Fortschritt-Berichte VDI

VDI

Reihe 8

Mess-,
Steuerungs- und
Regelungstechnik

Nr. 1246

Dipl.-Ing. Georg Tanzmeister,
München

Grid-based Environment Estimation for Local Autonomous Vehicle Navigation

Lehrstuhl für Steuerungs- und Regelungstechnik
Technische Universität München
Univ.-Prof. Dr.-Ing./Univ. Tokio Martin Buss

Grid-based Environment Estimation for Local Autonomous Vehicle Navigation

Georg Tanzmeister

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:	Jun.-Prof. Dr. rer. nat. Martin Kleinsteuber
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr.-Ing. habil. Dirk Wollherr
2.	Univ.-Prof. Dr.-Ing. Hans-Joachim Wünsche

Die Dissertation wurde am 13.04.2015 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 21.12.2015 angenommen.

Fortschritt-Berichte VDI

Reihe 8

Mess-, Steuerungs- und Regelungstechnik

Dipl.-Ing. Georg Tanzmeister, München

Nr. 1246

Grid-based Environment Estimation for Local Autonomous Vehicle Navigation

VDI verlag

Tanzmeister, Georg
Grid-based Environment Estimation for Local Autonomous Vehicle Navigation
Fortschr.-Ber. VDI Reihe 8 Nr. 1246. Düsseldorf: VDI Verlag 2016.
166 Seiten, 54 Bilder, 8 Tabellen.
ISBN 978-3-18-524608-1, ISSN 0178-9546,
€ 62,00/VDI-Mitgliederpreis € 55,80.

Keywords: Autonomous Vehicles – Environment Model – Grids – Mapping – Tracking – Navigation – Road Boundary – Collision Checking and Cost Evaluations – Laser Scanners – Radar Sensors

This dissertation is focused on the environment model for automated vehicles. A reliable model of the local environment available in real-time is a prerequisite to enable almost any useful activity performed by a robot, such as planning motions to fulfill tasks. It is particularly important in safety critical applications, such as for autonomous vehicles in regular traffic. In this thesis, novel concepts for local mapping, tracking, the detection of principal moving directions, cost evaluations in motion planning, and road course estimation have been developed. An object- and sensor-independent grid representation forms the basis of all presented methods enabling a generic and robust estimation of the environment. All approaches have been evaluated with sensor data from real road scenarios, and their performance has been experimentally demonstrated with a test vehicle.

Bibliographische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie; detaillierte bibliographische Daten sind im Internet unter http://dnb.ddb.de abrufbar.

Bibliographic information published by the Deutsche Bibliothek
(German National Library)
The Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliographie (German National Bibliography); detailed bibliographic data is available via Internet at http://dnb.ddb.de.

Als Manuskript gedruckt. Printed in Germany.
ISSN 0178-9546
ISBN 978-3-18-524608-1

Foreword

This thesis summarizes my research as a doctoral student at the Institute of Automatic Control Engineering (LSR) of the Technische Universität München and at the Department of Automated Driving, Active Safety, and Sensors of the BMW Group Research and Technology in Munich.

First, I want to thank Prof. Dirk Wollherr for supervising my thesis. It would not have been possible without him and I am grateful for the scientific freedom and the trust he has always given me to follow my own ideas. From the BMW Group, I want to thank first of all Martin Friedl, Werner Huber, Nico Kämpchen, and Helmut Spannheimer, for the extraordinary pleasant work environment, for giving me the opportunity to pursue this thesis, and for the support in every regard.

I would also like to thank all my colleagues at the BMW Group Research and Technology, in particular, Moritz Werling, Yves Pilat, Michael Aeberhard, Mohammad Bahram, Sebastian Rauch, Benjamin Gutjahr, and Thomas Hofmann, as well as all my colleagues at the LSR, in particular, Andreas Lawitzky, Daniel Carton, and Daniel Althoff, for the detailed and long discussions that often led to new ideas, but especially for the wonderful time we shared.

This thesis has been supported by the student projects of Sascha Steyer, Michael Schreibauer, Matthias Beck, Julian Thomas, and Eduard Feicho. I want to thank all of them for their work and their commitment.

Finally, my greatest thanks go to my parents for their constant support throughout my entire education, to my sister for her always encouraging words, and to my friends in Vienna who are keeping our friendship, as if we still lived in the same city.

Munich, 2015 Georg Tanzmeister

Contents

Notations

Abbreviations

CUDA	Compute Unified Device Architecture by Nvidia
DGPS	Differential Global Positioning System
DST	Dempster–Shafer Theory of Evidence
DSTMap	Dempster–Shafer Theory Map
FAMOD	Fast Approximate Morphological Grayscale Dilation
FOV	Field of View
GTAM	Grid-based Tracking And Mapping
MGCS	Map Grid Coordinate System
PMap	Particle Map
RCE	Road Course Estimation
ROC	Receiver Operating Characteristic
RRT	Rapidly Exploring Random Tree
SCS	Sensor Coordinate System
SGCS	Scan Grid Coordinate System
SLAM	Simultaneous Localization And Mapping
VCS	Vehicle Coordinate System
vHGW	van Herk-Gil-Werman Algorithm
vHGW-360	Modified van Herk-Gil-Werman Algorithm
WCS	World Coordinate System

Conventions

Scalars and *vectors* are denoted by lower case letters in italic type ($a, b, \ldots$). *Matrices* are denoted by upper case letters in italic type ($A, B, \ldots$). *Functions* are denoted by lower case letters ($f, g, \ldots$). *Curves* and *angles* are denoted by lower case Greek letters ($\tau, \omega, \ldots$). Number sets are denoted by upper case letters. Special number set, such as the set of natural numbers, are denoted by blackboard bold letters ($\mathbb{N}, \mathbb{R}, \ldots$). Other sets are denoted by standard calligraphic letters ($\mathcal{A}, \mathcal{B}, \ldots$). Note that in this thesis, it does not make a difference if the indices are superscripts or subscripts, e.g., $x_{ijk} = x_{ij}^{k}$. Probability density functions are denoted by $p(\cdot)$. The probability that a random variable Y has value y is denoted by $p(Y = y)$, but will be abbreviated as $p(y)$. The joint probability $p(x_1, x_2, \ldots, x_t)$ is denoted by $p(x_{1:t})$. The belief mass $m(\{A\})$ of the set $\{A\}$ in the Dempster–Shafer theory of evidence is abbreviated as $m(A)$. Single variables within this thesis may deviate from this notation to be conform with standard notation or to reduce ambiguities. These

deviations are, however, clearly highlighted. The mathematical notation that is used is given in the following:

A^{T}	transpose of matrix A
A^{-1}	inverse of matrix A
$\det(A)$	determinant of matrix A
$\det(a, b)$	determinant of matrix build by vectors a and b
$\mathrm{diag}(a, b)$	diagonal matrix with scalar entries a and b
$\|\|x\|\|$	Euclidean norm of vector x
$\|x\|$	absolute value of scalar x
$\|\mathcal{X}\|$	cardinality of set $\mathcal{X}$
$a\,b$	scalar or component-wise vector multiplication of a and b
$a \cdot b$	inner (dot) product of two vectors a and b
$\emptyset$	empty set
$\sphericalangle a, b$	angle between two vectors a and b

