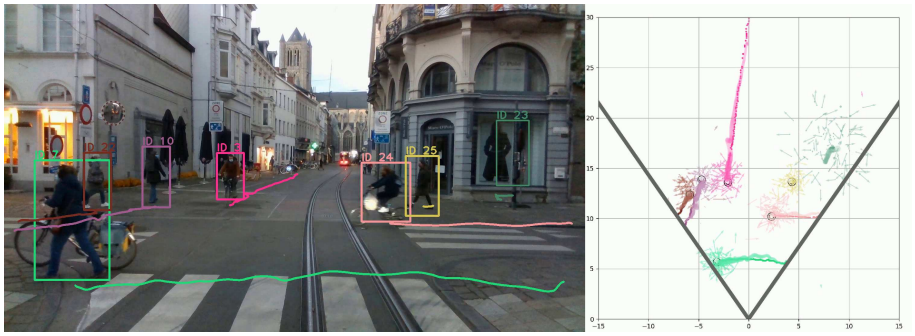


Figure 6.23: Visualization of tracked objects by the proposed tracker operating on fused camera-lidar detections from the IMECv2 dataset. Left: current state estimates and past trajectories rendered in the camera image; right: top-view of the scene illustrating the posterior distributions and trajectories. Same scene as the one shown in Figure 6.11.

and an input image resolution of 608x608px, for the lidar object detector we use the Centerpoint object detector [51] operating on the raw point cloud data, and for the radar detections we use our own Radar CNN as explained in Section 5.5. The baseline detectors operate on the late fusion principle, which we then compare to the proposed cooperate fusion. For camera-radar sensor combination we allow a two-way feedback, and for the lidar-camera-radar sensor combination we additionally allow feedback from the lidar to the camera. In both cases, feedback from the tracker in terms of predicted object positions flows back to the individual detectors.

Tracking is performed in global coordinates on the ground plane, where we apply our own odometry estimation algorithm. It is important to note that the tracker hyperparameters were re-optimized for every different sensor combination, however, we kept the same number of particles in all experiments. This way the experiments will demonstrate the differences in tracking accuracy achieved under equal algorithmic constraints in the tracker. We analyze the tracking accuracy using the following inputs:

- Camera-radar detections (late fusion)
- Camera-radar detections (cooperative fusion)
- Lidar-camera-radar detections (late fusion)
- Lidar-camera-radar detections (cooperative fusion)

In order to evaluate the strengths and weaknesses of the various sensor-sensor fusion schemes on object tracking, we compared their performance not only averaged across the dataset, but also across individual scene and environment settings using multiple performance metrics. The various evaluation criteria include global and local effects. Global effects such as time of day, ego-velocity, people density, scene complexity based on occlusion etc. do not change during individual sequences. These

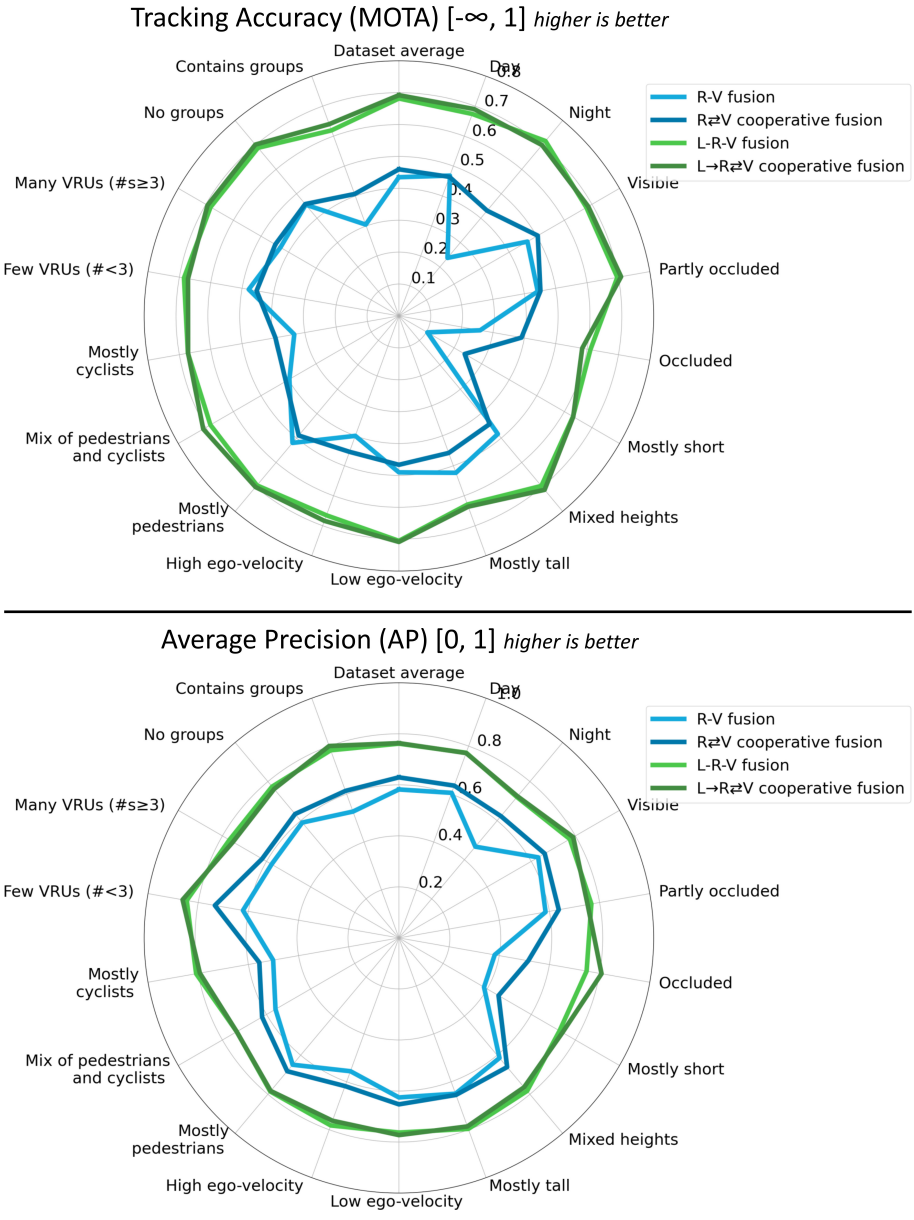


Figure 6.24: People tracking performance (AP and MOTA) breakdown on the IMEC v2 dataset.

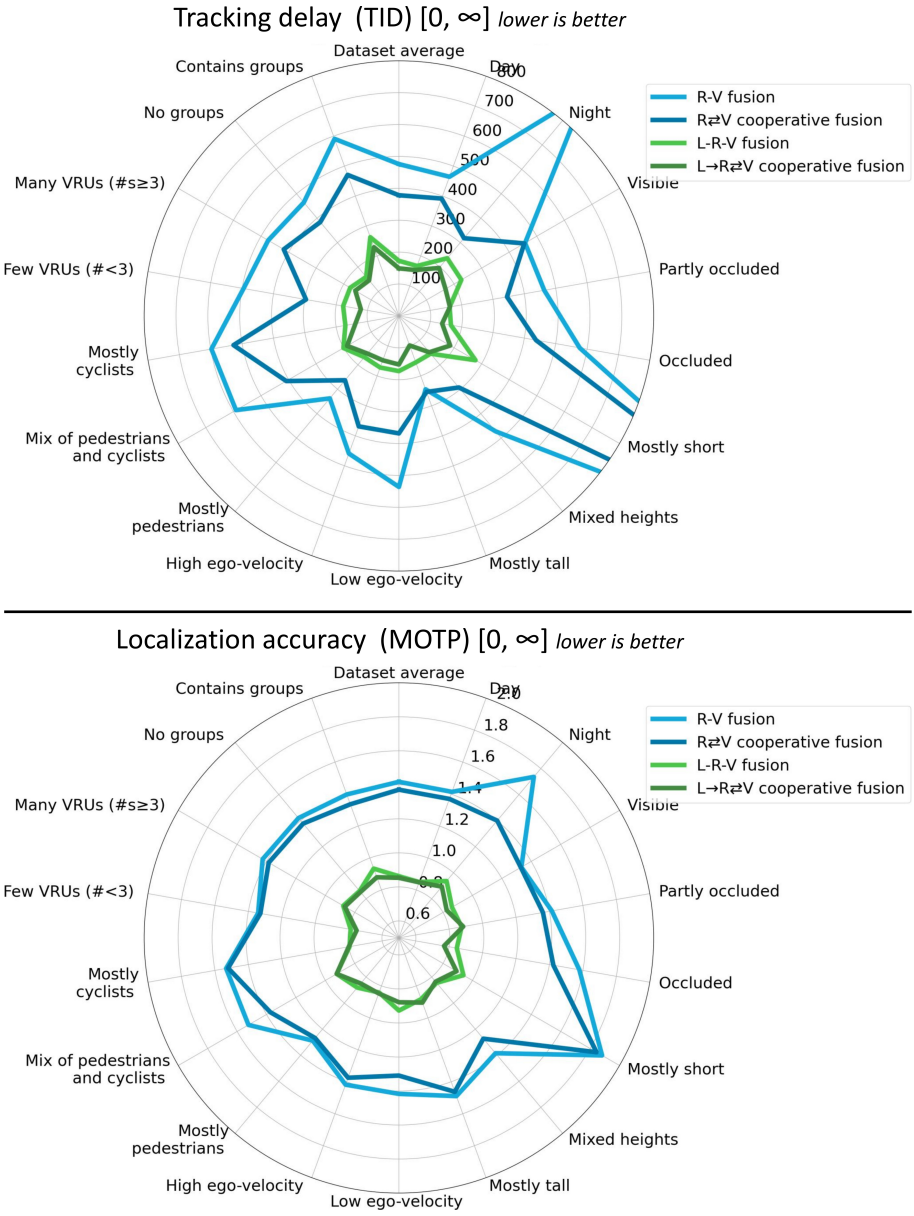


Figure 6.25: People tracking performance (TID and MOTP) breakdown on the IMEC v2 dataset.

criteria explain the more general tracking performance under a specific traffic setting or weather situation. Contrarily, the local criteria such as instance occlusion, person height, or motion pattern etc. are designed to selectively evaluate tracking performance with regard to specific characteristics of individual instances (observations) of objects, details in Section 6.9.1.

In all experiments we use the evaluation protocol from the KITTI tracking benchmark and the KITTI evaluation devkit software which we adapted to match ground truth labels and tracks on the ground plane. To that end we use a gating range of 2m Euclidean distance for tracker-ground truth association. Furthermore, we only evaluate targets and labels in the range of (0m,20m) and $\pm$35degrees, while objects beyond this range are considered as "don't care" and do not contribute to the results as they are of far lesser importance for making critical driving decisions.

The results shown in Figure 6.24 and Figure 6.25 demonstrate that, bi-directional cooperative radar-video fusion is significantly more accurate than classical late fusion (increase of 0.02 MOTA points and 0.05 AP points, 98ms lower TID and 4.6cm lower MOTP). The most significant improvements come in at night (increase of 0.19 MOTA points and 0.16 AP points, 590ms lower TID and 33.6cm lower MOTP) and for occluded objects (increase of 0.13 MOTA points and 0.13 AP points, 139ms lower TID and 15.5cm lower MOTP). The benefits of the cooperative feedback over late fusion are less pronounced when the system uses additional object detection from lidar (increase of 0.01 MOTA points and 0.002 AP points, 23ms lower TID and 0.9cm lower MOTP). We found that the lidar object detector is significantly more effective than the other detectors under circumstances of difficult lighting and bad weather. Cooperative fusion copes much better with difficult environment conditions than late fusion, only when there is a significant discrepancy in the performance of the employed sensors. The complete numerical results are shown in Table 6.1 and visualized as net-diagrams shown in Figure 6.24 and Figure 6.25.

KITTI dataset (Karlsruhe)

In the previous experiments we evaluated the performance of the proposed tracking algorithm against a late fusion control, using various detector combinations at input. We have shown that the tracker outperforms classical fusion methods and best performance is achieved when using lidar object detection. In this section we will compare the accuracy of the tracker against current state-of-the-art methods from the literature on a publicly available dataset using independent evaluation. For this experiment we used a camera-lidar fusion system where the camera performs detection and the lidar generates depth maps for ranging using the algorithm in Section 4.3.1.

For a fair comparison with the state-of-the-art, we run our tracker using the recommended camera detector "sub-category aware" (SubCNN) [231][2] which is also used by the competing method [232]. Image detections are matched to depth values from a completed depth image using our depth completion algorithm and then tracked by the

[2]available for download at: https://github.com/yuxng/MDP_Tracking

Condition \ Method	MOTA↑				AP↑				TID↓ [ms]				MOTP↓ [m]			
	R-V Late Fusion	R≓V Coop. Fusion	L-R-V Late Fusion	L→R≓V Coop. Fusion	R-V Late Fusion	R≓V Coop. Fusion	L-R-V Late Fusion	L→R≓V Coop. Fusion	R-V Late Fusion	R≓V Coop. Fusion	L-R-V Late Fusion	L→R≓V Coop. Fusion	R-V Late Fusion	R≓V Coop. Fusion	L-R-V Late Fusion	L→R≓V Coop. Fusion
Dataset average	0.435	0.460	0.682	**0.692**	0.582	0.629	0.761	**0.763**	476.1	378.4	172.8	**149.2**	1.418	1.372	0.863	**0.854**
Day	0.468	0.464	0.674	**0.690**	0.605	0.635	**0.772**	0.771	464.4	391.7	166.8	**153.1**	1.414	1.370	0.850	**0.848**
Night	0.237	0.431	**0.716**	0.699	0.467	0.623	0.721	**0.724**	908.6	318.1	236.2	**196.4**	1.736	1.400	0.937	**0.893**
Visible	0.466	0.503	0.680	**0.687**	0.631	0.661	0.775	**0.790**	457.5	454.7	226.5	**168.9**	1.333	1.329	0.861	**0.825**
Partly occluded	0.443	0.451	0.696	**0.708**	0.585	0.637	**0.767**	0.752	463.5	344.7	**161.7**	162.8	1.414	1.361	**0.875**	0.884
Occluded	0.259	0.391	**0.611**	0.584	0.382	0.516	0.748	**0.808**	576.5	437.1	165.9	**137.9**	1.579	1.424	0.846	**0.770**
Mostly short	0.103	0.239	0.631	**0.632**	0.386	0.452	0.728	**0.741**	1473.9	1566.5	277.8	**185.9**	1.881	1.844	0.939	**0.892**
Mixed heights	0.483	0.443	0.696	**0.712**	0.614	0.661	**0.783**	0.764	472.4	292.9	154.7	**148.7**	1.385	1.273	0.845	**0.836**
Mostly tall	0.523	0.457	0.629	**0.636**	0.650	0.655	**0.796**	0.786	244.9	253.6	155.1	**98.5**	1.490	1.463	**0.883**	0.905
Low ego-velocity	0.490	0.467	0.705	**0.709**	0.625	0.652	0.763	**0.773**	536.2	368.3	173.0	**152.3**	1.417	1.309	0.927	**0.878**
High ego-velocity	0.400	0.453	0.665	**0.684**	0.556	0.618	**0.782**	0.762	459.1	368.5	172.1	**148.0**	1.418	1.375	**0.842**	0.847
Pedestrians	0.520	0.489	0.694	**0.700**	0.649	0.682	**0.786**	0.783	337.5	263.2	168.7	**154.0**	1.289	1.268	0.881	**0.850**
Ped. and cyc.	0.397	0.410	0.683	**0.709**	0.559	0.620	**0.735**	**0.735**	590.7	408.4	201.1	**187.7**	1.523	1.374	**0.923**	**0.923**
Cyclists	0.334	0.394	**0.673**	0.672	0.501	0.556	**0.808**	0.793	597.8	527.4	170.7	**138.6**	1.533	1.521	**0.795**	0.800
Few VRUs (<3)	0.479	0.454	**0.685**	0.672	0.622	0.734	0.846	**0.862**	495.3	296.3	177.8	**121.3**	1.342	1.328	0.788	**0.753**
Many VRUs (≥3)	0.430	0.448	0.683	**0.694**	0.577	0.620	**0.768**	0.752	473.9	417.6	176.6	**158.1**	1.426	1.385	0.878	**0.868**
No groups	0.455	0.458	0.688	**0.701**	0.590	0.634	**0.774**	0.762	464.5	384.1	159.8	**145.7**	1.420	1.377	0.860	**0.852**
Contains groups	0.305	0.407	0.619	**0.640**	0.527	0.613	0.783	**0.801**	590.9	470.7	262.6	**229.3**	1.398	1.336	0.934	**0.880**

Table 6.1: Tracking results on the IMEC v2 people tracking dataset (R-radar, V-video, L-lidar).

proposed tracker on the ground plane. In these experiments we used additional optical flow fields (computed using the method described in [233]) to provide additional velocity measurements for the velocity of particles. The optical flow vectors represented as image pixel displacement vectors are transformed to ground plane motion using the camera calibration matrix and corresponding depth image.