Symbols

General	
$f(\cdot), g(\cdot), h(\cdot)$	functions
i, j, k	index or integer number
l	length index
n_x	number of entities x
$\mathcal{N}(\mu, \sigma)$	normal distribution with mean μ and variance σ
$\mathcal{N}(x; \mu, \sigma)$	normal distribution with mean μ and variance σ evaluated at x
$\mathrm{O}(\cdot)$	big O notation; Landau notation
q	robot configuration
R_α	rotation matrix of α degrees
t	time index
$\mathcal{U}(x, y)$	uniform distribution with lower bound x and upper bound y
w	weight
x	robot state
ε	small arbitrary number
η	normalizer
μ	mean
θ	orientation of the robot
σ	standard deviation
$\mathbb{B}$	set of boolean values
$\mathbb{N}$	set of natural numbers
$\mathbb{R}$	set of real-valued numbers
$\mathbb{R}_0^+$	set of non-negative real-valued numbers

Subscripts

$()_l, ()_r$	referring to the left and right
$()_{\min}, ()_{\max}$	referring to the minimum and maximum
$()_R, ()_T$	referring to the radial and tangential component
$()_S, ()_G$	referring to the start and goal
$()_t$	referring to the time instance t
$()_v$	referring to the velocity component
$()_x, ()_y$	referring to the position component; to the x and y component
$()_{[\cdot]}$	referring to the particle

Mapping and Tracking

a, b	scalar parameter
c	scalar conflict in Dempster's rule of combination
D	subset of frame of discernment denoting *dynamic*
F	subset of frame of discernment denoting *free*
$m(A)$	belief mass of set A in Dempster–Shafer theory
m_p	map grid representing belief masses from particle map
m_{s_i}	scan grid representing belief masses of sensor s_i
m_s	scan grid representing belief masses after sensor data fusion
m_t	map grid representing final belief masses at time instance t
n_{cells}	number of grid cells per dimension
n_χ^i	actual number of particles in cell i
$n_\chi^{i,\text{des}}$	desired number of particles in cell i
$n_\chi^{i,\max}$	maximum number of particles per cell
o_t^{MGCS}	origin of map grid coordinate system at time t
$p_{\text{surv}}(\chi_{[k]})$	survival probability of particle $\chi_{[k]}$
$p_{\text{surv}}^{\max}$	maximum survival probability
$p_{\text{surv}}^{\min}$	minimum survival probability
r	radius of circle on which vehicle rotates in local grid
S	subset of frame of discernment denoting *static*
v	2-D velocity vector
$v_{\max}$	maximum velocity
v^*	true 2-D velocity vector
$v_{[k]}$	velocity component of k-th particle
V	multivariate random variable denoting 2-D velocity vectors
w_{rand}	probability of sampling a random particle during resampling
$x_{[k]}$	position component of k-th particle
$\mathcal{X}_t$	set of particles at time instance t
$\mathcal{X}_S^i$	set of static particles in cell i
$\mathcal{X}_D^i$	set of dynamic particles in cell i
$\overline{\mathcal{X}}_t$	predicted set of particles from $\mathcal{X}_{t-1}$

z	sensor measurement
α	angle
$\chi_{[k]}$	k-th particle
$\delta(x;y)$	Dirac delta distribution at y evaluated at x
ν_t	grid map of velocity vectors at time t
ν_t^i	cell i of map ν_t
$\vartheta_{\min}$	minimum uncertainty
Θ	frame of discernment
$\mathrm{bel}(\cdot)$	belief
$\mathrm{betP}(\cdot)$	pignistic probability distribution
$\mathrm{pl}(\cdot)$	plausibility
$\oplus^{\mathrm{C}}$	conjunctive rule of combination
$\oplus^{\mathrm{D}}$	Dempster's rule of combination
$\oplus^{\mathrm{J}}$	Jøsang's cumulative rule of combination

Motion Planning and Road Course Estimation

$b_{\{l,r\}}$	boundary element of left/right boundary
B	binary obstacle grid map
$\mathcal{B}_{\{l,r\}}$	set of left/right road boundary cells
c	cost
C_i	cluster i, i.e., set of trajectories that are in i-th cluster
$\mathcal{C}$	configuration space
$\mathcal{C}_{\mathrm{costs}}$	configuration space costs
$\mathcal{C}_{\mathrm{free}}$	set of collision-free configurations
$\mathcal{C}_{\mathrm{obs}}$	configuration space obstacles
$d, d(\cdot,\cdot)$	distance; if not explicitly stated, standard Euclidean distance
$f_c(\tau)$	function yielding cluster of a path/trajectory τ
$f_m(q,u),\ f_m(x,u)$	motion model
$f_w(\cdot)$	weight function
$\mathcal{F}_w$	set of weight functions
l_a	axis length
$\mathcal{L}$	list of states
$\mathcal{L}_{\mathrm{closed}}$	closed list
$\mathcal{L}_{\mathrm{goal}}$	goal list
$\mathcal{L}_{\mathrm{open}}$	open list
M	grayscale grid map
n_c	number of clusters
n_{checks}	number of cost/collision evaluations
n_{cols}	number of columns of image/matrix
$n_{\mathrm{iter}}^{\max}$	maximum number of iterations
n_o	number of objects
n_{pixels}	number of pixels of image

n_{prim}	number of motion primitives
n_{rows}	number of rows of image/matrix
n_{slices}	number of layers of $\mathcal{C}_{\text{obs}}$ or $\mathcal{C}_{\text{costs}}$
n_τ	number of paths
o	occupied cell
$\mathcal{O}$	obstacle region; set of occupied cells
$\mathcal{O}_{\{l,r\}}$	set of occupied grid cells that are left/right of some separator
P	set of parameter
$\mathcal{P}$	polygon
$\mathcal{R}$	set of road courses
S	structural element; robot mask
$\mathcal{T}$	set of paths/trajectories
$\mathcal{T}_{\text{rep}}$	set of cluster representatives, i.e., the principal moving directions
u	action
U	action space
v	vector
v_{road}	estimated drivable velocity
$\mathcal{W}$	work space
α	steering angle of wheels of vehicle
β	semantic continuous road boundary
$\delta(\cdot)$	discretization function
φ	alternative symbol for road course
γ	generalized Voronoi diagram of semantic road boundaries
κ	curvature
$\lambda(\cdot)$	log odds ratio
π	alternative symbol for path/trajectory
ρ	road course
τ	path/trajectory
τ^c	path cells
τ^n	path nodes
τ_p	primary path
τ_s	smoothed path
ω	action trajectory
Ω	set of action trajectories
ξ	plausibility criterion
ψ	angle
$\text{pred}(\tau, \tau')$	path equivalence predicate between τ and τ'
$\text{proj}_{\mathcal{W}}(\cdot)$	workspace projection
$\oplus$	morphological dilation
$\ominus$	morphological erosion

Abstract

A reliable model of the local environment available in real-time is a prerequisite to enable almost any useful activity performed by a robot, such as planning motions to fulfill tasks. It is particularly important in safety critical applications, such as for autonomous vehicles in regular traffic. In this thesis, novel concepts for mapping, tracking, the detection of principal moving directions, cost evaluations in motion planning, and road course estimation have been developed. An object- and sensor-independent grid representation forms the basis of all presented methods enabling a generic and robust environment estimation.

Grid-based Tracking and Mapping (GTAM), a low-level approach for the simultaneous estimation of the dynamic and the static obstacles and their velocities is presented. Uncertainties are incorporated in a Dempster–Shafer environment model. The method overcomes the drawback of widely-used occupancy grid mapping, which is only defined for static environments and leads to artifacts, if applied when dynamic objects are in the perceptual field of the robot. The grid map of the static world from GTAM forms the basis of the subsequently presented methods.

The *principal moving directions* through the environment represent the main possible maneuvers of the vehicle for local navigation. They are detected by a path planning and path clustering approach. Two path planner families are combined in order to efficiently sample a set of collision-free paths. A path equivalence definition is provided to cluster the paths, which is motivated by path homotopy but does not require that all paths end at the same point.