Since the KITTI tracking benchmark measures tracking performance by means of MOTA computed at a single operating point, we output tracks with above a track score $\Lambda_t(h) > \tau$ that produces optimal MOTA, balancing between true positives, false positives and identity switches. The optimal value for the threshold parameter τ was obtained empirically by testing on a small validation sub-set of the data. Furthermore, we adjust the parameter τ use an adaptive mechanism based on the ego vehicle velocity (expressed in km/h):

$$\tau = \left[1 - \exp\left(\frac{-(v_{ego}+2)^2}{\sigma^2}\right)\right]^{\gamma}, \tag{6.65}$$

where the $\gamma = 0.1$ and $\sigma = 1$. This choice of parameters makes the threshold values lower when the vehicle is static and close to 1 if the vehicle is moving fast. This adaptive threshold is based on the belief that the overall performance of the tracker decreases with the increase of ego-velocity (as we saw in the previous experiment), thus the need for higher track existence threshold with the higher ego-velocity.

Based on the track score $\Lambda_t(h)$, a track is removed from the system when the score is below a threshold. In practice we found out that the optimal track removal threshold for the KITTI dataset is reached after a missing detection for 50 consecutive frames (5 seconds). Re-identification is performed for every fresh track that stays unassociated with the currently active tracklets. We do re-identification based on a combination of appearance similarity $KL\left(\mathbf{f}^{(k)}, \mathbf{g}\right)$ (Kullback-Leibler divergence between the detection and track HSV histogram) and position on the ground plane. Each tracked pedestrian is represented by the state estimate (mode) of the proposed particle filter for which we use $n_{pts} = 1000$ particles. The tracker takes around 26ms for the entire prediction, similarity computation, data association and update chain. The system runs on a Linux PC equipped with a six-core CPU i7-4930K, 64GB main memory and a Geforce TitanX (Pascal) graphics card. Without taking into account external data sources such as pedestrian detection or odometry, our tracker runs in real-time. A working proof of concept tracking code was written in GPU-accelerated Python (using cuDNN and Torch) and the Quasar programming language [47].

On table 6.2 we provide a snapshot of the official results table for the KITTI tracking competition at the time of submission (19.09.2018) where we compared our results to 20+ state-of-the-art trackers. Our tracker outperforms all published trackers with MOTA score of 0.504, Figure 6.26 top left. In the MOTP metric, which measures the positional accuracy of tracked pedestrians, we again achieve state-of-the-art performance among all published trackers with a score of 0.728, Figure 6.26 top right. In total, we report 14059 true positives with only 2043 false positives and 9207 false negatives. This high score indicates that the proposed method is able to reliably track

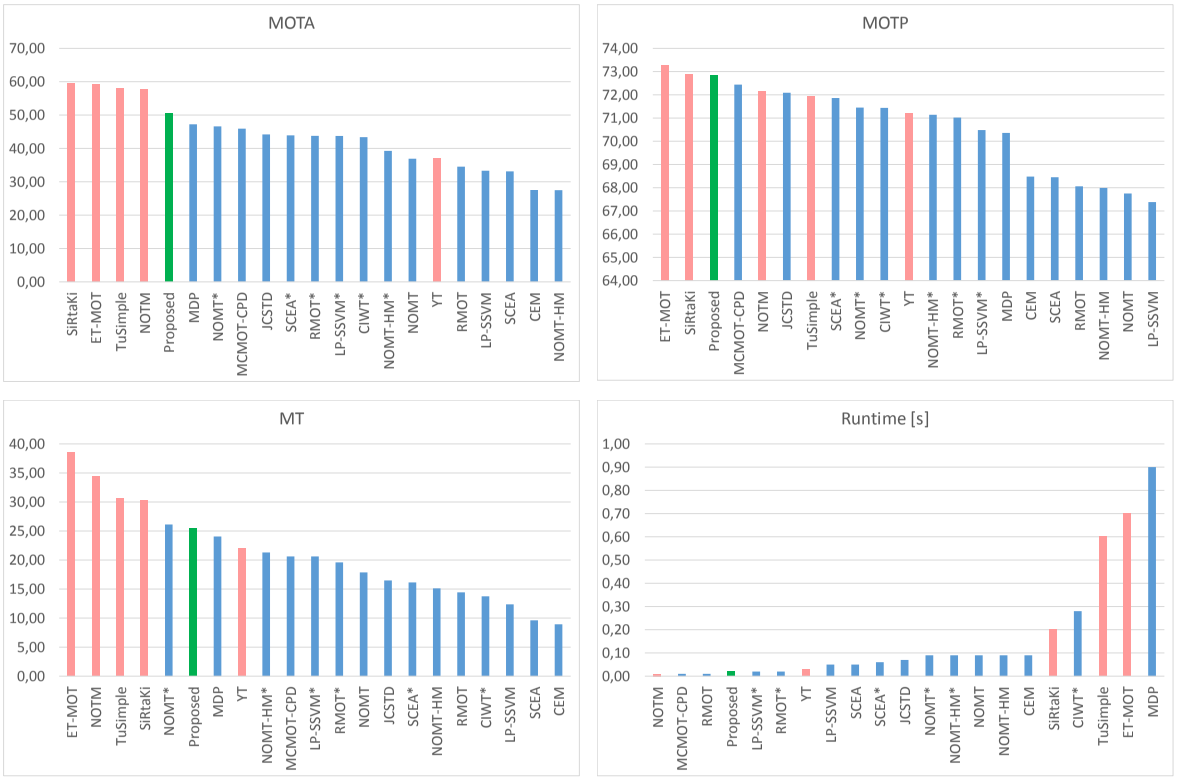


Figure 6.26: Comparison of the proposed method with various evaluated Pedestrian trackers on the KITTI tracking test set. Red indicates trackers with unknown source, blue bars are trackers published in the literature and green bars are results from the proposed tracker. A higher score MOTA, MOTP and MT score is better.

Method	Setting	MOTA	MOTP	MT	ML	IDS	FRAG	Runtime	Source
SiRtaKi	on	59.61 %	72.89 %	30.24 %	15.81 %	136	1164	0.2 s / GPU	Unknown
ET-MOT	on	59.10 %	73.26 %	38.49 %	10.31 %	316	1362	0.7 s / GPU	Unknown
TuSimple	on	58.15 %	71.93 %	30.58 %	24.05 %	138	818	0.6 s / 1 core	Unknown
NOTM		57.67 %	72.17 %	34.36 1.5%	19.24 %	108	799	0.01 s / 1 core	Unknown
Proposed	la on	50.39 %	72.85 %	25.43 %	27.84 %	235	967	0.02 s / GPU	PROPOSED
MDP	on	47.22 %	70.36 %	24.05 %	27.84 %	87	825	0.9 s / 8 cores	[232]
NOMT*		46.62 %	71.45 %	26.12 %	34.02 %	63	666	0.09 s / 16 cores	[185]
MCMOT-CPD		45.94 %	72.44 %	20.62 %	34.36 %	143	764	0.01 s / 1 core	[234]
JCSTD	on	44.20 %	72.09 %	16.49 %	33.68 %	53	917	0.07 s / 1 core	[235]
SCEA*	on	43.91 %	71.86 %	16.15 %	43.30 %	56	641	0.06 s / 1 core	[236]
RMOT*	on	43.77 %	71.02 %	19.59 %	41.24 %	153	748	0.02 s / 1 core	[237]
LP-SSVM*		43.76 %	70.48 %	20.62 %	34.36 %	73	809	0.02 s / 1 core	[238]
CIWT*	st on	43.37 %	71.44 %	13.75 %	34.71 %	112	901	0.28 s / 1 core	[186]
NOMT-HM*	on	39.26 %	71.14 %	21.31 %	41.92 %	184	863	0.09 s / 8 cores	[185]
NOMT		36.93 %	67.75 %	17.87 %	42.61 %	34	789	0.09 s / 16 core	[185]
YT	on	36.90 %	71.22 %	21.99 %	25.43 %	267	995	0.03 s / 4 cores	Unknown
RMOT	on	34.54 %	68.06 %	14.43 %	47.42 %	81	685	0.01 s / 1 core	[237]
LP-SSVM		33.33 %	67.38 %	12.37 %	45.02 %	72	818	0.05 s / 1 core	[238]
SCEA	on	33.13 %	68.45 %	9.62 %	46.74 %	**16**	717	0.05 s / 1 core	[236]
CEM		27.54 %	68.48 %	8.93 %	51.89 %	96	608	0.09 s / 1 core	[180]
NOMT-HM	on	27.49 %	67.99 %	15.12 %	50.52 %	73	732	0.09 s / 8 cores	[185]

Table 6.2: Results on KITTI pedestrian tracking dataset sorted by MOTA score. Setting on=On-Line, la=Laser Data, st=Stereo Data. Red rows belong to submissions from unknown sources.

most of the pedestrians without creating too many false positives. The temporal quality score MT ranks our method as $\#2$ with $MT = 25.43$ only behind [185] which is a "near on-line" method. We also report the recall rate of our method at 0.604 with a precision of 0.873 which gives an F1 score of 0.714. The number of identity switches for the KITTI test benchmark is 235 with a total of 967 fragmentations.

We suspect that the MOTA score relies heavily on the quality of the input object detections. Thus, using a detector with high recall and precision rates can potentially yield a tracker with higher MOTA. Our proposed tracker uses the same bounding boxes as [232] and by exploiting the depth information from lidar point clouds we are able to significantly improve upon the state-of-the art. Our method does the learning of motion behavior parameters off-line and has a much faster execution time. However, as seen on the results page, there exist trackers (with unknown source) that produce even higher MOTA scores. This can be due to many factors which are currently unknown and can not be objectively compared.

We conclude our analysis of the performance in the KITTI dataset with several qualitative tests that represent the border cases of our tracker i.e., we show several difficult scenarios where tracking is performed with high accuracy and several cases where it fails. In a classical multi-object tracking setup, most of the errors come when objects interact with the background or with each other. When a person walks past another person or an occluding object, parts of him/her become non-visible. Candidate object detections in such cases might fail completely or become less confident. This, in turn, causes problems in the data association where cost functions become ambiguous and matching becomes false. Our tracker tackles these problems using depth and azimuth information. In our experiments, border cases with difficult occlusions and sudden appearance changes become trivial. On Figure 6.27 we present two cases of

difficult occlusion where our method is able to continue tracking once the respective person re-appears without any loss of precision in the meantime.

In Figure 6.27, A) the vehicle is standing still on a pedestrian crossing and groups of people are walking in opposite directions. This sequence is particularly demanding since several occlusions happen in the middle portion that is covered by a shadow. Here appearance models change quickly and can not be updated due to occlusion. We observe that our tracker is able to keep track of every person regardless of occlusion which we suspect is possible only because we use additional depth and azimuth information in our particle filter. Once the tracked person becomes occluded, the particles continue to update based on the prior behavioral motion. When the person re-appears again, his/hers expected 3D position can be accurately matched to new measurements.

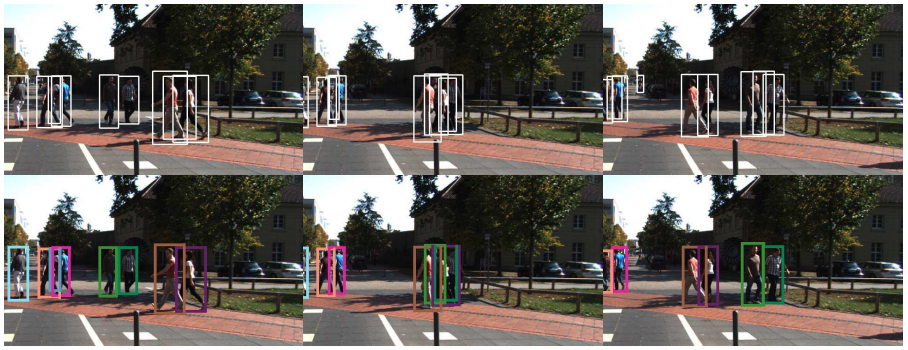
The scenario in Figure 6.27, B) presents a situation where the EGO vehicle is moving and detected pedestrians to the left are occluding more distant pedestrians in the background. Due to the relatively low frame rate, these occlusions happen in quick succession and the background person is only detected every other frame. However, from the images shown on the figure, it is visible that our tracker can estimate the position of the occluded person and easily re-engage tracking when fresh detections appear. In this scenario, much of the tracking information comes from the optical flow and odometry estimation. By knowing the distance of the person prior to his/hers occlusion, the tracker can estimate the optical flow field at that distance and continue to adjust the particle positions accordingly. This way, when the person re-appears, the expected location closely matches new detections.

Lastly, the three frames in Figure 6.27, C) visualize some of the fail cases of our method. Here highly confident tracklets appear in positions where there is a random object with a pedestrian like appearance. Since our tracker is based on detection and runs on-line, it makes the most informed decision in the present. Such tracking is not optimal since tracked objects can be far away and detection confidence might, at the present time, be low. Once the EGO vehicle drives close to these objects, their appearance improves and the detector disregards them as background objects. In the literature such ambiguities are easily solved off-line by forward-backward error validation, however, the problem still remains as one of the fundamental issues in detection based on-line systems.