The costs of paths often arise due to the particular workspace, such as the distances to the nearest obstacles in order to prefer high clearance. The concept of configuration space obstacles is generalized to *configuration space costs*, which allow costs and collisions to be performed in the configuration space, i.e., incorporating the robot shape. Furthermore, two algorithms for their efficient calculation on graphics hardware are presented.

The methods from above form the basis of an indirect approach to *road course estimation*. The road topology is extracted using the principal moving directions as boundary separators, and the road boundaries are individually estimated for each detected roadway given the grid map.

All developed methods have been evaluated with sensor data from real road environments and their performance has been experimentally demonstrated with a test vehicle.

Zusammenfassung

Ein aktuelles und zuverlässiges Umfeldmodell ist Kernkomponente praktisch jedes realen Robotersystems und unverzichtbar in sicherheitskritischen Anwendungen wie bei autonomen Fahrzeugen. Ein Roboter wird dadurch erst befähigt sinnvolle Aufgaben, wie beispielsweise einen bestimmten Ort zu erreichen, durchzuführen. In der vorliegenden Dissertation werden neuartige Konzepte für die lokale Kartierung, die Verfolgung von dynamischen Objekten, die Erkennung der Hauptbewegungsrichtungen, die Kostenevaluierung für Pfad- und Trajektorienplanung sowie die Schätzung des Fahrbahnverlaufs vorgestellt. Ihnen allen liegt eine gitterbasierte Darstellung zu Grunde, welche ohne objekt- und sensorspezifische Annahmen auskommt und dadurch eine sowohl generische als auch robuste Schätzung des Umfeldmodells ermöglicht.

Die Arbeit beginnt mit der Präsentation von *GTAM*, ein Verfahren bei dem gleichzeitig sowohl die statische als auch die dynamische Umgebung anhand von Sensordaten geschätzt wird. Im Gegensatz zu klassischen Belegungskarten, welche nur für statische Umgebungen definiert sind und bei denen dynamische Objekte zu ungewollten Artefakten führen, liefert das Verfahren ein einheitliches und konsistentes Abbild der Umgebung inklusive Geschwindigkeitsinformationen. Die Belegungskarte der statischen Umgebung bildet die Basis für die im Weiteren vorgestellten Methoden.

Die *Hauptbewegungsrichtungen* durch die lokale Umgebung repräsentieren die Manöveroptionen des Fahrzeugs. Sie werden durch eine Kombination aus Pfadplanung und -gruppierung erkannt. Dazu werden zwei verschiedene Pfadplanungsfamilien kombiniert und ein Äquivalenzkriterium definiert, welches durch die Pfadhomotopie motiviert ist.

Bei der kostenabhängigen Pfad- und Trajektorienplanung sind die Kosten oftmals durch die lokale Umgebung gegeben wie etwa Abstand zu Hindernissen. Um Form und Ausdehnung des Roboters für die Kostenberechnung, welche die Kollisionsprüfung miteinschließt, berücksichtigen zu können, wird das Konzept der Konfigurationsraumobjekte auf *Konfigurationsraumkosten* erweitert sowie zwei effiziente Algorithmen für deren Berechnung auf Grafikkarten vorgestellt.

Die obigen Ansätze bilden die Basis eines indirekten Verfahrens für die *Schätzung des Fahrbahnverlaufs*. Hierbei wird die lokale Topologie der Straße anhand der Hauptbewegungsrichtungen extrahiert und für jede erkannte Fahrbahn die zugehörige Randbebauung geschätzt.

Alle entwickelten Methoden wurden mit Realdaten aus Fahrten mit einem Versuchsfahrzeug in diversen Verkehrsszenarien evaluiert und deren Performanz demonstriert.

1 Introduction

The vision of cars that drive themselves range at least back to the 1920s, where first preliminary experiments with a self-driving vehicle have been conducted [153]. Signals to remote-control a driverless car were sent by a human out of a following second car. In addition to the long history and a large number of other research prototypes since then, the desire for autonomous vehicles has been continuously further increased due to numerous science fiction works, where the superiority of robotic cars is often highlighted.

The three biggest believed benefits of automated vehicles are: safety, efficiency, and comfort [53, 110, 134, 152]. Despite numerous safety systems already integrated in modern cars, the number of accidents and fatalities is still tremendous. In Germany alone, there were around 2.4 million accidents recorded by the police in 2013 [146]. It is estimated that a whopping 90-95% of all road accidents are caused by human errors [53, 134]. A study of the United States National Highway Traffic Safety Administration (NHTSA) showed that in around 80% of all accidents and in 65% of all near accidents the driver was inattentive [46]. In addition to increased safety, the effects on efficiency and comfort are manifold. The time spent in the vehicle can be used more efficiently, which is especially useful in our fast paced, always-reachable, and always-connected world. Much time is wasted for daily commutes, in traffic jams, or when circling around blocks while searching for an empty parking spot. New, large-scale mobility concepts are possible, such as robotic taxis that are called for pick up and drop off. Efficiency, however, also targets the costs. Due to increased safety and thus a decrease in the risk of damage, insurance rates are expected to drop for automated vehicles [110], as well as the overall fuel consumption [134].

Given the expected potentials, legal frameworks for the operation of autonomous vehicles are started to be established. Different *degrees* of automation have been defined. The German Federal Highway Research Institute (Bundesanstalt für Straßenwesen) has elaborated definitions ranging from driver only, to assisted driving, to partially automated, highly automated, and fully automated driving [58]. The NHTSA has also established similar levels of automation [117].

This thesis targets technological aspects of automated vehicles in the context of robotics. It presents new methods for the representation and the understanding of the environment around the vehicle based on real-time sensor data with focus on local navigation in real street environments. Before presenting the research topic and the contributions in Section 1.3 in more detail, the impact of prior knowledge for autonomous vehicle navigation is discussed. Most of the current autonomous and automated driving systems heavily rely on very detailed a-priori global maps of the environment, which are required to be available to the vehicle. Such maps are often regarded as the key component to a success of automated vehicles [110, 134, 152]. The impact of prior knowledge in the form of high-precision maps and the interaction with online, sensor-based knowledge is the topic of the next section.

1.1 Prior Knowledge for Autonomous Navigation in Road Scenarios

Since the 1980's and especially triggered by research projects, such as the Eureka Prometheus project [55], and challenges, such as the DARPA Grand and Urban Challenges [39, 40, 41], research on autonomous vehicles produced a large amount of autonomous vehicle systems, e.g., [1, 26, 45, 62, 81, 105, 112, 131, 137, 154, 155, 161, 180]. Although prior work on autonomous vehicles existed before, it usually relied on special modified and expensive road infrastructure for vehicle guidance. The Universität der Bundeswehr München with the vehicle *VaMoRs* and the 4-D approach [44, 45, 174] pioneered in autonomous vehicle navigation based on environment perception. Other research institutes, such as the Carnegie Mellon University with the *Navlab* vehicles [76] also established pioneer work in automated driving. Rather than presenting every system that has been developed since in detail, in this section, a navigation characterization is attempted according to the required amount of prior knowledge about the environment and its effect on navigation. A similar classification of navigation paradigms is found in [105]. Here, it is focused, however, on the influence of the available prior knowledge as well as the resulting system characterizations of an automated driving system in regular traffic.

1.1.1 High Prior Knowledge Navigation

Many of the early works in robot motion planning studied the problem of navigation detached from sensor-based environment perception. It is assumed that a given map of the environment exists, which holds all relevant information for performing a given task, such as to find a collision-free and feasible path from a start to a goal state. In classic robotic applications, the map typically holds the obstacles of the environment. Such a map provides a huge advantage for a robotic system. They are generated offline under supervision, at least partially, by a human and thus guarantee a high accuracy and, even more important, a guarantee that they are correct.

For autonomous vehicles, the a-priori map typically holds the exact position of every lane, or often rather the center of each lane, with an accuracy that exceeds the one provided by maps available from standard consumer navigation systems today [132]. If the lane centers are available with high accuracy and a certain level of continuity, then they can directly serve as reference paths, which a vehicle can follow. Hence, the navigation problem is greatly facilitated. In the simplest case, an autonomous vehicle can simply follow the reference path. Simultaneously, a high amount of robustness is guaranteed, since the reference paths are proven to be valid, if the map is consistent with the environment. The majority of autonomous vehicles and automated driving systems make use of such high-precision maps. They thus rely on a high amount of prior knowledge about the environment, in which they are about to navigate. Note that the words precision and accuracy are not to be confused with the map itself, but denote the matching quality with the real world. Figure 1.1 shows such a highly automated driving system [1, 79]. The ego vehicle (white) uses the stored map information (3 lanes in this scenario) to drive highly automated on a regular highway. Visible are also the dynamic objects (blue boxes),

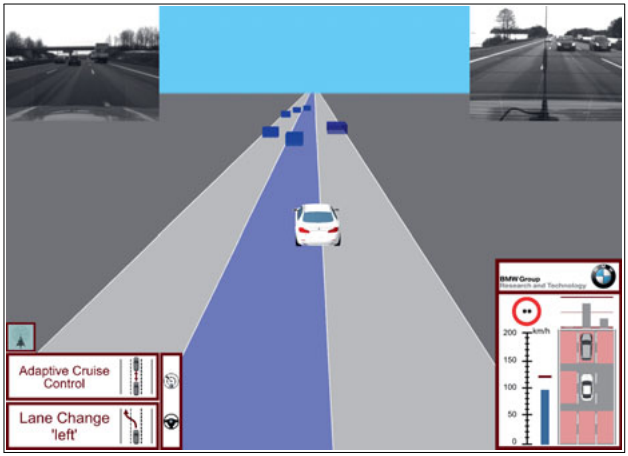


Figure 1.1: Highly automated driving system based on an a-priori high-precision map [1, 79].

which are tracked with sensor data from different sensor technologies integrated around the vehicle [11], the high-level planning module [15, 18], and camera images to the front and to the rear.

Robustness is indeed a key concern for autonomous vehicles. Unlike in other robotic systems, such as robotic vacuum cleaners for example, errors in the navigation are easily fatal. Hence, the benefit of a high amount of prior knowledge is obvious. Moreover, numerous rules control the traffic in real street environments, and an autonomous vehicle has to follow these rules the same way as human drivers need to. The violation of a traffic rule may also result in a fatal accident. Traffic rules, as well as static traffic signs, can also be integrated into the offline map.

With such a precise map, the complexity and requirements on sensor-based environment perception and interpretation decrease tremendously. As long as the map is valid, the vehicle will be driving on a valid path. Additionally, sensor-based algorithms can use the map to increase their robustness. Object tracking, for example, is facilitated, since the structure of the road is known, or traffic lights are easier to detect, since it is known where they are located in the world. Furthermore, static obstacles are only relevant within the area of the given lanes. Some static obstacles, such as road boundaries or walls in an indoor parking lot can also be included in the map.

The map alone is not enough, however. Its information has to be transformed into the current coordinate system of the vehicle, i.e., it needs to be localized with high precision in this map. Often, Differential GPS (DGPS) is used, but DGPS alone is usually not accurate enough, though. Especially in dense urban areas, multipath issues decrease the accuracy. Furthermore, DGPS is not always available, such as in tunnels. Therefore, it is often coupled with localization with landmarks that are also stored. Sometimes, whole scans or grid maps are recorded to localize with raw sensor measurements.

The robustness, however, comes at the price of availability. On the one hand, the system is only available, whenever such a specific a-priori map exists. Most automated vehicles use their own map with specific features, which needs to be specifically created. And on the other hand, as soon as the map is inconsistent with the real environment, the system fails. Creation and maintenance of the map usually require a huge effort. Whenever parts of the road layout change, the stored reference paths become invalid. Unfortunately, this happens often in real street environments. Road construction sites are a prominent example of such situations. In addition to road changes, localization features can also change. Most of the localization methods are robust enough to deal with a certain amount of wrong or missing features. However, outdoor environments are not as controllable as indoor environments and seasonal changes have a huge impact. Changes from bright and vivid vegetation during the summer season to snow during the winter season pose enormous challenges to localization systems.

In this paradigm, there exists an additional challenge in detecting whether or not the map is indeed consistent with the current road environment. Usually, however, it is still relied on a human for the detection. In order to detect deviations from the map, the true current road model needs to be estimated, at least to some extent, with the sensors and compared to the one that is stored in the map. This directly leads us to the next paradigm, i.e., maneuvering solely based on sensor information.

1.1.2 Sensor-based Navigation

Contrary to relying on an a-priori, high-precision map of the environment, is autonomous navigation solely based on sensors. Even with a high-precision map, sensor information is essential to detect and understand the dynamic environment, such as other traffic participants, obstacles, or electronic traffic signs. A stored map alone is never sufficient. In this paradigm, however, the sensors are used to estimate *all* relevant information about the local environment. In particular, the structure of the road and its semantics are estimated online in real-time, in order tell the motion planning modules where the vehicle is allowed to drive. It is assumed that the global guidance is given by the road layout, rather than stored in a map, and the task of the robot is to follow the road.

Many consumer robotic products, such as robotic vacuum cleaners mentioned above or robotic lawn mowers, follow this paradigm. The robot does not know the structure of the environment a-priori, but needs to use its sensors to build a model of it, which is used for navigation. Navigation in turn serves to fulfill a task, such as to clean a room or to walk a humanoid robot to a goal location [173]. This allows the robot to act in unknown and unstructured environments. An autonomous vehicle in an offroad environment can navigate in a similar way solely based on sensor information. It can move arbitrarily through the environment, as long as it does not collide with other objects. The only a-priori information that is relied on are GPS coordinates that hold the goal position.

In real street environments, there is however substantially less liberty in the choice of the robot motions. Unlike with an a-priori map, it is not known to the vehicle, how the course of the road and the individual lanes are evolving. Perhaps, the ego-lane is ending and the vehicle needs to change lane in order to continue to drive. Not only information about the ego lane is important, however, but all lanes of the road are required for lane changes

and in order to predict and interpret the environment. The local road model needs to be precise and particularly robust. If the estimation goes wrong, the vehicle will eventually leave the true lane possibly leading to fatal accidents.

Purely sensor-based navigation in regular street environments thus sets very high requirements on the algorithms for environment perception and interpretation. Even for experienced human drivers it takes a considerable mental effort to drive in complex street scenarios for the first time. Consider arriving and driving through a foreign city at night without a navigation system. If it requires a human effort to perform a task, in particular for a task that was intentionally designed to be performed by humans, such as driving in street environments structured for humans, the difficulty is expected to be tremendous to be performed by a machine.

On the other side, tasks such as precise and robust localization in a global map are nonexistent, since if there is no global map of the environment, there is nothing to localize within. Neither are problems due to multiple coordinate systems of the online, sensor-based map and the offline map.

Estimating the current road model as robust and as accurate in order solely rely on it for autonomous vehicle navigation is still an open problem. Many individual subproblems need to be solved in order to estimate the complete road model with sensors and this thesis targets important building blocks. Before going into more detail of the environment model and the particular contributions of this thesis, hybrid approaches of map-based and sensor-based navigation are discussed.