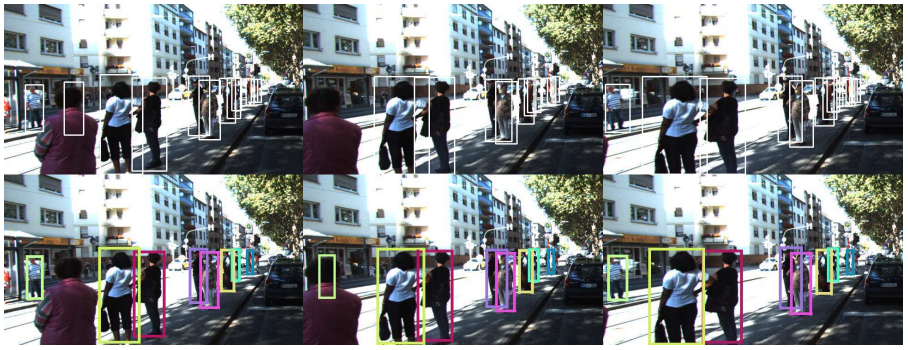
NuScenes dataset (Boston, Singapore)

This section presents an overview of our experimental evaluation of the proposed tracker against the state of the art on the public benchmark NuScenes [168]. NuScenes provides multi-sensor data covering even broader range of driving scenarios than KITTI and is therefore one of the most popular benchmarks for perception algorithms to date. Specifically, the NuScenes benchmark provides 6 camera views captured in the visible light spectrum and a 360 degree 3-D point cloud captured by a 32 beam Velodyne lidar. Each frame is labeled for the position and orientation of road users in the following categories: car, truck, bus, trailer, construction vehicle, pedestrian motorcycle, bicycle, traffic cone and barrier. The vehicle ego-position is recorded by

A) *Difficult lighting and occlusion:*



B) *Vehicle motion and occlusion:*



C) *Fail cases:*

Figure 6.27: Examples of our tracking output in typical situations that are considered difficult in a classical tracking setup. Frames with raw pedestrian detections are shown on top of tracked pedestrians for the respective sequence. In A) the focus is on the pair of people (green tracklets) passing with the other pair of people and coming out from a tree shadow. While their appearance changes and there is a complete occlusion, they are effectively tracked. In B) the focus is on the person across the street who gets occluded due to the motion. Our method is able to compensate for this motion and continue tracking when he becomes visible again. In C) we show three situations of persistent clutter where our method is tracking random objects that resemble pedestrians.

an IMU and sequential data frames can be registered within a reasonable accuracy. We use off the shelf state-of-the-art object detectors to detect candidate objects on the individual data streams and perform fusion at the object candidate level. Fused objects are then integrated over time and tracked in global 3-D coordinates using the IMU ego-position.

Contrary to many approaches in the literature which apply early or late fusion, we use our cooperative camera-lidar fusion method to compute the input for the tracker. The proposed system consists of a camera and lidar processing pipelines, a cooperative fusion module and a tracker. Up to 10 consecutive point clouds captured by the lidar are registered into a single, dense, point cloud which represents the 3-D structure of the environment. We apply an off-the-shelf lidar object detector Centerpoint [51] using the aggregated point cloud as input. This CNN detects the position, shape, orientation, velocity and class of objects. It achieves state of the art performance on the NuScenes detection benchmark, while the paper is one of the most cited in 2021. Centerpoint has an efficient implementation in PyTorch and allows for commercial usage through the MIT license.

The camera processing pipeline consists of 6 simultaneously captured images by individual cameras oriented with, slightly overlapping, fields of view. We apply the state of the art FCOS3D [52] camera object detector on each image and then aggregate the individual image detections into a common detection list using the camera calibration matrices. FCOS3D is a monocular 3-D object detector, providing the same output structure as Centerpoint. However, due to the monocular data input, the distance component of estimated objects has poor quality and we only use the output information in the image plane. In the fusion module, we use 3-D information and classification output from Centerpoint and fuse it with the image plane information and classification outputs from FCOS3D. In order to accurately match objects detected in the point cloud to image detections we project the lidar detections onto the correct camera view using the camera calibration matrices as explained in Section 5.6. Furthermore, we apply cooperative feedback links from the lidar to the camera and from the tracker to detector.

The output of the fusion module is a single list of detections having detection information optimally fused from the two detection sources. Theoretically, these fused objects should be no worse than the best detector in the ensemble in terms of average precision. In order to confirm that the proposed fusion method operates as expected we performed detection analysis on the NuScenes object detection dataset. We compared the outputs of the individual camera and lidar detectors to the fused detections on the validation sub-set for which ground truth data is readily available. It is important to note that the FCOS3D camera object detector performs poorly when evaluated on the ground plane labels due to the loss of range information in the camera images. To gain a better insight of the true detection performance of the camera detector we modified the NuScenes evaluation code to project the 3-D ground truth labels onto the camera view and evaluate on the camera plane using the standard 2-D IoU index at four thresholds (0.3, 0.4, 0.5 and 0.6). This way errors in the range estimation by FCOS3D will not influence the computed average precision.

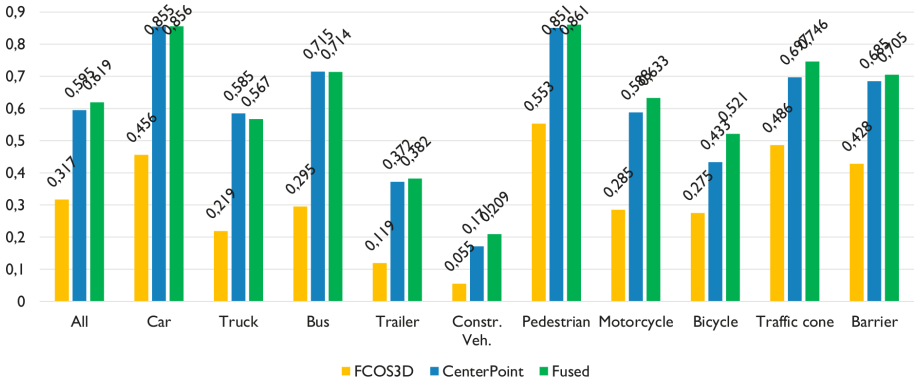


Figure 6.28: Detection performance on the NuScenes validation subset expressed in average precision (higher is better). Detections of "FCOS3D" are evaluated against ground truth labels projected on the image plane, while "Centerpoint" and "Fused" are evaluated on the ground plane.

The bar plot on Figure 6.28 shows the measured average precision values for detected objects of various classes as well as the mean average precision for all classes. On average we observed an increased detection performance by 4% (0.619 vs. 0.595) by fusing camera and lidar detections. The gains in detection performance are most noticeable for the classes "construction vehicle" 22.2%, "bicycle" 20.3%, "motorcycle" 7.6%, "traffic cone" 7%, while for the rest of the classes the benefits are smaller. On Fig.6 we present one set of precision recall curves computed at evaluation threshold of 2.0m. On this plot we can observe the same trend of increased precision at any given recall value for the fused detections.

Contrary to KITTI, the nuScenes tracking benchmark measures average tracking accuracy (AMOTA) at multiple operating points by varying a threshold over the track score. Additionally, this evaluation benchmark measures the typical tracking metrics such as: track positional accuracy, consistency of identities, track fragmentation, track initialization delay, etc. However, the official ranking is made according to the AMOTA metric which summarizes the false positive rate, false negative rate and identity switch rate at multiple recall levels. Therefore, in order to compare to the state of the art on these benchmarks we tune our tracker to obtain the most true positive tracks at each false positive rate, while at the same time reducing the number of identity switches. This diminishes the effects of the sensor-sensor and tracker-sensor cooperation since the feedback loops are only effective when optimized to a specific operating point. For example, the proposed tracker in the previous experiments was optimized for high-precision and used a high-recall secondary update step for ambiguous or missing detections. For NuScenes we are inclined to optimize for high-recall which improves the benchmark metric, but also severely increases the computation time. Note that at deployment, the tracker needs to be fine-tuned to a specific operating point which best satisfies the operational requirements such as false alarm rate

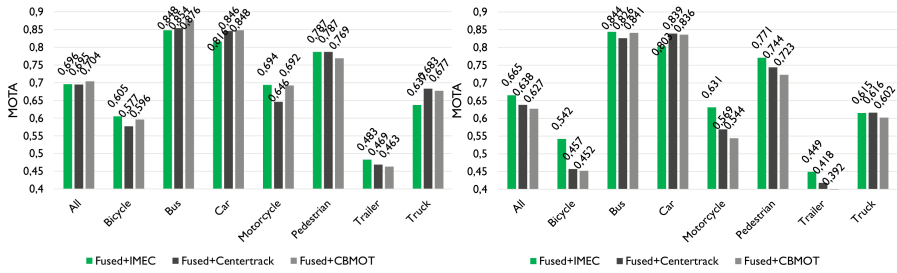


Figure 6.29: Tracking performance using the same set of high-recall, fused camera-lidar detections on the NuScenes validation sub-set. Left: using the official nuScenes devkit, right: evaluation in an online setting.

or identity switch rate. This is usually not the same operating point that is used for the public competition at NuScenes. We will investigate these effects at the end of the section.

For maximal AMOTA on the benchmark, the proposed tracker was adapted to a pure tracking by detection principle. Each detection is associated to a track, while left-over detections spawn a new track and unassociated tracks are removed after several steps of inactivity. For best tracking performance the input detections are not subject to threshold and every detection is allowed to spawn a track. To speed up the execution time we use a bootstrap particle filter algorithm which uses the motion model as the proposal function. A track is terminated after 1.5s of inactivity, meaning that it has not been associated with a detection for more than 1.5s. We use the proposed behavioral motion model which randomly mutates the radial and tangential components of each particle velocity at every time step.

Due to the limited number of submission trials to the official NuScenes tracking competition, which uses test set with labels unknown to the user, we resorted to the NuScenes validation benchmark for which ground truth labels are available for offline experimentation. The data in this validation set consists of an equal amount of sequences (150) and has content similar to the test set. According to the respective papers, the FCOS3D and Centerpoint detectors have not been trained on the NuScenes validation data. We used the official benchmark code to conduct tracking evaluation using the NIPS 2019 tracking settings and compared to two state-of-the-art trackers: Centertrack [51] and CBMOT [239], see left graph in Figure 6.29. At the time of writing these two methods were the two best performing camera-lidar, open-source trackers tested on the nuScenes dataset.

Our tracker reaches MOTA score that is competitive to the state of the art score of CBMOT (-1.1%) over all object classes. Importantly, our method outperforms the state of the art for the vulnerable road user classes “bicycle” (1.5%), “motorcycle” (0.3%) and “pedestrian” (2.3%), but scores lower for the classes of larger vehicles “bus” (-3.3%), “car” (-3.9%) and “truck” (-6.2%). Although the differences are marginal, in the following we provide a possible interpretation. The improved

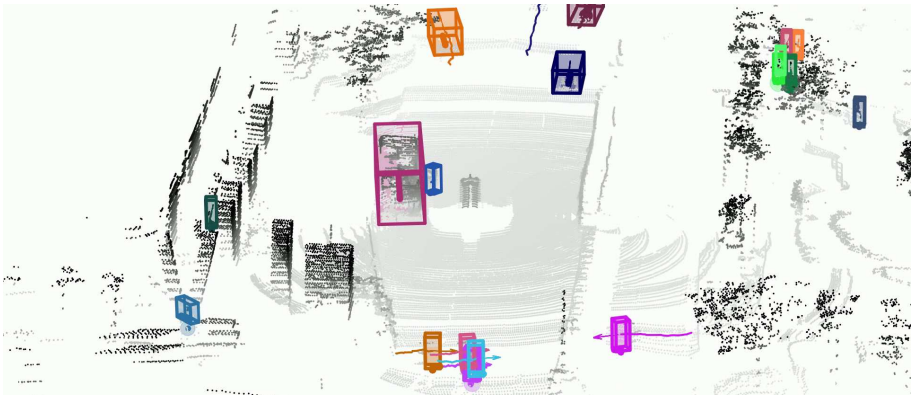


Figure 6.30: Visualization of tracked objects by the proposed tracker overlaid on the input lidar point cloud; nuScenes scene-0061.

tracking for the vulnerable road user classes can, to a degree, be explained by the fact that our state and motion models were been specifically developed for tracking pedestrians and cyclists. We suspect that our models consider too much freedom in the location and motion when tracking objects with larger mass. We did not fine-tune the motion models to the dynamics of each class, which would be a straight-forward quality improvement advised for the future research or when deploying the system in the real-world.

While using the standard NuScenes devkit we discovered a weakness in the way the code evaluates on-line trackers against the labeled ground truth. Specifically, prior to computing the tracking metrics, the evaluation code performs pre-processing of the submitted results by averaging track scores over their life cycle, formally:

$$\Lambda_t^{'}(h) = \frac{\sum_{i=t_0}^{t} \Lambda_i(h)}{|\{t_0, ..., t\}|}, \tag{6.66}$$

where $\Lambda_t^{'}(h)$ is the new track score for the hypothesis h, and takes the same (average) value at each time instance. Replacing the current track score with the average over its life cycle introduces future information (breaks the causal system assumption) and makes the evaluation protocol less than ideal for assessing on-line tracking performance. This remark has also been made in a recent paper by Pang et al. [240] where the authors independently came to the conclusion that the NuScenes tracking evaluation does indeed introduce future information into the submitted results.

To overcome this issue and measure the true on-line tracking accuracy, we propose to remove the track score averaging step in the nuScenes evaluation devkit. By applying a threshold on the current track score, the updated evaluation code computes an Average MOTA score which better reflects the real-world performance of an on-line tracker. Therefore, we repeated the experiments on the nuScenes validation set

without applying an average on the track scores and obtained the results shown on the right graph in Figure 6.29. Without the disadvantage introduced by the track score averaging, the proposed tracker significantly outperforms both Centertrack (+4.2%) and CBMOT (+6.1%) trackers in terms of AMOTA scores, using the same set of fused camera-lidar detections at input. Moreover, in an on-line evaluation setting, the proposed tracker performs much better than the state of the art CBMOT for the vulnerable road user classes "bicycle" (+19.9%), "motorcycle" (+15.9%) and "pedestrian" (+6.6%), while achieving similar performance for the vehicle classes.

In our opinion, these results are more representative of the performance of the tracker when deployed in a real-world application than what is reported by the original NuScenes devkit. The qualitative accuracy of the proposed tracker is visualized by the 3-D rendering of the NuScenes traffic scene "0061", shown in Figure 6.30. In this figure we can see the track hypotheses of various tracked objects (trucks, cars, pedestrians and bicycles) overlaid on the raw 3-D lidar point cloud. Each tracked object is depicted as a 3-D bounding box with a unique color, motion vector as well as its trajectory from the past 2 seconds.

Finally, we performed evaluation of the complete detection and tracking system in a deployment setting i.e., running the detectors at a reasonable, high-precision operating point and measuring the on-line tracking accuracy. The detection operating point was chosen in such a way that the evidence which the sensors communicate to the tracker is at 80% precision, meaning that on average there is one false positive for each 4 true detections. In order to achieve this we set different detection thresholds for the different object classes detected by FCOS3D and Centerpoint. These thresholds were optimized over an independent validation sub-set. Furthermore, we apply both our detector-detector and tracker-detector feedback loops and evaluate their effect on tracked objects. We compare our results by feeding the same detections to two state-of-the-art multi-object trackers, CenterTrack and CBMOT.

In order to not overload this final analysis, we show a summary of the most important metrics in Table 6.3. Particularly, we focus on the average MOTA, the average localization error (MOTP) and the Best Possible Recall (BPR). Using the same set of detections in a deployment setting, the proposed method outperforms CBMOT and CenterTrack by 10% and 11% in terms of AMOTA, 13% and 15% in terms of AMOTP and 12% and 13% in terms of BPR respectively. Moreover, when we activate the cooperative lidar-to-camera (L$\rightarrow$C) and tracker-to-detector (T$\rightarrow$L$\rightarrow$C) feedback loops, the proposed tracker outperforms the state-of-the art CBMOT and CenterTrack by 13% and 15% in terms of AMOTA, 24% and 27% in terms of AMOTP and 14% and 15% in terms of BPR respectively.

From these experiments it becomes clear that in a deployment setting, the proposed tracker based on switching observation models, a behavioral motion model, and dual update cycles significantly outperforms the state of the art in the most relevant tracking metrics. Moreover, the additional cooperative feedback loops (detector-detector and tracker-detector) further improve tracking by recalling more confident detections at the same false alarm rate. These feedback loops require no re-training, and at worst, achieve the accuracy of late fusion.

Method	AMOTA↑	AMOTP↓ [m]	BPR↑
CenterTrack (late fusion)	0.554	0.965	0.606
CBMOT (late fusion)	0.562	0.948	0.610
Proposed (late fusion)	0.619	0.838	0.684
Proposed (cooperative*)	0.633	0.780	0.685
Proposed (cooperative**)	**0.637**	**0.760**	**0.698**

Table 6.3: Summary of the tracking performance compared to the state of the art on NuScenes. Each tracker is fed by a reduced sub-set of ground truth detections. The proposed cooperative* method applies detector-detector feedback, while the proposed cooperative** applies an additional tracker-detector feedback.

6.10 Conclusion and practical implications

This chapter proposes a theoretical framework as well as guidelines for the practical implementation of a novel multi-object tracker based on multiple, imperfect, detectors. The work outlined in this chapter was published as two research articles in the journals MDPI Sensors 2019 (volume 19, issue 2) [7] and MDPI Sensors 2020 (volume 20, issue 17) [8], and as articles in the proceedings of the IEEE Intelligent Transportation Systems Conference 2019 [14], the IEEE Symposium on Communications and Vehicular Technology 2019 [15] and the IEEE SENSORS 2022 conference [18].

The tracking output is a list of confidently tracked object hypotheses which can be communicated to other autonomous vehicle sub-systems such as collision avoidance or path planning. The proposed tracker has multiple benefits over classical MOT: first, a decision about the existence and the location of each object is made on-line, achieving real-time operation. Second, the state variables are modeled using non-parametric probability distributions that have much higher parameter freedom allowing them to fit to ambiguous tracking situations. Third, the tracker adapts its sensor model parameters given the observational data making it robust to transient changes of sensor characteristics. Fourth, the proposed dual-update cycle makes use of an efficient representation of sub-threshold observational evidence which can reconstruct missing detections better than similar approaches based on imputations. Finally, through the use of a tracker-to-detector feedback loop, the system becomes aware of the regions with compromised seeing and recovers more detections without introducing false alarms. The combination of the above mentioned design choices allows the system to effectively operate in cases of localized loss of detection as well as in the complete failure of individual sensors.

The main contribution, made to the adaptive observation model and dual update tracking logic, facilitates better handling of missing detections which are often encountered in real-world applications. Tracking is performed on the ground plane, agnostic to the sensor configuration, where we assume that individual sensors provide detections that are missing at random. Tracking-by-detection is performed for tracks which are well supported by observational evidence, and tracking-before-detection is

performed for tracks with missing detections using a 2-D grid of sub-threshold observational data. This novel sensor model can update PF weights at a considerably lower algorithmic complexity over multiple-imputations particle filter. Simulated, as well as real-world experiments show that this method has clear advantages over optimal trackers such as KF, PF as well as MI-PF. The output of the proposed tracker remains invariant to missing observations even when as many as 50% of the detections are missing at random.

Experiments on data captured in real driving scenarios using various combinations of cameras, radar and lidar support the findings from the synthetic tests. Using two datasets developed in-house, we were able to do fine-grained experiments which measure the benefits of the novel components under various traffic and weather conditions. At the time of submission, the proposed tracker achieved state-of-the-art tracking performance on the KITTI tracking benchmark for tracking pedestrians in a cooperative camera-lidar setup. Furthermore, the same tracker was tested on the much larger nuScenes tracking benchmark where it performed competitively with state-of-the-art trackers such as Centertrack and CBMOT.

How this tracking accuracy connects to metrics that are more pertinent to autonomous driving, however, remains to be seen. For instance, the track initialization latency of a nearby object with potential for collision is more significant than the positional precision of a tracked object in the distance, i.e., one that offers no threat of collision. Collision risk is not taken into consideration by perception criteria, which are now the gold standard for evaluating tracking performance. Even though the motivation and goal is improved safety, the lack of accurate safety benchmarks causes authors in the literature to optimize their methods on potentially wrong aspects of road user tracking.

Future work

Complexity considerations

Multi-object tracking remains to be a complex topic where multiple algorithms need to interact in unison for optimal performance. Our experience has so far shown that the tracking system gains the most from having better object detection at the input. In our controlled experiments we have observed that even simple tracking algorithms can achieve good enough results when fed with excellent detection input. Contrary to this, even complicated off-line tracking methods will have a hard time connecting cluttered or missing detections. Thus, when designing a detector-tracker system it is important to consider the instantaneous detection accuracy of the detectors, the temporal accuracy of the tracking algorithm as well as the computational load of the complete system in unison.

Context modeling of likelihoods

In situations when the camera detectors are operating under sub-optimal viewing conditions (night, adverse weather, glare, etc.) the interpretation of detection scores

is different from the nominal one. Thus far, we relied on a single set of likelihood functions modeling object existence which are optimized over the complete training set containing frames captured in daytime, nighttime and adverse weather. There is an obvious potential for improvement if the system can switch to a likelihood function more appropriate to the detection scores in the specific type of scene, the same way as it does for the positional accuracy. For example, if the average illumination of the frame drops below a certain level, the fusion content dependent likelihood function which factors in the changes in precision and accuracy.

Overlapping camera views

When the fields of view of multiple *same-modality* sensors overlap, it is possible for the same object to be detected in multiple views. In this situation there is a potential for improvement because a single object detected, for example, by two cameras increases the joint camera-camera detection likelihood even before fusion with ranging sensors. However, larger objects entering or leaving the scene will be detected only partially which usually results in lower detection scores. One can see that the problem of camera-camera fusion in overlapping fields of view is not trivial and thus, for this research we refrained from a detailed modeling.

Tracking on low-level features

Beyond using adaptive detection thresholds to recover missing detections, it is also possible to track existing hypotheses without detection, using low level features extracted from the sensor measurements. For example, when a lidar point cloud gets slightly distorted around an object and the detector produces a false negative, we can still track using the blob of points knowing that this is the object of interest even though we cannot accurately classify it. The same principle can also be applied for the camera detector, for example, when an image region loses contrast due to glare an object detector can produce a false negative, but we can still track the motion of the content using optical flow. This thesis demonstrates a computationally efficient approach to tracking before detection, however, the proposed method is not a definitive approach and further research is needed to study an optimal solution. A typical fail scenario happens with occlusion, where low-level features such as image motion vectors contain incorrect information and can lead to deteriorated tracking. Care needs to be taken when tracking before detection as to not update an already good track with the incorrect low-level information.

End-to-end tracking by deep learning

End-to-end fusion has recently advanced, achieving precision that surpasses that of traditional techniques by combining simultaneous detection and tracking. Such end-to-end methods, however, have a serious problem in that they assume the continuous flow of data from all sensors. This thesis has demonstrated that this assumption

is untrue in practical situations. Sensing failures lead to catastrophic deterioration in accuracy of early fusion models because of the problem of domain shift. A potential improvement to the robustness in these cases would be to model sensing failure in the training process. A solution as simple as data augmentation using random sensor degradation might already prove effective, but further research into the detection robustness is needed.

Beyond these possible research paths, it is clear that we need detection and tracking models with better representations of uncertainty. New datasets and measures are needed to evaluate the impact of perceptual accuracy and uncertainty on driving safety. Academics and business partners are currently debating the topic of understandable perception, but no standard has yet been developed. Since this thesis only covers the perception chain, it is difficult to predict with certainty how the recommended techniques might influence driving safety. However, the method is based on well founded, Bayesian theory with demonstrated improvements over multiple benchmarks, which gives hope that it will inspire new research and lead to safe autonomous driving in the near future.

7 Overall conclusion and outlook

7.1 Conclusions

Better sensors and computers are one of the primary components that will enable completely autonomous vehicles thanks to technological advancements spurred by the consumer electronics market's unparalleled demand. The market for intelligent automobiles is expanding at the time this thesis is being written, with new models being able to better understand their surroundings and carry out automatic driving assistance functions. These breakthroughs speed up the development of lidar and radar technologies by lowering their price and expanding their availability to original equipment manufacturers in the automotive industry. The other big enabling factor for autonomous vehicles is the increase in efficient, low-cost computing power. Some of the current GPUs and neural computing chips are specifically designed for deep learning and computer vision tasks. Such devices make it possible for the complex detection and tracking algorithms to run in real-time, drawing minimal power from the batteries.

We are fortunate to have at our disposal outstanding computer vision technologies based on single sensor data. Deep learning-based algorithms for feature extraction, segmentation, lane detection, depth prediction and completion, de-noising, object detection, etc. have undergone extensive testing and documentation in order to serve as the foundation of larger systems. Optimal data fusion concepts are also available, and their efficacy under ideal sensing circumstances is well established. Multi-sensor object tracking utilizing Bayesian theory has historically produced respectable and clear findings by relying on numerous such algorithms working together harmoniously. But under real-world hardware limitations, tracking utilizing fused data is still a topic of

current research.

Applied sensor fusion appeared to be lagging behind sensor and computing technology when this study first began. There was an obvious need for academic research that can offer methods for optimal information fusion using these new and better sensors under realistic situations, as the majority of the object tracking literature was established in the latter part of the 20^{th} century. Using data fusion from cutting-edge sensors, this thesis focuses on ego-localization, object identification, and tracking as elements of the environmental perception problem. Robustness, or the management of sensor failures where existing state-of-the-art solutions appear to underperform, was and still is a specific focus of attention.

In Chapter 2 we proposed the use of a Bayesian framework for the tracking of object existences and locations on the ground plane from a moving vehicle. It offers the fundamental probabilistic concepts for formulating hypotheses and assessing their plausibility in light of data from sensors. The framework uses multi-sensor fusion to identify and track other road users in global coordinates while computing vehicle odometry. We suggested to loosely couple the location of an object with the idea that it exists; in other words, we presupposed that the belief in an object's existence is constant over relatively limited areas and irrespective of the precise position. By reducing the state space's size, we were able to quickly test hypotheses using separate location and existence likelihoods.

Our framework bases its design decisions on the reality that the system will be used in a real-world traffic environment with a variety of road users, faulty sensors, and fluctuating weather conditions. Additionally, there is a strong chance that the system may be subject to cyber attacks, vandalism, or other inexplicable occurrences that could damage a sensor. These practical issues, which are hardly covered in the literature, are the subject of a sizable section of the study in this thesis. The suggested architecture offers an accurate object tracking system that is also resistant to sensor failures and ambiguity.

In Chapter 3 a novel ego-localization method was proposed based on the principle of registration of local occupancy maps computed by the on-board sensors of the vehicle. By assuming a locally flat world and the 2-D occupancy grid model, the proposed method estimates the 3-DOF vehicle odometry (X-Y translation and yaw) by computing the apparent shift and rotation of consecutive occupancy maps. We use the fast and robust image registration method Phase-only Correlation which decouples the estimation of rotation from the estimation of translation in two independent steps. The resulting odometry is accurate and highly robust to noise, and the method has a low algorithmic complexity, a highly desirable combination of characteristics which makes it applicable for real-world tasks.

In Chapter 4 we discussed methods for reconstruction of high-resolution range information which, combined with camera pixel data, provides accurate ranging of objects detected in the image. We were mainly interested in the completion of sparse depth images that are generated by projecting lidar point cloud onto a camera image. We proposed multiple methods for depth completion that strike a different balance between their reconstruction accuracy and their computational complexity.

Experimental results showed that the depth completion method based on semantically-aware multi-lateral filter provides acceptable reconstruction accuracy at a very low computational cost. However, a significant part of the research was also spent in exploring more accurate methods based on convolutional neural networks and deep learning. The proposed CNN-based depth completion methods provided excellent reconstructions which at the tame of writing, have proved to be competitive with the state of the art on the KITTI depth completion benchmark.

In Chapter 5 we introduced the concept of cooperative sensor fusion for the purpose of instantaneous detection of road users. This concept is built on the late fusion principle where each sensor applies their own individual object detection algorithm. We proposed the addition of a low-bandwidth feedback line which carries sensor-agnostic detection information of high confidence. Individual sensors can tap into this feedback line to get cross-sensor prior about the existence of objects across their detection field and adjust the detection thresholds locally. The resulting fusion improves both object detection on the ground plane as well as in the image.

Results from real-world experiments confirmed the benefit of our cooperative fusion method in terms of better recall, lower false positives, and higher positional accuracy. Our merged detections perform much better than single-sensor detection and better than those produced via conventional late fusion. We found that the cooperative fusion works best when one or more sensors have distinct advantages over the others in terms of sensing. The differences were especially pronounced in low-light situations when the camera can recover lost information thanks to the luminance invariant sensors (radar and lidar).

In Chapter 6 we discussed several multi-sensor, multi-object tracking methods that are suitable for application in a real-world autonomous vehicle perception system. Operating under various traffic and weather conditions, perception in autonomous vehicles calls for robust methods, but at the same time limited to low computational complexity to enable real-time operation. We proposed a multi-object tracker based on particle filters with switching observational models that cope with changing sensor characteristics, and a dual update cycle that can cope with complete loss of observations. The existence of each track is modeled by a Bayes filter with a static state, while the motion of objects is modeled by a novel behavioral model that is learned off-line.

Through simulation, we were able to demonstrate the distinct advantages each component (such as changing observation models, behavioral motion models, particle sample sizes, etc.) has over alternative approaches. We put the entire perception system to the test on four distinct multi-object tracking datasets in order to demonstrate how well it performs in real-world situations. These datasets, which reflect a wide range of traffic and meteorological situations, were collected across numerous urban regions on three continents. The suggested technique demonstrated state-of-the-art tracking performance in terms of numerous metrics assessing object recall, clutter suppression, temporal consistency, as well as response time.

7.2 Valorisation

The interdisciplinary nature of the topics covered in this PhD allowed us to transform the findings into several sustainable products and solutions that are making a measurable benefit to society. These products are continuing to create value from the gained knowledge, improving economic prosperity and positively impacting the environment. Parts of the methods explained in this dissertation have been incorporated into the practical implementations within the five research projects listed below.

- **Environmental Model based Driver Assistance (EMDAS)**, project-140647 funded by the Flemish Agency for Innovation by Science and Technology (IWT); Aug 2015 - Jul 2017

The main objective of this research project was to demonstrate an Autonomous Guided Transport System for public environments i.e., a driverless shuttle bus carrying personnel on private and public roads within the Brussels Airport. Multiple shared objectives were set and achieved by a consortium of the following companies and research institutes: Flanders DRIVE, Ghent University, VDL Groep, LMS, TML, TomTom and Xenics. The involvement from Ghent University was mainly focused on the creation of dynamic 2-D risk maps consisting of positions and intentions of static and moving objects which determine the risk for the vehicle. This is reflected in weights for driving speeds and possible locations (indicated that the speed has to be turned low) and from which positions additional observations have to be done before proceeding.

- **Avoidance of collisions and obstacles in narrow lanes (AVCON)**, an ICON-project HBC/2017.0390 funded by the Flemish Agency for Innovation by Science and Technology (IWT); Jan 2018 - Jul 2018

The AVCON project proposed a flexible path planning and tracking solution, aiming to be applicable to several application domains, and which is able to dynamically avoid an (unforeseen) obstacle by an overtake maneuver. These algorithms are implemented on an autonomous forklift and the overall system performance is measured by the time-of-arrival of an obstacle avoidance maneuver. The autonomous forklift used an array of camera and lidar sensors and the main research contribution to the project was the modeling of the camera and lidar observation models described in Section 6.6.

- **Cooperative radar-video sensing for urban environments**, an IMEC B-Project 25-DS (internally funded); Aug 2018 - Jan 2020

The scope of this research was to create a perception system for object tracking that combines sensor fusion at the object detection level with radar, lidar, RGB, and/or infrared cameras. This entails identifying false information and discarding it, as well as determining more precise estimates of the locations and motions of objects. Real-time, low-latency detection of vehicles, people on foot, and bicycles in congested metropolitan areas under varying weather and illumination conditions was the main focus. The project's goal is to show how sensor output and cooperative fusion for radar and camera systems might be advantageous.