1.1.3 Low Prior Knowledge Navigation

Examining the two paradigms presented above, two observations are made regarding an autonomous vehicle system. On the one hand, relying on high-precision maps of the environment requires a considerable amount of maintenance of the map and it cannot be guaranteed that it will always be valid. Eventually, an automated vehicle will encounter an unmapped situation, such as a temporary construction site. Even if the map is learned online with a backend server using data from all available vehicles, at least the first car that enters the unmapped area, has to be able to cope with the situation. If the system is capable of detecting that the map is indeed wrong or it gets the information about a road change over communication, such as from a server, from the infrastructure, or from other vehicles, it needs to come to a safe state. Performing an emergency braking maneuver allows reaching a safe state in some scenarios, such as low speed environments like parking lots. In highway scenarios, performing an unreasonable (from a human's perspective) immediate stop does not only block the traffic, but possibly leads to severe accidents. Hence, the system needs to be able to continue to drive, at least to some extent and for some amount of time, based on the sensor information alone.

On the other hand, only relying on the sensors for navigation, is prone to errors. First, without any a-priori information, the system does not know, where it is supposed to go. Even for offroad navigation, a rough global route to the goal greatly facilitates the navigation task and decreases the chance of reaching a dead end. An autonomous system is preferred that maneuvers optimally on a global scale over one that misses, e.g., a highway exit, because the exit sign was currently occluded by another vehicle. Standard navigation

systems are common tools to help getting from a start to a goal and humans also use a-priori knowledge about the environment. It is substantially less demanding to drive a route, which was driven before, then driving on unknown streets.

The difference between this paradigm and the high prior knowledge paradigm is not necessary the map itself. It is the amount of trust that the environment does indeed and accurately correspond to that map. Less trust in the map means that the system needs to rely more on its sensors. For example, if the reference paths are not directly used from a stored map, but are estimated online with the help of possibly-inaccurate prior knowledge, slight deviations between the real environment and the map are likely to be managed by the system. The prior map can even just be used for rough global navigation. This way, even if the prior map is locally entirely wrong, such as at a highway construction site that turns in an S-shaped curve onto the roadway of the opposite direction, autonomous navigation may still be continued until eventually the prior map is correct again.

The transition between the amount of required and relied prior knowledge is smooth. Figure 1.2 puts sensor-based navigation and navigation based on a prior map side-by-side. As discussed above, the higher the accuracy and the trust in the prior map, the more robust is the system, as long as the map is valid. On the other hand, the system is only available where such an accurate map does exist and where it is indeed valid. The more it is relied on the prior map and the more accurate it needs to be, the more effort it takes to maintain it up-to-date. At the other end are the requirements for sophisticated and robust artificial intelligence. If the vehicle needs to find the course to maneuver by itself, the requirements for the local environment model rise accordingly. And third, the focus of the system design varies between a precise map paired with a precise localization within this map, and precise and redundant sensors and perception algorithms.

From the discussions above, it follows that a truly autonomous vehicle system needs to be able to maneuver locally solely based on the sensor information, at least to come to a safe state, even if a precise map is usually available. Prior knowledge, of course, helps to improve the performance.

Although the importance rises with decreasing map accuracy, the environment model, which is the focus of this thesis, is a building block of any real world autonomous vehicle.

1.2 Environment Model

The environment model comprises the whole pipeline, from raw sensor data, such as range and bearing readings or color pixels, to the understanding and the interpretation of the whole scene, such as mutual dependencies between traffic participants [4]. The environment model builds the basis for planning the motions of the robot in order to achieve intelligent actions and to solve tasks.

The amount of individual subproblems involved from raw sensor data to a complete understanding of the scene is tremendous and underlines the complexity involved. A variety of different sensors are used that possess and measure different individual characteristics. In the robotics community, most often laser scanners, camera systems, radar sensors, and ultrasonic sensors are used. There are algorithms that are particularly designed for one sensor, such as traffic sign recognition for cameras, but also sensor independent algorithms.

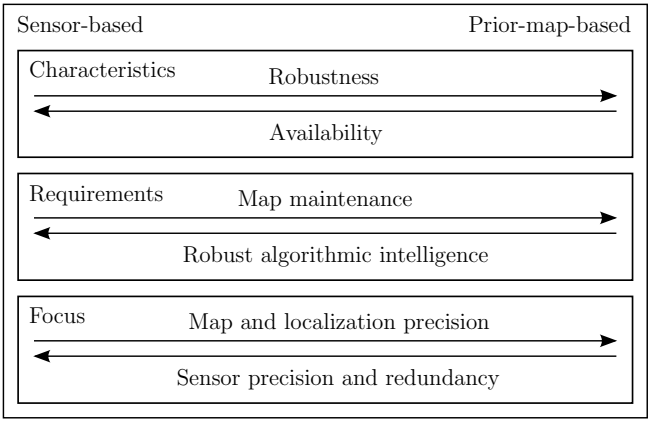


Figure 1.2: Autonomous vehicle navigation paradigms.

In order to increase robustness, the data from different sensors is often fused, since every sensor has strengths and weaknesses [80]. Sensor data fusion further increases the complexity. Without completeness, but in order to give the reader an impression of the tasks involved regarding the environment model for autonomous vehicles, in the following several objectives are given: detecting the static obstacles, detecting and tracking the dynamic objects, classification of objects, classification of scenes, vehicle indicator recognition, lane marking recognition, curb stone detection, detection of the principal moving directions, road boundary estimation, road model estimation, traffic sign recognition, traffic light recognition, prediction of objects, or the prediction of the whole scene.

The incomplete list of the problems from above is not meant, however, to suggest that many of the problems are already solved. The truth is, although huge progress has been made over the last decades, most of the proposed algorithms still do not meet the quality required to enable purely sensor-based autonomous vehicle navigation in arbitrary street scenarios, in particular in complex urban environments.

The first focus of this thesis is a robust, low-level model of the local environment representing the static and the dynamic objects. Rather than relying on specific features and on particular sensors, all algorithms developed are sensor-independent. This eases fusion of diverse sensor data. The low-level representation is in form of a *grid*, i.e., a discretization of the world into equal-sized 2-D cells. Although other representations have been proposed, e.g., stixels [17], grid-based representations have already been, and still are, popular for mapping for a considerable amount of time [157]. Due to the grid representation, no assumptions about the size, the shape, or specific characteristics of the objects in the environment need to be made. The grid representation is general and allows representing arbitrary, a-priori unknown objects, since any obstacle needs to be detectable, even those, which the system has not encountered before.

A grid-based representation provides a solid basis for *extractors*, i.e., algorithms that try to extract information. This is the second focus of this thesis. For example, grid-based

road boundary estimation and the detection of the principal moving directions are classes of extractors. The benefit of extractors on grids, rather than on raw sensor data, is that they already work with a filtered and sensor-independent representation. Moreover, grids can keep past and currently-occluded information, which eases information extraction. More details about the contributions of this thesis are given next.

1.3 Main Contributions and Outline of the Thesis

This thesis targets five components of the local environment model: mapping, low-level tracking, extraction of the principal moving directions, definition and generation of a structure for efficient collision and cost evaluation, and road boundary estimation. The main contributions are summarized in this section. More details about the characteristics of the approaches and the contributions are given in Section x.1.2 of Chapters 2–5. The related work is also given in the individual chapters, rather than in a single separate chapter, in order to directly put it in the context of the developed concepts.