- **HiSilicon – IMEC cooperation on automotive sensor fusion**, a research project funded by HiSilicon; Apr 2020 - Apr 2022

The objective of this cooperation was to develop an object tracking perception system that fuses radar, lidar, RGB, and/or infrared cameras with sensor fusion at the object detection level. This involves making more accurate predictions of the locations and motions of objects as well as recognizing incorrect information and eliminating it. The main goal was real-time, low-latency identification of automobiles, persons on foot, and bicycles in congested urban settings under a variety of weather and lighting circumstances. The project's objective is to demonstrate the potential benefits of sensor output and cooperative fusion for radar and camera systems.

- **AI Flanders Research Program** - Project funded by the Ministry of Economy, Science and Innovation in Flanders; July 2019 - June 2022.

The work outlined in this dissertation was used partly in the Real-time and power-efficient AI in the edge challenge of the project (WP2 Inter Device Algorithms and WP3 Intra-Device Algorithms). The research leads to application-oriented cases far ahead of the state of the art for distributed and hierarchical AI systems, advanced signal processing, and learning algorithms for extracting actionable information from the edge. Specifically, our involvement was in the development of a cooperative muti-node tracking method where we demonstrated the benefits of cooperative fusion of camera and radar detectors. Furthermore, demonstrating the sensor-to-sensor handover and cooperative adjustment of low-level edge AI/vision parameters. In the intra-device research one of the milestones is the selective processing in early radar/video/thermal fusion and deep learning restricted to regions of interest defined by local analysis of learned scene entry points.

- **Future APT Living Lab for Autonomous Public Transport (LivingLAPT)** - Project-22332, funded by the European Institute of Innovation & Technology (EIT) Regional Innovation Scheme (RIS); January 2022 - December 2023.

This project aims to develop and continuously examine an integrated approach to a trustworthy, safe and inclusive transit system involving autonomous shuttles operating in shared environments alongside other users including cyclists, pedestrians and motorized vehicles in PEARL (Person Environment Activity Research Laboratory) and Cities living labs. The project is a collaboration effort between the city of Helmond (Netherlands), city of Prague (Czech republic), Eindhoven University of Technology, University College London, PowerHUB, Ghent University, City of Hasselt, Municipality of Ricany, Future Mobility Network, Staf Cars, Applied Autonomy and Kongsberg Municipality. The involvement in this project was in providing existing software building blocks for 3D mapping and for safety assessment around autonomous vehicles into reporting tools for assessing driving safety and environmental changes. We captured onboard sensor data and evaluated the performance of the shuttle in field trials. Readiness assessment for successful implementation of an autonomous bus shuttle from a safety and governance perspective was carried out through assessment of local governance.

- **SafeNav** - Project-101077026, funded by the European Climate, Infrastructure and Environment Executive Agency (CINEA); October 2022 - October 2025.

SafeNav's ambition is to develop a digital solution to autonomously detect vessels, cetaceans, containers, and other submerged and semi-submerged objects in the marine environment in real time, employing data collection and fusion from various state-of-the-art sensors, met-ocean data and other relevant information sources, thus aiding the navigators with all-in-one place, easy to process information and visualization required for quick decision making based on the Convention of International Regulations for Preventing Collisions at Sea.

7.3 Outlook

The environmental perception research for autonomous vehicles seems to be heading in the direction of multi-sensor fusion methods based on deep learning. As of 2022, most novel methods try to solve the object detection and tracking in an end-to-end manner by early fusion in a joint multi-sensor feature space. However, the performance of deep learning methods remains unpredictable in border cases, which makes such methods undesirable for safety-critical applications. This thesis provides an alternative approach where modules of deep-learning-based computer vision algorithms are integrated into a classical Bayesian framework. The fail cases of the proposed fusion system are well understood and the amount of drop in performance is bounded. As we saw in Chapter 6, modeling the scene context can bring significant benefits over generalized models. However, modeling context variables can become cumbersome, especially in situations with multiple interacting factors (for example, nighttime and fog and glare, etc.) where we do not exactly know the causal direction. An obvious avenue of future research is to delegate the learning of context for Bayesian fusion methods into a neural network that can better learn the causality from labeled data.

Furthermore, despite the measurable improvements over the state of the art in several perception fields, this thesis does not claim to have fully solved the autonomous vehicle perception problem. On the theoretical aspect, we need to put more effort into proving the convergence of trackers under the constraints of missing detections and sensor failures. Moreover, the effectiveness of the proposed sensor-sensor cooperative feedback was only shown experimentally. It is essential that we find the theoretical performance bounds of these algorithms.

We encountered a problem with test standardization in some experiments performed for this thesis. We are compelled to employ metrics that may or may not be related to real-world performance in order to compare to the literature. Additionally, approaches are often developed and evaluated on the same dataset, therefore the performance across datasets is unknown. Finally, rather than computing dataset average performance, we need to focus our efforts on building granular experiments, testing on data splits across varied traffic and weather situations. Such enhanced experimental techniques will provide information on particular issues that are not visible with the current evaluation.

On the hardware side, special consideration should be paid to the actual sensors on which the algorithms are implemented. This implies that the models must be trained using a broader variety of sensors, including variants of the same sensor modality. For example, running camera detectors on low-end cameras or compressed video streams, or lidar detectors on scans from low-end lidar. We can be certain that, for the foreseeable future, the perception algorithms will not perform best across sensors from various automobile manufacturers due to differences in hardware design. There is presently no minimal standard for camera sensors, lenses, radars, and lidars that will allow level-5 autonomous driving.

Greater photo-realism in the depiction of simulated surroundings is made possible

by the same GPU and neural processing technology that allows for the fast execution of perception algorithms. As of 2023, we need to pay particular attention to the advancement of simulation software for autonomous driving (such as NVIDIA DRIVE Sim and Siemens PreScan). Researchers may test their algorithms in-the-loop using these simulators to build realism-based artificial worlds with accurate physics and sensor models. Additionally, we may create traffic conditions using the simulation program that are hard to replicate in the actual world and act as performance standards in crucial circumstances.

Finally, while performing the study for this thesis, we encountered several legislative challenges that made it difficult to collect data and made it illegal to test autonomous robots on public roadways. Improved collaboration between academic institutions and governmental regulatory agencies appears to be required in order to relax regulations for prototype testing and data sharing on campus. Even though this way of thinking may unintentionally result in more errors, errors are a necessary part of learning, and the sooner we make them, the sooner we can remedy them.

Bibliography

[1] StatBel. *Road traffic accidents with personal injury in Belgium which resulted in a police report.*, 2021.

[2] Mobility European Comission and Transport. *Road safety in the EU*, 2021.

[3] Regional office for Europe World Health Organization. *European status report on road safety. Towards safer roads and healthier transport choices*, 2009.

[4] C Kpmg, G Silberg, R Wallace, G Matuszak, J Plessers, C Brower, and D Subramanian. *Self-driving cars: The next revolution*. Kpmg: Seattle, WA, USA, 2012.

[5] Daniel J Fagnant and Kara Kockelman. *Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations*. Transportation Research Part A: Policy and Practice, 77:167–181, 2015.

[6] SAE International Society of Automotive Engineers. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicle*, 2018.

[7] Martin Dimitrievski, Peter Veelaert, and Wilfried Philips. *Behavioral Pedestrian Tracking Using a Camera and LiDAR Sensors on a Moving Vehicle*. Sensors, 19(2), 2019.

[8] Martin Dimitrievski, David Van Hamme, Peter Veelaert, and Wilfried Philips. *Cooperative Multi-Sensor Tracking of Vulnerable Road Users in the Presence of Missing Detections*. Sensors, 20(17), 2020.

[9] Martin Dimitrievski, David Van Hamme, Peter Veelaert, and Wilfried Philips. *Robust Matching of Occupancy Maps for Odometry in Autonomous Vehicles*. In Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3: VISAPP, (VISIGRAPP 2016), pages 626–633. INSTICC, SciTePress, 2016.

[10] M. Dimitrievski, P. Veelaert, and W. Philips. *Semantically aware multilateral filter for depth upsampling in automotive LiDAR point clouds*. In 2017 IEEE Intelligent Vehicles Symposium (IV), pages 1058–1063, June 2017.

[11] Martin Dimitrievski, Bart Goossens, Peter Veelaert, and Wilfried Philips. *High resolution depth reconstruction from monocular images and sparse point clouds using deep convolutional neural network*. In Unconventional and Indirect Imaging, Image Reconstruction, and Wavefront Sensing 2017, volume 10410, page 104100H. International Society for Optics and Photonics, 2017.

[12] M. Dimitrievski, P. Veelaert, and W. Philips. *Information feedback loop for improved pedestrian detection in an autonomous perception system*. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pages 3119–3124, 2018.

[13] Martin Dimitrievski, Peter Veelaert, and Wilfried Philips. *Learning Morphological Operators for Depth Completion: 19th International Conference, ACIVS 2018, Poitiers, France, September 24-27, 2018, Proceedings*, pages 450–461. 09 2018.

[14] M. Dimitrievski, L. Jacobs, P. Veelaert, and W. Philips. *People Tracking by Cooperative Fusion of RADAR and Camera Sensors*. In 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pages 509–514, 2019.

[15] Martin Dimitrievski, David Van Hamme, Lennert Jacobs, Peter Veelaert, Heidi Steendam, and Wilfried Philips. *Tracking road users by cooperative fusion of radar and camera sensors*. In 26th Symposium on Communications and Vehicular Technology in the Benelux (SCVT 2019), Abstracts, page 2, 2019.

[16] Martin Dimitrievski, Ivana Shopovska, David Van Hamme, Peter Veelaert, and Wilfried Philips. *Weakly supervised deep learning method for vulnerable road user detection in FMCW radar*. In 23rd IEEE International Conference on Intelligent Transportation Systems, 2020.

[17] Martin Dimitrievski, Ivana Shopovska, David Van Hamme, Peter Veelaert, and Wilfried Philips. *Automatic labeling of vulnerable road users in multi-sensor data*. In 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), pages 2623–2630, 2021.

[18] Dimitrievski, Martin and Van Hamme, David and Philips, Wilfried. *Perception system based on cooperative fusion of lidar and cameras*. In IEEE SENSORS 2022, Proceedings, page 4, 2022.

[19] Hong Cheng. *Autonomous Intelligent Vehicles*. Springer-Verlag London Limited, 2011.

[20] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. *Vision meets Robotics: The KITTI Dataset*. International Journal of Robotics Research (IJRR), 2013.

[21] Sebastian Schuon, Christian Theobalt, James Davis, and Sebastian Thrun. *High-quality scanning using time-of-flight depth superresolution*. pages 1 – 7, 07 2008.

[22] Sebastian Thrun. *Learning Occupancy Grid Maps with Forward Sensor Models*. Autonomous Robots, 15(2):111–127, Sep 2003.

[23] Joseph Redmon and Ali Farhadi. *YOLO9000: Better, Faster, Stronger*. CoRR, abs/1612.08242, 2016.

[24] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. CoRR, abs/1804.02767, 2018.

[25] Ronald P. S. Mahler. *Statistical Multisource-Multitarget Information Fusion*. Artech House, Inc., USA, 2007.

[26] Ba-Ngu Vo, Ba-Tuong Vo, and Dinh Phung. *Labeled Random Finite Sets and the Bayes Multi-Target Tracking Filter*. IEEE Transactions on Signal Processing, 62(24):6554–6567, 2014.

[27] Zia Khan, T. Balch, and F. Dellaert. *MCMC-based particle filtering for tracking a variable number of interacting targets*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(11):1805–1819, 2005.

[28] Harold W. Kuhn. *The Hungarian Method for the assignment problem*. Naval Research Logistics Quarterly, 2:83–97, 1955.

[29] Van Hamme, David. *Robust ego-localization using monocular visual odometry*. PhD thesis, Ghent University, 2016.

[30] Vlaminck, Michiel. *Mobile 3D mapping and localization using active depth sensors*. PhD thesis, Ghent University, 2020.

[31] Eric Wood, Adam Duran, Evan Burton, Jeffrey Gonder, and Kenneth Kelly. *EPA GHG Certification of Medium- and Heavy-Duty Vehicles: Development of Road Grade Profiles Representative of US Controlled Access Highways*. 5 2015.

[32] A. Geiger, P. Lenz, and R. Urtasun. *Are we ready for autonomous driving? The KITTI vision benchmark suite*. In 2012 IEEE Conference on Computer Vision and Pattern Recognition, pages 3354–3361, June 2012.

[33] FranÃ§ois Pomerleau, Francis Colas, Roland Siegwart, and StÃ©phane Magnenat. *Comparing ICP variants on real-world data sets*. Autonomous Robots, 04 2013.

[34] F. Moosmann and C. Stiller. *Velodyne SLAM*. In 2011 IEEE Intelligent Vehicles Symposium (IV), pages 393–398, June 2011.

[35] Ji Zhang and Sanjiv Singh. *LOAM: Lidar Odometry and Mapping in Real-time*. In Robotics: Science and Systems Conference, Pittsburgh, PA, July 2014.

[36] H. Javan Hemmat, E. Bondarev, G. Dubbelman, and P.H.N. With, de. *Improved ICP-based pose estimation by distance-aware 3D mapping*. In 9th International Conference on Computer Vision and Theory (VISAPP 2014), January 5-8, 2014, Lisbon, Portugal, volume 3, pages 360–367, 2014. conference; VISAPP 2014 ; Conference date: 01-01-2014.

[37] Sebastian Scherer, Joern Rehder, Supreeth Achar, Hugh Cover, Andrew Chambers, Stephen Nuske, and Sanjiv Singh. *River Mapping from a Flying Robot: State Estimation, River Detection, and Obstacle Mapping*. Autonomous Robots, 33(1-2):189–214, aug 2012.

[38] H. Moravec and A. Elfes. *High resolution maps from wide angle sonar*. In Proceedings. 1985 IEEE International Conference on Robotics and Automation, volume 2, pages 116–121, Mar 1985.

[39] David Kortenkamp, R. Peter Bonasso, and Robin Murphy, editors. *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*. MIT Press, Cambridge, MA, USA, 1998.

[40] F. Homm, N. Kaempchen, J. Ota, and D. Burschka. *Efficient occupancy grid computation on the GPU with lidar and radar for road boundary detection*. In 2010 IEEE Intelligent Vehicles Symposium, pages 1006–1013, June 2010.

[41] Yang Chen and GÃ©rard Medioni. *Object Modeling by Registration of Multiple Range Images*. Image Vision Comput., 10:145–155, 01 1992.

[42] Martin A. Fischler and Robert C. Bolles. *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*. Commun. ACM, 24(6):381–395, June 1981.

[43] Sei Nagashima, Koichi Ito, Takafumi Aoki, Hideaki Ishii, and Koji Kobayashi. *A High-Accuracy Rotation Estimation Algorithm Based on 1D Phase-Only Correlation*, pages 210–221. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[44] Hassan Foroosh, Josiane Zerubia, and M. Berthod. *Extension of phase correlation to sub-pixel registration*. IEEE Transactions on Image Processing, 11:188–200, 01 2002.

[45] Kuglin C. D. and Hines D. C. *The phase correlation image alignment method*. In Proceedings of the International Conference on Cybernetics and Society, San Francisco, CA, USA, pages 163–165, 1975.

[46] Bart Goossens. *Dataflow management, dynamic load balancing, and concurrent processing for real-time embedded vision applications using Quasar*. INTERNATIONAL JOURNAL OF CIRCUIT THEORY AND APPLICATIONS, 46(9):1733–1755, 2018.

[47] Jonas De Vylder and Bart Goossens. *Quasar: A Programming Framework for Rapid Prototyping*. page 1. NVIDIA, 2016.