Information about the static and the dynamic objects with their corresponding states in the local environment of the robot are the building blocks for almost any mobile robot system. Mapping the static environment and tracking the dynamic objects are complex but established tasks. However, they are usually performed separately using occupancy grid mapping [52, 157] for the static part, which does not require shape or object assumptions, and object tracking for the dynamic part, which does use shape and object assumptions and yields a list of objects. Chapter 2 presents a novel approach termed grid-based tracking and mapping (GTAM) that simultaneously estimates the static and the dynamic environment in a uniform grid-based representation using the Dempster–Shafer theory of evidence [42, 43, 142]. It avoids inconsistencies due to different environment representations and does not require that measurements must be a-priori divided into belonging to a static or to a dynamic object. It is also shown how the evidential Dempster–Shafer model is transformed to a standard probabilistic model. This low-level environment model builds the basis for all subsequent chapters.

The knowledge of the local street topology is important to detect forks in the road and road junctions. It is also essential for the road course and road boundary estimation presented later. Chapter 3 consists of two parts. First, a method is given to efficiently sample a set of collision-free and feasible paths through the environment in the absence of goal poses, which combines two different classes of planning algorithms. And then, the paths are clustered according to the obstacles in the local environment to find the principal moving directions. Compared to other techniques, which estimate the road network or the principal moving directions in road scenarios, this approach is able to provide the principal directions at arbitrary road shapes and is not restricted to quadratic or clothoid models, as often used. Due to motion planning, estimations over long horizons in complex road scenarios, such as urban roads or road construction sites, are achieved.

The most time-consuming part for most motion planning algorithms, such as the path planner presented in Chapter 3, is collision checking and cost evaluation. Precomputation techniques have the potential of providing a major performance increase. Most notably are the configuration space obstacles [96], a concept that reduces collision checking of the whole

robot shape against the obstacles to a single look-up. Chapter 4 generalizes this concept to configuration space costs. In addition to collisions, they allow the costs, incorporating the whole robot shape, also to be performed by a single look-up. Two approaches are presented to efficiently calculate the configuration space costs. They were implemented on a GPU and their performance substantially outperforms direct implementations.

Chapter 5 describes the road course and road boundary estimation. It builds upon all previously presented methods. It uses the Bayesian occupancy grid of the static environment from the grid-based tracking and mapping from Chapter 2 and extracts the principal moving directions, as described in Chapter 3. This is only possible to be performed in real-time due to a precalculation of the configuration space costs. Workspace cost maps designed for the particular application increase the quality of the principal moving directions and are directly incorporated in the configuration space. The extraction of the road courses and their corresponding boundaries is substantially facilitated using the principal moving directions, as they already separate the environment. Since the separators lie between the boundaries, their extraction is achieved locally. Furthermore, the number of potential road courses is also known, as it equals the number of principal moving directions. The road courses are validated according to their shape and recursively filtered to exclude implausible estimations.

All algorithms and systems developed as part of this thesis were implemented on, and tested with, an autonomous vehicle platform, given in Appendix A.1. They were thus all proven to run in real-time, which is mandatory for any online autonomous vehicle algorithm. Real-time, for the environment model, means that they run faster than the refresh-rate of the sensors, which was at 12-15 Hz. The whole system represented in this thesis, combining mapping, tracking, configuration space cost calculation, path planning, clustering, and road course estimation runs in real-time on a single standard computer. Most of the methods parallelize well and were implemented on a GPU, which has proven to be well capable of performing arbitrary parallel computations [3]. The hardware and software platform is given in Appendix A.2.

Finally, Chapter 6 gives qualitative and quantitative evaluations of the presented algorithms. Moreover, the road course estimation was used to autonomously navigate a vehicle through an unmapped road construction site. The boundaries consisted of traffic cones, parked vehicles, bushes, and walls of buildings. A fork in the road posed additional challenges. The vehicle did not have prior knowledge about the environment and did not use GPS. The thesis concludes with Chapter 7.

2 Grid-based Tracking and Mapping

A robust representation of the environment is essential for any mobile robot system. It forms the basis needed by many other modules, in particular by motion planning algorithms. Additionally, many recognition tasks, such as road course estimation, often build upon a filtered environment model, rather than building upon raw sensor data. It is therefore crucial for the overall performance of the whole robot system. Mapping and tracking are often done separately yielding different environment representations, which is likely to lead to inconsistencies. Often grids represent the static environment and a list of objects the dynamic environment. This chapter presents a combined estimation of the static and the dynamic environment in a uniform representation termed Grid-based Tracking and Mapping (GTAM). The framework allows integrating unclassified raw sensor measurements and provides a real-valued continuous estimate of the velocities, the static and dynamic occupancy, as well as the free space in a uniform grid-based representation. This chapter is based on work published in [8] in the context of this thesis.

2.1 Introduction

As given above, mapping and tracking are often two independent tasks. In mapping, feature-less occupancy grid mapping [52] has evolved to the standard approach in the robotics community. It allows representing arbitrary object shapes and data association, i.e., the problem of determining correspondences between sensor measurements and existing tracks, is solved implicitly with the grid structure. Tracking, on the other hand, is often done using model and shape assumptions [11].

2.1.1 State of the Art

In occupancy grid mapping, the world is assumed to be static. It is divided into a set of discretized cells rather than having a list of objects which implies making assumptions of how objects look like or what their shape is. A real-world object can be represented by an arbitrary number of cells. Although grid representations usually have higher memory requirements, the benefit of not having to deal with data segmentation and association often outweighs. Moreover, there are also hierarchical techniques minimizing the required memory and computation times [138], which are especially useful for 3-D mapping.

In standard grid mapping, the occupancy probability of each cell, which corresponds to a certain area in the world, represents the probability of that cell being occupied by a static object. Since every cell has attached a binary random variable, it simultaneously represents the probability of it being free space. The environment is, however, rarely entirely static. In some robotic applications, this is negligible. If the map is stored for later use and the

robot moves slowly through the environment, then it will most likely be able to make more observations of a certain area as being free space, than of it being temporarily occupied by a dynamic object. If the map is created and used in real-time, however, then cells that are temporarily occupied due to dynamic objects represent unwanted artifacts in the static map. Consider, e.g., an autonomous vehicle that uses the real-time generated grid map for collision checking against the static obstacles, and consider the case where another vehicle is driving in front of the ego vehicle. In such a scenario, the robot will constantly try to re-plan around the vehicle driving in front of it, or even trigger an emergency collision avoidance maneuver, since the way up front seems to be blocked by a static obstacle.

If the map is updated with a sensor that cannot measure the dynamics, such as a laser scanner, often inconsistencies between the map built so far and the current scan are used to detect and filter out measurements belonging to dynamic objects [24, 127, 140, 165, 166, 167]. It follows the idea that, if parts of the map were previously observed as free space, but are now observed to be occupied, and vice-versa, if the previous observations were occupied, but now result in free space, the occupancy observations must come from a dynamic object. Sometimes, the Dempster–Shafer theory of evidence is used to better model these inconsistencies as conflicts [113]. Although directly deleting these inconsistencies works in some scenarios, especially for fast objects that move parallel to the sensor axis, they often completely fail. Consider, e.g., a large vehicle crossing perpendicular to the robot sensor. Then, one particular cell will be occupied over multiple frames by different parts of the object. Moreover, using inconsistencies as evidence for dynamic information counteracts the original idea of filtering sensor measurements to deal with noise. Additionally, it is only feasible for very accurate sensors, although there are also approaches that try to classify the conflicts as either noise or dynamic [118]. Apart from approaches based on conflicting information, there also exists work that tries to detect the specific shape of the artifacts in the grid caused by the moving objects [168].

Similar important is a robust representation of the dynamic environment and the literature in the field of object tracking is vast. In some work, the inconsistencies during mapping are used as input to an object tracker [24, 127, 140, 167]. In other work, it is directly relied on an object tracker [119] to find and filter out the measurements belonging to the tracks. Obviously though, relying on object tracking simply transfers the problem, and with the notion of objects and tracks, in comparison to cells or data points, comes the data association problem. All of the above approaches have in common that the decision of whether a single sensor measurement belongs to a static or to a dynamic object is binary and its uncertainty is not modeled.

In the following, some of the approaches that combine the estimation of the static and the dynamic environment are presented in more detail. In [101], Rao-Blackwellized SLAM is combined with conditional particle filters for tracking, but all measurements are used for creating the map leading to problems in crowded environments. In SLAM with generalized objects [167], a joint posterior over all static as well as dynamic objects, together with the robot pose, is calculated, in contrast to SLAM with DATMO [167] by the same authors, which divides the problem and requires that measurements can be separated with regard to their dynamics. SLAM with generalized objects is in general computational infeasible, though, as the authors point out, and it builds upon the notion

of objects. In [32], a model-free, grid-based approach, the Bayesian Occupancy Filter (BOF), is presented. It uses a four-dimensional grid, with two dimensions for the location and two for the velocity. A four dimensional representation is, however, undesirable, since memory scales exponentially with the number of dimensions, and the velocities need to be discretized. Later, the BOF was reformulated to work on a two-dimensional grid, where each cell contains two probability distributions, one for the occupancy and one for the velocity [151]. The velocities are discretized into a histogram so that between consecutive estimations, the velocity corresponds to an exact integer cell displacement, which limits velocity accuracy, leads to errors, and strongly couples cell resolution to velocity resolution. In addition, due to its formulation, it requires, for every cell, a summation over all possible antecedent cells and velocities, which is expensive. Although, it can be speeded-up by using an existing map of the environment [25, 61], mapping techniques are especially useful in unknown environments.

In a different, particle-based approach [35, 36] the velocities do not need to be discretized, but are estimated as continuous distribution. The particles have a continuous position and a continuous speed and can move independently of the grid structure between cells. The authors describe the particles as both, velocity hypotheses and the building blocks of the environment. The particles in a particular cell represent, on the one hand, the velocity distribution, and on the other hand does the number of particles represent the occupancy likelihood. The approach yields promising results, but the exact role of the particles and what probability distribution they approximate is not entirely clarified. The resulting grids do not explicitly model free-space, as the absence of particles may either be due to unknown, not observed areas or to areas that are measured to be free. Moreover, the approach loses information about previously observed areas, since the particles die out if no measurements continuously support them, which is undesirable for the static objects. The first part of the presented approach in this chapter also uses a particle filter and was inspired by [35]. It is, however, only used to estimate velocity distributions. Occupancy as well as free space information is derived and filtered in a novel Dempster–Shafer model, as will be described in detail.

2.1.2 Approach and Contribution

In this chapter, a novel method termed Grid-based Tracking and Mapping (GTAM) is presented that simultaneously estimates the static as well as the dynamic environment. It is entirely grid-based and therefore does not rely on model and shape assumptions or on data association. It is used to create maps online in highly dynamic environments and to detect and track the dynamic world in a cell-based manner. In the following, the characteristics of GTAM are given:

- A particle filter is used to estimate continuous velocity distributions.
- The initial velocity sampling distribution for the particles is a combination of a uniform distribution and a Dirac distribution at $(0, 0)^{\mathrm{T}}$. This allows exactly modeling the static world and enables the filter to converge against the environment.

- Particle weighting is divided into a cell-based weight, equal for all particles in the cell, and a particle-specific intra-cell weight.
- The map uses a novel Dempster–Shafer frame of discernment, which explicitly allows differentiating between occupancy evidences from static objects, occupancy evidences from dynamic objects, and evidences, where the distinction is not known.
- Classification between static and dynamic is not binary using thresholds, but continuous evidences are derived based on the velocity distribution. Uncertainties in the classification are therefore considered.
- The ability of tracking temporarily occluded objects can be controlled with a particle survival probability and traded-off against computational performance.

The rest of this chapter is structured as follows. Section 2.2 presents the fundamentals, such as the environment model that is used and the way how the scan grids are generated. Section 2.3 then presents the grid-based velocity estimator, i.e., the particle map, which is used to derive evidences for static and dynamic occupancy. These evidences, together with free space, are filtered over time to increase robustness over noise, which is described in Section 2.4. Finally, in Section 2.5 first results are given, which will be complemented in Chapter 6.

2.2 Fundamentals

Before going into details of how the grid-based tracking and mapping works, some fundamentals need to be given including the environment model used, Section 2.2.1, an overview of the main components, Section 2.2.2, and the scan grid generation, Section 2.2.3.

2.2.1 Dempster–Shafer Environment Model

As mentioned above, in this work the Dempster–Shafer Theory (DST) of evidence [42, 43, 142] is used instead of the more common Bayesian inference as used in standard occupancy grid mapping [157]. Before presenting the environment model used in this work, the essentials are quickly revisited.

The Dempster–Shafer Theory of Evidence

The Dempster–Shafer theory of evidence can be seen as a generalization of the Bayesian theory [43]. Uncertainties are better represented in this model. While in the Bayesian model, a low probability of $p(A) = 1 - p(\neg A)$ implies a high probability of its negate, DST allows a specification of an *evidence* also for the latter, together with uncertainty about the system. Rather than probability distributions, the DST model deals with degrees of belief that are represented by belief functions. Central to the DST is the *frame of discernment* Θ, which represents the hypotheses about the world. The hypotheses are the atoms or singletons of Θ. However, instead of only regarding the atoms of Θ, in the DST model, every element of the power set 2^Θ is considered. This gives greater flexibility and

allows integrating partial knowledge in the form of evidence for the supersets of the atoms. Every set $A \in 2^\Theta$ is assigned a *basic belief mass* $m(A)$ using the *basic belief assignment* $m : 2^\Theta \to [0, 1]$ such that

$$\sum_{A \subseteq \Theta} m(A) = 1 \qquad \text{and} \qquad m(\emptyset) = 0. \tag{2.1}$$

It represents the evidence of a particular set A, not to be confused, however, with the combined evidence of all subsets of A. The DST defines lower and upper bounds for the support of the set A: the *belief*

$$\mathrm{bel}(A) = \sum_{B \subseteq A} m(B), \tag{2.2}$$

sometimes also called the total amount of justified specific support given to A [145], and the *plausibility*

$$\mathrm{pl}(A) = \sum_{B \cap A \neq \emptyset} m(B), \tag{2.3}$$

also referred to as the maximum amount of potential specific support that is given to A [145]. Two different basic belief assignments m_1 and m_2 can be combined with Dempster's rule of combination

$$m_1(A) \oplus^{\mathrm{D}} m_2(A) = \frac{1}{1-c} \sum_{B \cap C = A} m_1(B)\ m_2(C) \tag{2.4}$$

$\forall A, B, C \subseteq \Theta \neq \emptyset$ with the conflict

$$c = \sum_{B \cap C = \emptyset} m_1(B)\ m_2(C). \tag{2.5}$$

The particular application of the Dempster–Shafer theory of evidence is shown in the following sections.

FSD–Frame for Modeling the Environment

Now that some of the basics of the DST have been revisited, the frame of discernment, that represents the environment model, is presented. Evidential occupancy grids have been used before. Often, however, using the simple 2-class frame of discernment $\Theta = \{F, O\}$, i.e., free-occupied [113, 125, 176]. With this low-dimensional model, the benefits of the Dempster–Shafer theory in representing uncertainties are already exploited. In the Bayesian world, an occupancy probability close to 0.5 can either come from a low number of observations using an initial occupancy probability of 0.5, or it can come from a large number of contradicting measurements. Even greater is the utility of the DST, however, if the frame contains more than two atoms. Then, evidences for the supersets of atoms can be specified that are different from the total uncertainty Θ, as shown below.