[48] David Van Hamme, Werner Goeman, Peter Veelaert, and Wilfried Philips. *Robust monocular visual odometry for road vehicles using uncertain perspective projection*. EURASIP JOURNAL ON IMAGE AND VIDEO PROCESSING, pages 10:1–10:18, 2015.

[49] Michiel Vlaminck, Hiep Luong, and Wilfried Philips. *Have I seen this place before? A fast and robust loop detection and correction method for 3D Lidar SLAM*. SENSORS, 19(1), 2019.

[50] John Lekner and Michael C. Dorf. *Why some things are darker when wet*. Appl. Opt., 27(7):1278–1280, Apr 1988.

[51] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. *Center-based 3D Object Detection and Tracking*. CVPR, 2021.

[52] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. *FCOS3D: Fully Convolutional One-Stage Monocular 3D Object Detection*. 2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW), pages 913–922, 2021.

[53] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, USA, 2 edition, 2003.

[54] Szeliski R. *Structure from motion. In: Computer Vision. Texts in Computer Science.* Springer, 2011.

[55] S. Suwajanakorn, C. Hernandez, and S. M. Seitz. *Depth from focus with your mobile phone*. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3497–3506, 2015.

[56] Ruo Zhang, Ping-Sing Tsai, J. E. Cryer, and M. Shah. *Shape-from-shading: a survey*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(8):690–706, 1999.

[57] Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng. *Learning Depth from Single Monocular Images*. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, Advances in Neural Information Processing Systems 18, pages 1161–1168. MIT Press, 2006.

[58] Beyang Liu, Stephen Gould, and Daphne Koller. *Single Image Depth Estimation From Predicted Semantic Labels*. pages 1253–1260, 06 2010.

[59] David Eigen, Christian Puhrsch, and Rob Fergus. *Depth Map Prediction from a Single Image Using a Multi-Scale Deep Network*. In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS'14, pages 2366–2374, Cambridge, MA, USA, 2014. MIT Press.

[60] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. *Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields*. IEEE Trans. Pattern Anal. Mach. Intell., 38(10):2024–2039, October 2016.

[61] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. *Deeper Depth Prediction with Fully Convolutional Residual Networks*. 10 2016.

[62] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. CoRR, abs/1512.03385, 2015.

[63] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. CoRR, abs/1505.04597, 2015.

[64] Jennifer Dolson, Jongmin Baek, Christian Plagemann, and Sebastian Thrun. *Upsampling Range Data in Dynamic Environments*. pages 1141–1148, 06 2010.

[65] C. Premebida, J. Carreira, J. Batista, and U. Nunes. *Pedestrian detection combining RGB and dense LIDAR data*. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 4112–4117, 2014.

[66] Cristiano Premebida, Luis Garrote, Alireza Asvadi, A. Pedro Ribeiro, and Urbano Nunes. *High-resolution LIDAR-based Depth Mapping using Bilateral Filter*. CoRR, abs/1606.05614, 2016.

[67] Jason Ku, Ali Harakeh, and Steven Lake Waslander. *In Defense of Classical Image Processing: Fast Depth Completion on the CPU*. CoRR, abs/1802.00036, 2018.

[68] Nick Schneider, Lukas Schneider, Peter Pinggera, Uwe Franke, Marc Pollefeys, and Christoph Stiller. *Semantically Guided Depth Upsampling*. In Bodo Rosenhahn and Bjoern Andres, editors, Pattern Recognition, pages 37–48, Cham, 2016. Springer International Publishing.

[69] Jonathan Long, Evan Shelhamer, and Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. CoRR, abs/1411.4038, 2014.

[70] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. *Sparsity Invariant CNNs*. CoRR, abs/1708.06500, 2017.

[71] Pei An, Wenxing Fu, Yingshuo Gao, Jie Ma, Jun Zhang, Kun Yu, and Bin Fang. *Lambertian Model-Based Normal Guided Depth Completion for LiDAR-Camera System*. IEEE Geoscience and Remote Sensing Letters, 19:1–5, 2022.

[72] Yiming Zhao, Lin Bai, Ziming Zhang, and Xinming Huang. *A Surface Geometry Model for LiDAR Depth Completion*. IEEE Robotics and Automation Letters, 6(3):4457–4464, 2021.

[73] Dennis Teutscher, Patrick Mangat, and Oliver WasenmÃŒller. *PDC: Piecewise Depth Completion utilizing Superpixels*. In 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), pages 2752–2758, 2021.

[74] Shreyas S. Shivakumar, Ty Nguyen, Ian D. Miller, Steven W. Chen, Vijay Kumar, and Camillo J. Taylor. *DFuseNet: Deep Fusion of RGB and Sparse Depth Information for Image Guided Dense Depth Completion*. In 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pages 13–20, 2019.

[75] Saif Imran, Yunfei Long, Xiaoming Liu, and Daniel Morris. *Depth Coefficients for Depth Completion*. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 12438–12447, 2019.

[76] Alex Wong and Stefano Soatto. *Unsupervised Depth Completion With Calibrated Backprojection Layers*. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 12747–12756, October 2021.

[77] Maximilian Jaritz, Raoul De Charette, Emilie Wirbel, Xavier Perrotton, and Fawzi Nashashibi. *Sparse and Dense Data with CNNs: Depth Completion and Semantic Segmentation*. In 2018 International Conference on 3D Vision (3DV), pages 52–60, 2018.

[78] Kaiyue Lu, Nick Barnes, Saeed Anwar, and Liang Zheng. *From Depth What Can You See? Depth Completion via Auxiliary Image Reconstruction*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[79] Zhixin Guo, Wenzi Liao, Peter Veelaert, and Wilfried Philips. *Occlusion-robust Detector Trained with Occluded Pedestrians*. In ICPRAM, 2018.

[80] Zhixin Guo, Wenzi Liao, Yifan Xiao, Peter Veelaert, and Wilfried Philips. *Deep Learning Fusion of RGB and Depth Images for Pedestrian Detection*. In BMVC, 2019.

[81] Zhixin Guo, Wenzhi Liao, Yifan Xiao, Peter Veelaert, and Wilfried Philips. *An Occlusion-Robust Feature Selection Framework in Pedestrian Detection*. Sensors, 18(7), 2018.

[82] C. Tomasi and R. Manduchi. *Bilateral filtering for gray and color images*. In Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271), pages 839–846, Jan 1998.

[83] Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. *Fast Feature Pyramids for Object Detection*. PAMI, 2014.

[84] Doyeon Kim, Woonghyun Ga, Pyungwhan Ahn, Donggyu Joo, Sehwan Chun, and Junmo Kim. *Global-Local Path Networks for Monocular Depth Estimation with Vertical CutDepth*, 01 2022.

[85] Dominik Scherer, Andreas Müller, and Sven Behnke. *Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition*. In Proceedings of the 20th International Conference on Artificial Neural Networks: Part III, ICANN'10, pages 92–101, Berlin, Heidelberg, 2010. Springer-Verlag.

[86] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. *Multi-View 3D Object Detection Network for Autonomous Driving*. CoRR, abs/1611.07759, 2016.

[87] Jonathan Masci, Jesús Angulo, and Jürgen Schmidhuber. *A Learning Framework for Morphological Operators using Counter-Harmonic Mean*. CoRR, abs/1212.2546, 2012.

[88] L. J. Van Vliet. *Robust local max-min filters by normalized power-weighted filtering*. In Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004., volume 1, pages 696–699 Vol.1, Aug 2004.

[89] Jesús Angulo. *Pseudo-morphological Image Diffusion Using the Counter-Harmonic Paradigm*. In Jacques Blanc-Talon, Don Bone, Wilfried Philips, Dan Popescu, and Paul Scheunders, editors, Advanced Concepts for Intelligent Vision Systems, pages 426–437, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[90] A. Vedaldi and K. Lenc. *MatConvNet – Convolutional Neural Networks for MATLAB*. In Proceeding of the ACM Int. Conf. on Multimedia, 2015.

[91] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. CoRR, abs/1412.6980, 2014.

[92] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. *Make3D: Learning 3D Scene Structure from a Single Still Image*. IEEE Trans. Pattern Anal. Mach. Intell., 31(5):824–840, May 2009.

[93] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr DollÃ¡r. *Microsoft COCO: Common Objects in Context*, 2014.

[94] P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, volume 1, pages I–I, 2001.

[95] N. Dalal and B. Triggs. *Histograms of oriented gradients for human detection*. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 1, pages 886–893 vol. 1, 2005.

[96] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. *Object Detection with Discriminatively Trained Part-Based Models*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9):1627–1645, 2010.

[97] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. *Region-Based Convolutional Networks for Accurate Object Detection and Segmentation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(1):142–158, 2016.

[98] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition*. CoRR, abs/1406.4729, 2014.

[99] Ross B. Girshick. *Fast R-CNN*. CoRR, abs/1504.08083, 2015.

[100] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. CoRR, abs/1506.01497, 2015.

[101] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. *Mask R-CNN*. CoRR, abs/1703.06870, 2017.

[102] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. *Feature Pyramid Networks for Object Detection*. CoRR, abs/1612.03144, 2016.

[103] Jianfeng Wang and Xiaolin Hu. *Convolutional Neural Networks with Gated Recurrent Connections*. CoRR, abs/2106.02859, 2021.

[104] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. *SSD: Single Shot MultiBox Detector*. CoRR, abs/1512.02325, 2015.

[105] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. *Focal Loss for Dense Object Detection*. CoRR, abs/1708.02002, 2017.

[106] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C. Berg. *DSSD : Deconvolutional Single Shot Detector*. CoRR, abs/1701.06659, 2017.

[107] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. *Scale-Aware Trident Networks for Object Detection*. CoRR, abs/1901.01892, 2019.

[108] Bharat Singh, Mahyar Najibi, and Larry S. Davis. *SNIPER: Efficient Multi-Scale Training*. CoRR, abs/1805.09300, 2018.

[109] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. CoRR, abs/2004.10934, 2020.

[110] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. *You Only Learn One Representation: Unified Network for Multiple Tasks*. CoRR, abs/2105.04206, 2021.

[111] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. *Swin Transformer V2: Scaling Up Capacity and Resolution*. CoRR, abs/2111.09883, 2021.

[112] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. *SimMIM: A Simple Framework for Masked Image Modeling*. CoRR, abs/2111.09886, 2021.

[113] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*, 2016. cite arxiv:1612.00593.

[114] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. In NIPS, 2017.

[115] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. *PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud*. CoRR, abs/1812.04244, 2018.

[116] Weijing Shi and Ragunathan Rajkumar. *Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud*. CoRR, abs/2003.01251, 2020.

[117] Anshul Paigwar, Özgür Erkent, Christian Wolf, and Christian Laugier. *Attentional PointNet for 3D-Object Detection in Point Clouds*. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 1297–1306, 2019.

[118] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. *3D Object Detection with Pointformer*. CoRR, abs/2012.11409, 2020.

[119] Yin Zhou and Oncel Tuzel. *VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection*. CoRR, abs/1711.06396, 2017.

[120] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. *PointPillars: Fast Encoders for Object Detection from Point Clouds*. CoRR, abs/1812.05784, 2018.

[121] Yan Yan, Yuxing Mao, and Bo Li. *SECOND: Sparsely Embedded Convolutional Detection*. Sensors, 18(10), 2018.

[122] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. *Point-Voxel CNN for Efficient 3D Deep Learning*. CoRR, abs/1907.03739, 2019.

[123] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. *Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution*, 2020.

[124] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. *PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection*. CoRR, abs/1912.13192, 2019.

[125] Gregory P. Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K. Wellington. *LaserNet: An Efficient Probabilistic 3D Object Detector for Autonomous Driving*. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 12669–12678, 2019.

[126] Meytal Rapoport-Lavie and Dan Raviv. *It's All Around You: Range-Guided Cylindrical Network for 3D Object Detection*. CoRR, abs/2012.03121, 2020.

[127] Zhidong Liang, Zehan Zhang, Ming Zhang, Xian Zhao, and Shiliang Pu. *RangeIoUDet: Range Image based Real-Time 3D Object Detector Optimized by Intersection over Union*. In 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 7136–7145, 2021.

[128] R. Schmidt. *Multiple emitter location and signal parameter estimation*. IEEE Transactions on Antennas and Propagation, 34(3):276–280, March 1986.

[129] Louis Scharf and Cedric Demeure. *Statistical signal processing : detection, estimation, and time series analysis*. Reading, Mass. Addison-Wesley Pub. Co., 1991.

[130] S. Heuel and H. Rohling. *Pedestrian recognition in automotive radar sensors*. In 2013 14th International Radar Symposium (IRS), volume 2, pages 732–739, June 2013.

[131] O. Schumann, C. Wohler, M. Hahn, and J. Dickmann. *Comparison of random forest and long short-term memory network performances in classification tasks using radar*. In 2017 Sensor Data Fusion: Trends, Solutions, Applications (SDF), pages 1–6, Oct 2017.

[132] Xingjian SHI, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. *Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting*. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28, pages 802–810. Curran Associates, Inc., 2015.

[133] J. Lombacher, K. Laudt, M. Hahn, J. Dickmann, and C. WÃ¶hler. *Semantic radar grids*. In 2017 IEEE Intelligent Vehicles Symposium (IV), pages 1170–1175, June 2017.

[134] Daniel Brodeski, Igal Bilik, and Raja Giryes. *Deep Radar Detector*. CoRR, abs/1906.12187, 2019.

[135] Guoqiang Zhang, Haopeng Li, and Fabian Wenger. *Object Detection and 3D Estimation via an FMCW Radar Using a Fully Convolutional Network*. CoRR, abs/1902.05394, 2019.

[136] Andras Palffy, Jiaao Dong, Julian Kooij, and Dariu Gavrila. *CNN based Road User Detection using the 3D Radar Cube*. 5:1263 – 1270, 01 2020.

[137] Bence Major, Daniel Fontijne, Amin Ansari, Ravi Teja Sukhavasi, Radhika Gowaikar, Michael Hamilton, Sean Lee, Slawomir Grzechnik, and Sundar Subramanian. *Vehicle Detection With Automotive Radar Using Deep Learning on Range-Azimuth-Doppler Tensors*. In The IEEE International Conference on Computer Vision (ICCV) Workshops, Oct 2019.

[138] Charles Ruizhongtai Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. *Frustum PointNets for 3D Object Detection from RGB-D Data*. CoRR, abs/1711.08488, 2017.

[139] Zhixin Wang and Kui Jia. *Frustum ConvNet: Sliding Frustums to Aggregate Local Point-Wise Features for Amodal 3D Object Detection*. CoRR, abs/1903.01864, 2019.

[140] F. Farahnakian, M. Haghbayan, J. Poikonen, M. Laurinen, P. Nevalainen, and J. Heikkonen. *Object Detection Based on Multi-sensor Proposal Fusion in Maritime Environment*. In 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), pages 971–976, Dec 2018.

[141] Sourabh Vora, Alex H. Lang, Bassam Helou, and Oscar Beijbom. *PointPainting: Sequential Fusion for 3D Object Detection*. CoRR, abs/1911.10150, 2019.

[142] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. *Multimodal Virtual Point 3D Detection*. NeurIPS, 2021.

[143] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. *Deep Parametric Continuous Convolutional Neural Networks*. CoRR, abs/2101.06742, 2021.

[144] Vishwanath A. Sindagi, Yin Zhou, and Oncel Tuzel. *MVX-Net: Multimodal VoxelNet for 3D Object Detection*. CoRR, abs/1904.01649, 2019.

[145] Yingwei Li, Adams Wei Yu, Tianjian Meng, Ben Caine, Jiquan Ngiam, Daiyi Peng, Junyang Shen, Bo Wu, Yifeng Lu, Denny Zhou, Quoc V. Le, Alan Yuille, and Mingxing Tan. *DeepFusion: Lidar-Camera Deep Fusion for Multi-Modal 3D Object Detection*, 2022.

[146] Jin Hyeok Yoo, Yecheol Kim, Ji Song Kim, and Jun Won Choi. *3D-CVF: Generating Joint Camera and LiDAR Features Using Cross-View Spatial Feature Fusion for 3D Object Detection*. CoRR, abs/2004.12636, 2020.

[147] Xuanyao Chen, Tianyuan Zhang, Yue Wang, Yilun Wang, and Hang Zhao. *FUTR3D: A Unified Sensor Fusion Framework for 3D Detection*, 2022.

[148] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela Rus, and Song Han. *BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation*, 2022.

[149] Zehui Chen, Zhenyu Li, Shiquan Zhang, Liangji Fang, Qinghong Jiang, Feng Zhao, Bolei Zhou, and Hang Zhao. *AutoAlign: Pixel-Instance Feature Aggregation for Multi-Modal 3D Object Detection*, 2022.

[150] Yanwei Li, Xiaojuan Qi, Yukang Chen, Liwei Wang, Zeming Li, Jian Sun, and Jiaya Jia. *Voxel Field Fusion for 3D Object Detection*, 2022.

[151] Xuyang Bai, Zeyu Hu, Xinge Zhu, Qingqiu Huang, Yilun Chen, Hongbo Fu, and Chiew-Lan Tai. *TransFusion: Robust LiDAR-Camera Fusion for 3D Object Detection with Transformers*, 2022.

[152] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. *PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation*, 2017.

[153] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Lake Waslander. *Joint 3D Proposal Generation and Object Detection from View Aggregation*. CoRR, abs/1712.02294, 2017.

[154] Jian Han, Yaping Liao, Junyou Zhang, Shufeng Wang, and Sixian Li. *Target fusion detection of LiDAR and camera based on the improved YOLO algorithm*. Mathematics, 6(10):213, 2018.

[155] Su-Birm Park, Fabio Tango, Olivier Aycard, Aris Polychronopoulos, Ullrich Dr.-Ing. Scheunert, and Thomas Tatschke. *ProFusion2 - Sensor Data Fusion for Multiple Active Safety Applications*. 2006.

[156] Z. Wang, Y. Wu, and Q. Niu. *Multi-Sensor Fusion in Automated Driving: A Survey*. IEEE Access, 8:2847–2868, 2020.

[157] Uwe Knauer and Udo Seiffert. *A comparison of late fusion methods for object detection*. In 2013 IEEE International Conference on Image Processing, pages 3297–3301, 2013.

[158] Su Pang, Daniel Morris, and Hayder Radha. *CLOCs: Camera-LiDAR Object Candidates Fusion for 3D Object Detection*, 2020.

[159] Eduardo Arnold, Mehrdad Dianati, and Robert de Temple. *Cooperative Perception for 3D Object Detection in Driving Scenarios using Infrastructure Sensors*. CoRR, abs/1912.12147, 2019.

[160] Jingda Guo, Dominic Carrillo, Sihai Tang, Qi Chen, Qing Yang, Song Fu, Xi Wang, Nannan Wang, and Paparao Palacharla. *CoFF: Cooperative Spatial Feature Fusion for 3D Object Detection on Autonomous Vehicles*. CoRR, abs/2009.11975, 2020.

[161] S. Nadimi and Bir Bhanu. *Physics-Based Cooperative Sensor Fusion for Moving Object Detection*. pages 108–108, 01 2004.

[162] S. Yenkanchi and Q. M. J. Wu. *Cooperative fusion for road obstacles detection using laser scanner and camera*. In 2016 12th World Congress on Intelligent Control and Automation (WCICA), pages 983–986, June 2016.

[163] J. Dickmann, J. Klappstein, M. Hahn, N. Appenrodt, H. Bloecher, K. Werber, and A. Sailer. *Automotive radar the key technology for autonomous driving: From detection and ranging to environmental understanding*. In 2016 IEEE Radar Conference (RadarConf), pages 1–6, May 2016.

[164] Hermann Rohling. *SOME RADAR TOPICS: WAVEFORM DESIGN, RANGE CFAR AND TARGET RECOGNITION*. 2006.

[165] Wei-Yu Lee, Martin Dimitrievski, Ljubomir Jovanov, and Wilfried Philips. *Spatio-Temporal Consistency for Semi-supervised Learning Using 3D Radar Cubes*. In 2021 IEEE Intelligent Vehicles Symposium (IV), pages 785–790, 2021.

[166] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. CoRR, abs/1502.03167, 2015.

[167] Gianni Allebosch, Simon Van den Bossche, Peter Veelaert, and Wilfried Philips. *Camera-Based System for Drafting Detection While Cycling*. Sensors, 20:1241, 02 2020.

[168] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. *nuScenes: A multimodal dataset for autonomous driving*. CoRR, abs/1903.11027, 2019.

[169] Dariu Gavrila. *Pedestrian Detection from a Moving Vehicle*. In Proceedings of the 6th European Conference on Computer Vision-Part II, ECCV 2000, pages 37–49, Berlin, Heidelberg, 2000. Springer-Verlag.

[170] D. M. Gavrila, M. Kunert, and U. Lages. *A multi-sensor approach for the protection of vulnerable traffic participants the PROTECTOR project*. In IMTC 2001. Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference. Rediscovering Measurement in the Age of Informatics (Cat. No.01CH 37188), volume 3, pages 2044–2048 vol.3, 2001.

[171] Elzbieta Macioszek, Paulina Swierk, and Agata Kurek. *The Bike-Sharing System as an Element of Enhancing Sustainable Mobility-A Case Study based on a City in Poland*. Sustainability, 12(8):1–29, 2020.

[172] *Regulation (EC) No 78/2009 of the European Parliament and of the Council of 14 January 2009 on the type-approval of motor vehicles with regard to the protection of pedestrians and other vulnerable road users, amending Directive 2007/46/EC and repealing Directives 2003/102/EC and 2005/66/EC*. Official Journal of the European Union, 2009.

[173] R. Kalman. *On the general theory of control systems*. IRE Transactions on Automatic Control, 4(3):110–110, 1959.

[174] Georg A. Gottwald, Lewis Mitchell, and Sebastian Reich. *Controlling Overestimation of Error Covariance in Ensemble Kalman Filters with Sparse Observations: A Variance-Limiting Kalman Filter*. Monthly Weather Review, 139(8):2650 – 2667, 2011.

[175] Laura Leal-Taixe, Anton Milan, Konrad Schindler, Daniel Cremers, Ian D. Reid, and Stefan Roth. *Tracking the Trackers: An Analysis of the State of the Art in Multiple Object Tracking*. CoRR, abs/1704.02781, 2017.

[176] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. *Globally-optimal Greedy Algorithms for Tracking a Variable Number of Objects*. In Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '11, pages 1201–1208, Washington, DC, USA, 2011. IEEE Computer Society.

[177] A. Milan, K. Schindler, and S. Roth. *Multi-Target Tracking by Discrete-Continuous Energy Minimization*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(10):2054–2068, Oct 2016.

[178] Yutong Ban, Sileye Ba, Xavier Alameda-Pineda, and Radu Horaud. *Tracking Multiple Persons Based on a Variational Bayesian Model*, pages 52–67. Springer International Publishing, Cham, 2016.

[179] Caglayan Dicle, Octavia I. Camps, and Mario Sznaier. *The Way They Move: Tracking Multiple Targets with Similar Appearance*. In The IEEE International Conference on Computer Vision (ICCV), December 2013.

[180] A. Milan, S. Roth, and K. Schindler. *Continuous Energy Minimization for Multitarget Tracking*. IEEE TPAMI, 36(1):58–72, 2014.

[181] Laura Leal-Taixe, Michele Fenzi, Alina Kuznetsova, Bodo Rosenhahn, and Silvio Savarese. *Learning an Image-based Motion Context for Multiple People Tracking*. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2014.

[182] Loic Fagot-Bouquet, Romaric Audigier, Yoann Dhome, and Frederic Lerasle. *Improving Multi-frame Data Association with Sparse Representations for Robust Near-online Multi-object Tracking*, pages 774–790. Springer International Publishing, Cham, 2016.

[183] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M. Rehg. *Multiple Hypothesis Tracking Revisited*. In Computer Vision (ICCV), IEEE International Conference on. IEEE, December 2015.

[184] H. Kieritz, S. Becker, W. Hubner, and M. Arens. *Online multi-person tracking using Integral Channel Features*. In 2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pages 122–130, Aug 2016.

[185] Wongun Choi. *Near-Online Multi-target Tracking with Aggregated Local Flow Descriptor*. CoRR, abs/1504.02340, 2015.

[186] Aljosa Osep, Wolfgang Mehner, Markus Mathias, and Bastian Leibe. *Combined Image- and World-Space Tracking in Traffic Scenes*. In ICRA, 2017.

[187] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. *Tracking The Untrackable: Learning To Track Multiple Cues with Long-Term Dependencies*. CoRR, abs/1701.01909, 2017.

[188] Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. *Multi-Person Tracking by Multicut and Deep Matching*. CoRR, abs/1608.05404, 2016.

[189] T. Darrell, G. Gordon, M. Harville, and J. Woodfill. *Integrated Person Tracking Using Stereo, Color, and Pattern Detection*. International Journal of Computer Vision, 37(2):175–185, Jun 2000.

[190] Joaquin Salas and Carlo Tomasi. *People Detection Using Color and Depth Images*, pages 127–135. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[191] M. Bansal, S. H. Jung, B. Matei, J. Eledath, and H. Sawhney. *A real-time pedestrian detection system based on structure and appearance classification*. In 2010 IEEE International Conference on Robotics and Automation, pages 903–909, May 2010.

[192] B. K. Dan, Y. S. Kim, Suryanto, J. Y. Jung, and S. J. Ko. *Robust people counting system based on sensor fusion*. IEEE Transactions on Consumer Electronics, 58(3):1013–1021, August 2012.

[193] J. Han, E. J. Pauwels, P. M. de Zeeuw, and P. H. N. de With. *Employing a RGB-D sensor for real-time tracking of humans across multiple re-entries in a smart environment*. IEEE Transactions on Consumer Electronics, 58(2):255–263, May 2012.

[194] Max Bajracharya, Baback Moghaddam, Andrew Howard, Shane Brennan, and Larry H. Matthies. *A Fast Stereo-based System for Detecting and Tracking Pedestrians from a Moving Vehicle*. 28, 10 2009.

[195] Rafael Muñoz Salinas, Eugenio Aguirre, and Miguel García-Silvente. *People Detection and Tracking Using Stereo Vision and Color*. Image Vision Comput., 25(6):995–1007, June 2007.

[196] Emilio J. AlmazÃ¡n and Graeme A. Jones. *A Depth-based Polar Coordinate System for People Segmentation and Tracking with Multiple RGB-D Sensors*. In IEEE ISMAR 2014 Workshop on Tracking Methods and Applications, Sept 2014.

[197] D. M. Vo, L. Jiang, and A. Zell. *Real time person detection and tracking by mobile robots using RGB-D images*. In 2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014), pages 689–694, Dec 2014.

[198] Rafael Munoz-Salinas. *A Bayesian plan-view map based approach for multiple-person detection and tracking*. Pattern Recognition, 41(12):3665 – 3676, 2008.

[199] Rafael Munoz-Salinas, Rafael Medina-Carnicer, Francisco Madrid-Cuevas, and A Carmona-Poyato. *People detection and tracking with multiple stereo cameras using particle filters*. 20:339–350, 07 2009.

[200] Rafael Munoz-Salinas, Miguel Garcia-Silvente, and Rafael Medina-Carnicer. *Adaptive multi-modal stereo people tracking without background modelling*. 19:75–91, 02 2008.

[201] Wongun Choi, Caroline Pantofaru, and Silvio Savarese. *Detecting and Tracking People using an RGB-D Camera via Multiple Detector Fusion*. In Workshop on Challenges and Opportunities in Robot Perception, at the International Conference on Computer Vision (ICCV), 11/2011 2011.

[202] W. Choi, C. Pantofaru, and S. Savarese. *A General Framework for Tracking Multiple People from a Moving Camera*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(7):1577–1591, July 2013.

[203] Dominique Gruyer, Aurelien Cord, and Rachid Belaroussi. *Vehicle Detection and Tracking by Collaborative Fusion Between Laser Scanner and Camera*. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 5207–5214, 2013.

[204] Raphaël Labayrade, Cyril Royere, Dominique Gruyer, and Didier Aubert. *Cooperative Fusion for Multi-Obstacles Detection With Use of Stereovision and Laser Scanner*. Autonomous Robots, 19(2):117–140, Sep 2005.

[205] Tiantian Bao, Zhenkai Zhang, and Mohamad Sabahi. *An Improved Radar and Infrared Sensor Tracking Fusion Algorithm Based on IMM-UKF*. pages 420–423, 05 2019.

[206] J. Burlet and M. Dalla Fontana. *Robust and efficient multi-object detection and tracking for vehicle perception systems using radar and camera sensor fusion*. IET Conference Proceedings, pages 24–24(1), January 2012.

[207] K. Lee, Y. Kanzawa, M. Derry, and M. R. James. *Multi-Target Track-to-Track Fusion Based on Permutation Matrix Track Association*. In 2018 IEEE Intelligent Vehicles Symposium (IV), pages 465–470, June 2018.

[208] A. Pereira Barata, F. W. Takes, H. J. van den Herik, and C. J. Veenman. *Imputation Methods Outperform Missing-Indicator for Data Missing Completely at Random*. In 2019 International Conference on Data Mining Workshops (ICDMW), pages 407–414, 2019.

[209] M. E. Silbert and C. A. Rea. *Track-to-track fusion with missing information: Empirical study: Tracking a variety of target types*. In 2012 15th International Conference on Information Fusion, pages 1661–1667, 2012.

[210] Yaakov Bar-Shalom, Fred Daum, and Jim Huang. *The probabilistic data association filter*. Control Systems, IEEE, 29:82 – 100, 01 2010.

[211] Arsene Fansi Tchango, Vincent Thomas, Olivier Buffet, Fabien Flacher, and Alain Dutech. *Tracking Multiple Interacting Targets Using a Joint Probabilistic Data Association Filter*. 06 2014.

[212] Ruben M. Claveria, David Acuna, Ren'e A. M'endez, J. F. Silva, and Marcos Eduardo Orchard. *Application of Multiple-imputation-particle-filter for Parameter Estimation of Visual Binary Stars with Incomplete Observations*. 2016.

[213] S. Jin, H. Huang, Y. Li, Y. Ren, Y. Wang, and R. Zhong. *An Improved Particle Filter Based Track-Before-Detect Method for Underwater Target Bearing Tracking*. In OCEANS 2019 - Marseille, pages 1–5, 2019.

[214] A.S. Housfater, Xiao-Ping Zhang, and Yifeng Zhou. *Nonlinear Fusion of Multiple Sensors with Missing Data*. volume 4, pages IV – IV, 06 2006.

[215] X. Zhang, A. S. Khwaja, J. Luo, A. S. Housfater, and A. Anpalagan. *Multiple Imputations Particle Filters: Convergence and Performance Analyses for Nonlinear State Estimation With Missing Data*. IEEE Journal of Selected Topics in Signal Processing, 9(8):1536–1547, Dec 2015.

[216] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[217] Rudolph Van Der Merwe, Arnaud Doucet, Nando De Freitas, and Eric Wan. *The Unscented Particle Filter*. 13, 01 2001.

[218] Chris Snyder. *Particle filters, the "optimal" proposal and high-dimensio nal systems*. 2011.

[219] Spruyt, Vincent and Ledda, Alessandro and Philips, Wilfried. *Robust arm and hand tracking by unsupervised context learning*. SENSORS, 14(7):12023–12058, 2014.

[220] Michael Isard and Andrew Blake. *ICONDENSATION: Unifying Low-Level and High-Level Tracking in a Stochastic Framework*. In Proceedings of the 5th European Conference on Computer Vision-Volume I - Volume I, ECCV '98, pages 893–908, London, UK, UK, 1998. Springer-Verlag.

[221] David Hall, Feras Dayoub, John Skinner, Haoyang Zhang, Dimity Miller, Peter Corke, Gustavo Carneiro, Anelia Angelova, and Niko SÃŒnderhauf. *Probabilistic Object Detection: Definition and Evaluation*. In 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), pages 1020–1029, 2020.

[222] F. Caron, M. Davy, E. Duflos, and P. Vanheeghe. *Particle Filtering for Multisensor Data Fusion With Switching Observation Models: Application to Land Vehicle Positioning*. IEEE Transactions on Signal Processing, 55(6):2703–2719, June 2007.

[223] D. B. Rubin. *Multiple Imputation for Nonresponse in Surveys*. Wiley, 1987.

[224] Augustine Kong, Jun S. Liu, and Wing Hung Wong. *Sequential Imputations and Bayesian Missing Data Problems*. Journal of the American Statistical Association, 89(425):278–288, 1994.

[225] Amira Mimouna, Ihsen Alouani, Anouar Ben Khalifa, Yassin El Hillali, Abdelmalik Taleb-Ahmed, Atika Menhaj, Abdeldjalil Ouahabi, and Najoua Essoukri Ben Amara. *OLIMP: A Heterogeneous Multimodal Dataset for Advanced Environment Perception*. Electronics, 9(4):560, 2020.

[226] M. Meyer and G. Kuschk. *Automotive Radar Dataset for Deep Learning Based 3D Object Detection*. In 2019 16th European Radar Conference (EuRAD), pages 129–132, 2019.

[227] Anton Milan, Laura Leal-Taixe, Ian D. Reid, Stefan Roth, and Konrad Schindler. *MOT16: A Benchmark for Multi-Object Tracking*. CoRR, abs/1603.00831, 2016.

[228] Francesco Zanlungo, Drazen Brscic, and Takayuki Kanda. *Pedestrian Group Behaviour Analysis under Different Density Conditions*. Transportation Research Procedia, 2:149–158, 2014. The Conference on Pedestrian and Evacuation Dynamics 2014 (PED 2014), 22-24 October 2014, Delft, The Netherlands.

[229] Keni Bernardin and Rainer Stiefelhagen. *Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics*. EURASIP Journal on Image and Video Processing, 2008(1):246309, May 2008.

[230] Bo Wu and Ram Nevatia. *Detection and Tracking of Multiple, Partially Occluded Humans by Bayesian Combination of Edgelet based Part Detectors*. International Journal of Computer Vision, 75(2):247–266, Nov 2007.

[231] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. *Subcategory-Aware Convolutional Neural Networks for Object Proposals and Detection*. In 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), pages 924–933, March 2017.

[232] Y. Xiang, A. Alahi, and S. Savarese. *Learning to Track: Online Multi-object Tracking by Decision Making*. In 2015 IEEE International Conference on Computer Vision (ICCV), pages 4705–4713, Dec 2015.

[233] C. Liu. Beyond Pixels: Exploring New Representations and Applications for Motion Analysis. Doctoral Thesis. Massachusetts Institute of Technology. May 2009. PhD thesis.

[234] Byungjae Lee, Enkhbayar Erdenee, Songguo Jin, Mi Young Nam, Young Giu Jung, and Phill Kyu Rhee. *Multi-class Multi-object Tracking Using Changing Point Detection*, pages 68–83. Springer International Publishing, Cham, 2016.

[235] W. Tian and M. Lauer. *Joint tracking with event grouping and temporal constraints*. In 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pages 1–5, Aug 2017.

[236] J. H. Yoon, C. R. Lee, M. H. Yang, and K. J. Yoon. *Online Multi-object Tracking via Structural Constraint Event Aggregation*. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1392–1400, June 2016.

[237] J. H. Yoon, M. H. Yang, J. Lim, and K. J. Yoon. *Bayesian Multi-object Tracking Using Motion Context from Multiple Objects*. In 2015 IEEE Winter Conference on Applications of Computer Vision, pages 33–40, Jan 2015.

[238] Shaofei Wang and Charless C. Fowlkes. *Learning Optimal Parameters for Multi-target Tracking with Contextual Interactions*. CoRR, abs/1610.01394, 2016.

[239] Nuri Benbarka, Jona Schröder, and Andreas Zell. *Score refinement for confidence-based 3D multi-object tracking*. arXiv preprint arXiv:2107.04327, 2021.

[240] Ziqi Pang, Zhichao Li, and Naiyan Wang. *SimpleTrack: Understanding and Rethinking 3D Multi-object Tracking*. arXiv preprint arXiv:2111.09621, 2021.

[241] Paul J. Werbos. *Applications of advances in nonlinear sensitivity analysis*. In R. F. Drenick and F. Kozin, editors, System Modeling and Optimization, pages 762–770, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg.

A

The pinhole camera model

This section formally explains the pinhole camera model as an approximation of the image formation process. We use these equations anytime we need to project a point which lies in 3-D space onto a flat surface (camera image). Throughout this thesis, the pinhole camera model was used mainly to project a point cloud captured by a 3-D lidar onto a camera view and compute a depth map. The rotational nature of the 3-D lidar head results in a three dimensional point cloud which represents the geometrical structure of the environment as a sparse and non-uniform sample. A representation of such 3-D points scanned from an urban environment in the German city of Karlsruhe is presented on the top plot on Figure A.1 while the corresponding camera image is presented on the bottom.

By projecting each 3-D point $\mathbf{z}_j$ from the captured point cloud $\mathcal{Z} = \{\mathbf{z}_j\}\,; \mathbf{z}_j \in \mathbb{R}^3$ onto the camera sensor we can obtain an image representation of the structure of the environment. For the sake of notational simplicity, this analysis re-uses the symbol $\mathbf{z}$ to represent a 3-D point, which should not be confused with the definition of an object detection. For the projection to produce the correct image, matching the content captured by the camera, we need several calibration parameters. Firstly the relative lidar to camera pose matrix $P_{l\to c} = (R_{l\to c}\,|\mathbf{t}_{l\to c})$ consisting of the rotation matrix $R \in \left\{\mathbb{R}^{3\times 3}\,\middle|R^T R = I\,, |R| = 1\right\}$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$, defining the rigid transform between the lidar and camera coordinates systems. Even though the two sensors are usually mounted in close proximity to each other, there are small differences in their locations and pose which can lead to a bad projection if not accounted for.

Formally, each point in the lidar coordinate system $\mathbf{z}_l$ can be transformed to a 3-D point in the camera coordinate system $\mathbf{z}_c$ by applying the rigid transformation using

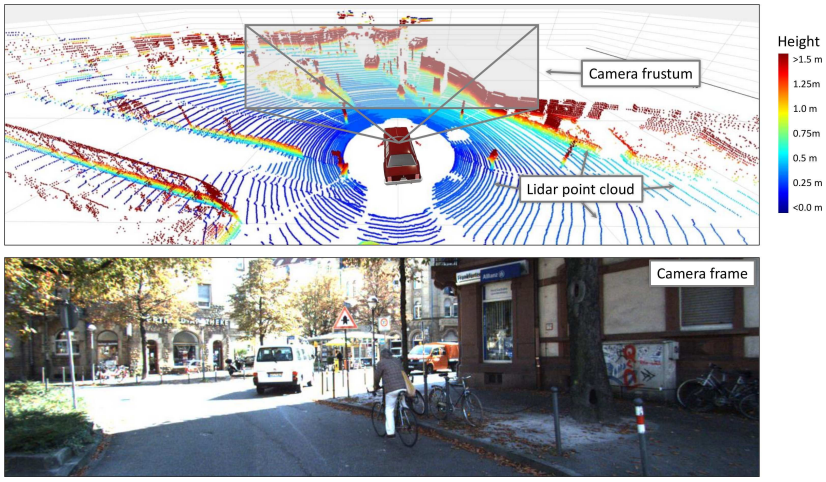


Figure A.1: Urban traffic scene captured by 3D lidar and visible light camera. Top: point cloud color coded to height, bottom: corresponding camera frame. Data from the KITTI dataset.

homogeneous coordinates:

$$\begin{bmatrix} \mathbf{z}_c \\ 1 \end{bmatrix} = \begin{bmatrix} R_{l\to c} & \mathbf{t}_{l\to c} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{z}_l \\ 1 \end{bmatrix}. \tag{A.1}$$

Then, in order to project this point $\mathbf{z}_c$ onto the correct pixel position $[u, v]^T$ on the camera imaging plane, where u is the pixel row number and v is the pixel column number, we need to apply a geometrical image formation process. Without going into unnecessary details, the pinhole camera model can be used as a good approximation of the image formation process, and the image coordinates can be computed through the perspective projection:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{z_c} K_c P_{l\to c} \begin{bmatrix} \mathbf{z}_l \\ 1 \end{bmatrix}, \tag{A.2}$$

where the projection matrix $K_c P_{l\to c}$ consists of the camera extrinsic matrix relative to the lidar: $P_{l\to c}$ and the intrinsic matrix K_c which defines the physical parameters of the camera imaging sensor. These intrinsic parameters are the focal lengths $\{f_x, f_y\}$, the offset of the principal point $\{u_0, v_0\}$ wrt. to the sensor origin and the skewness of the plane s. Formally K is be constructed from the following transformations and the

zero vector $\mathbf{0} = [0, 0, 0]^T$:

$$K = \begin{bmatrix} 1 & 0 & u_0 \\ 0 & 1 & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_u & 0 & 0 \\ 0 & f_v & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{s}{f_u} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{0}. \tag{A.3}$$

Finally, depending on the lens quality of the camera, the ideal image coordinates $[u, v]$ of each projected point will not match to the real observable coordinates $[u_d, v_d]$ which are usually distorted. The amount of lens distortion causing the discrepancy between each $[u, v]$ and $[u_d, v_d]$ is varying across the imaging sensor and is a function of the lens optical design. Again, without going into unnecessary details, to a safe degree of accuracy, we can assume that this lens distortion can be modeled using the radial distortion model. The model uses a quadratic (or sometimes higher order polynomial) function to map distorted pixel locations from their ideal projected positions:

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = \left(1 + k_1 r^2\right) \begin{bmatrix} u - u_0 \\ v - v_0 \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}, \tag{A.4}$$

where k_1 is a distortion coefficient and r is the radius of the pixel relative to the principal point: $r = \sqrt{(u - u_0)^2 + (v - v_0)^2}$. By setting the pixel value at position$[u, v]$ to the range of the corresponding 3-D point, $D_{u,v} \leftarrow \|\mathbf{z}_j\|_2$, we generate a depth image D. Similarly, if the lidar provides additional measurements such as infra-red reflectance, we can set the corresponding pixel value u, v to the measured infra-red reflectance value at the corresponding lidar 3-D point and generate an infra-red reflectance image $I^{[IR]}$.

B

Contraharmonic Mean Filter derivation

This section formally explains the partial derivatives for the proposed approximation of the Contraharmonic Mean Filter in Subsection 4.4.2. This filter approximation is differentiable with respect to the input image pixel values $I(\mathbf{u})$ and both the filter mask w and the filter order k, as shown in [87]. Thus, we can use gradient descent on these parameters, optimizing them using the loss function $L(w, k; I(\mathbf{u}))$ and back-propagation of its gradients [241]. Back-propagation applies the chain rule of derivatives to propagate the gradients down to the input layer, multiplying them by the Jacobian matrices of the traversed layers. The gradient of the CHM filter consists of the partial derivatives of L with respect to the filter mask w, the filter order k and the input I and can be computed as:

$$\frac{\partial L}{\partial w} = \tilde{I}^{k+1} * f_1(\mathbf{u}) + \tilde{I}^{k} * f_2(\mathbf{u}), \tag{B.1}$$

$$\frac{\partial L}{\partial k} = I^{k+1} \cdot \log(I) \cdot \left(\frac{I}{I^k * w} * \tilde{w} \right) + I^{k} \cdot \log(I) \cdot (f_2(\mathbf{u}) * \tilde{w}), \tag{B.2}$$

where $\tilde{I}$ and $\tilde{w}$ indicate flipping along the spatial dimensions, $\cdot$ is the element-wise multiplication and $u(x)$ and $v(x)$ are the two partial results of the back-propagation:

$$f_1(\mathbf{u}) = \frac{I(\mathbf{u})}{(I^k * w)(\mathbf{u})}; \quad f_2(\mathbf{u}) = \frac{-I(\mathbf{u})\left(I^{k+1} * w\right)(\mathbf{u})}{(I^k * w)(\mathbf{u})}, \tag{B.3}$$

In our application where we encounter empty regions which we need to complete we can safely fix the CHM filter order k to a large enough positive value. Thus, we are mainly focused on learning the content of the structuring element w. Finally, the

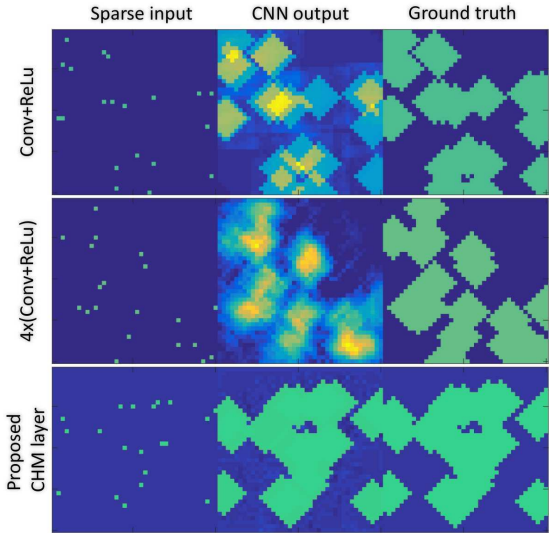


Figure B.1: Using a CNN to learning the morphological dilation operation with diamond structuring element of radius 5. From top to bottom: a simple CNN model, a CNN with 4 convolutional in a chain, and a CNN consisting of the proposed CHM filter block.

partial derivative with respect to the input $I(\mathbf{u})$ is:

$$\frac{\partial L}{\partial I} = f_1(\mathbf{u}) + f_2(\mathbf{u}) = \frac{I(\mathbf{u}) - I(\mathbf{u})\left(I^{k+1} * w\right)(\mathbf{u})}{\left(I^k * w\right)(\mathbf{u})}, \tag{B.4}$$

To illustrate the learning power of the proposed morphological block consider the following toy example: we want to train a CNN module which mimics a diamond morphological dilation operator of radius 5. In this toy experiment we will train three different CNN models, the first consisting of a single convolutional layer and a ReLu activator, the second will be a deeper model consisting of 4 such blocks in a chain, and the last will be a model consisting of a single CHM filter block. Since a deep CNN can be considered as an universal function approximator, simulating an arbitrarily simple function such as morphological dilation should not pose a problem. We will train all three network models using the same optimizer, loss function and other hyper-parameters. At the end of each training run, all three CNN models output a vastly different image as can be seen on B.1. The output from the simplest model seems to learn the shape of the diamond structuring element, however it is unable to generate the desired dilation output where the results of dilating pixels inside the structuring element should not accumulate. The more complex network consisting of 4 convolutional operators seems to perform better, however some accumulation of the responses is still evident. Contrarily to the previous two, a single CHM filter block is able to reconstruct the morphological dilation operator almost perfectly.

Aerial view of a traffic interchange with multiple autonomous vehicles.