In order to distinguish between evidence for occupancy coming from static objects and evidence for occupancy coming from dynamic objects, the frame of discernment

$$\Theta = \{F, S, D\} \tag{2.6}$$

is proposed. Thus, the following individual sets exist in this model:

$\{F\}$	evidence for free space
$\{S\}$	evidence for static occupancy
$\{D\}$	evidence for dynamic occupancy
$\{S, D\}$	evidence for occupied, i.e., either static or dynamic occupancy
$\{F, S\}$	not used, always conflicting
$\{F, D\}$	not used, always conflicting
Θ	*unknown*, i.e., evidence for either free space, static occupancy, dynamic occupancy, or static-dynamic occupancy

This model has the particular advantage that dynamic and static occupancy is separated. Thus, if a sensor, or an algorithm processing the sensor data, can determine whether or not a measurement corresponds to a static or to a dynamic object, the belief mass can directly be assigned to $m(S)$ and $m(D)$ respectively. On the other hand, if a sensor cannot determine whether or not a measurement is due to a static or a dynamic object, the belief can be assigned to the superset mass $m(\{S, D\})$. Furthermore, if two sensors are to be fused and one can measure the dynamics and the other cannot, the information of both can directly be incorporated. Next, an overview of the approach, which is centered on the given environment model, is presented.

2.2.2 GTAM Overview

In this section, the main components of the grid-based tracking and mapping are introduced, and it is shown how they work together.

As the name suggests, GTAM consistently works in a grid-representation, i.e., the environment is divided into a number of cells, where each cell represents certain properties of a specific subspace of the environment. This environment representation, as used in occupancy grid mapping, has shown great utility for several reasons. First, compared to model-based representations such as bounding-boxes, no shape assumptions about the environment are made. Therefore, every possible object shape can be represented, if the grid resolution is high enough and discretization artifacts are ignored. Second, the data association problem is avoided, as it is already provided by the grid structure. Third, cell independence assumptions allow efficient operations on the grid. And fourth, due to the grid being essentially an image, algorithms and ideas from related fields, such as computer graphics and computer vision, can be used. In particular, due to general-purpose computing on graphics hardware with platforms such as Nvidia Compute Unified Device Architecture (CUDA) [122], computation-intensive operations can still be performed in real-time, which is mandatory for autonomous vehicles.

The basic entities in many grid mapping techniques, such as in this approach, are scan grids. They represent the data of the sensor measurement at a particular time in the form

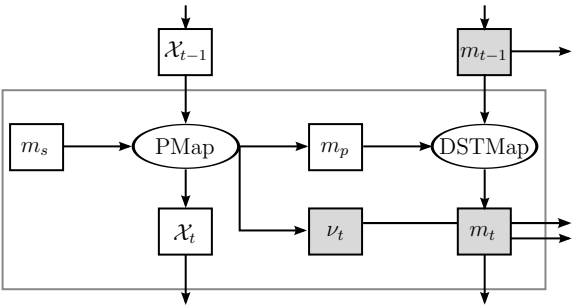


Figure 2.1: Overview of GTAM. Shaded variables are the suggested output of the system.

of a grid. In the scan grid generation process, probabilities, or evidences, for the hypotheses occupied and free are deduced based on the measurement z_t. If multiple sensors are used, their scan grids can be fused, as one way of sensor data fusion. Both, scan grid generation and fusion are touched in Section 2.2.3. The scan grid m_s uses the environment model from Section 2.2.1. Intentionally, the same letter m is used for both, grid maps and belief masses in the Dempster–Shafer theory, since the maps use the DST model presented above. Hence, the first step involves computing evidences for free and static-dynamic, i.e., $m_s(F)$ and $m_s(\{S, D\})$, which are the input for the particle map (PMap). Figure 2.1 shows the overview in form of a diagram.

The map ν_t at time t is a grid-based velocity map obtained from the particle map described in Section 2.3. In the PMap, velocity distributions are estimated using particle filters and therefore represented in the form of a set of particles $\mathcal{X}_t$. With the set $\mathcal{X}_t$, i.e., the estimated cell velocity distributions, the belief m_p is inferred that holds the evidences for static and dynamic occupancy, $m_p(S)$ and $m_p(D)$. This step is different to existing literature, which, as described above, either requires to have the distinction between static and dynamic sensor measurements already available or estimates it using binary classifiers. Here, the dynamic information, i.e., the velocities of the environment, is estimated and continuous beliefs about the static and the dynamic environment are generated.

The belief m_p is the input of the Dempster–Shafer theory map (DSTMap), described in Section 2.4, which fuses m_p with the belief m_{t-1} from time $t-1$ to generate the current belief m_t. This step filters the free space evidences from the scan grid as well as the $\{S, D\}$ and the static evidences in order to create a robust map.

2.2.3 Scan Grid Generation and Fusion

The scan grids are the fundamental elements of the filters. They are generated from the data of the sensor measurement z_t at a particular time t. Similar to occupancy grid mapping, as described in [157], inverse sensor models are used. However, instead of probabilities, belief masses for the sets $\{S, D\}$ and $\{F\}$ are derived from the sensor data, as shown in the following. It is noted, that it is assumed that the sensor data is already processed with regard to reflections from the ground.

Laser Scan Grids

Laser range finders are accurate sensors. They measure the range and the bearing to the obstacles in their field of view. In this work, 4-layer laser scanners are used for the experimental evaluations.

Different sensor models, depending on the characteristics of the scanner, as well as different scan grid generation algorithms, exist. Since the sensor takes the measurement in polar space (range and bearing) and the maps are usually updated and stored in Cartesian space, a conversion is required. In [71, 72], a polar scan grid is generated first, where each row corresponds to a laser beam. This way, a single laser beam, or more precisely the model of the laser beam, if it is assumed that the horizontal resolution of the scanner is large enough in order not to miss the smallest objects in width at the maximum distance, travels exactly through one cell at a time. The polar scan grid is then transformed into a Cartesian scan grid using texture mapping, i.e., the process of coloring 3D objects using 2D images of the texture of the objects, known from computer graphics [69].

An alternative approach is to directly calculate the scan grid as a Cartesian grid. Here, a model similar to the inverse range sensor model from [157] is used. Since the operations are independent for every grid cell, instead of relying on some sort of raycasting, it performs well on parallel hardware, such as GPUs. First, all four layers are projected onto the same plane, since the vertical field of view is small. Let the index j iterate over all bearings of the measurement z_t taken from position x_t, which pass through or overlap the current cell i, which itself has the position x^i. And let $j' = \arg\min_j z_t^j$ be the index of the one bearing passing through cell i with the smallest range measurement. If the cell i is out of the sensor range, then j' is -1. The scan grid belief mass for occupied

$$m_s^i(\{S, D\}) = \begin{cases} \max\limits_j \; m_{\text{occ}} \exp\left(-\dfrac{\left(||x^i - x_t|| - z_t^j\right)^2}{2\sigma^2}\right) & \text{if } j' \geq 0 \\ 0 & \text{else} \end{cases} \tag{2.7}$$

is calculated using a Gaussian distribution with maximum occupancy evidence m_{occ}. The belief mass for free, with maximum evidence m_{free}, for cell i is calculated according to:

$$m_s^i(F) = \begin{cases} \max\big(m_{\text{free}} - m_s^i(\{S, D\}), 0\big) & \text{if } j' \geq 0 \wedge ||x^i - x_t|| < z_t^{j'} \\ 0 & \text{else} \end{cases} \tag{2.8}$$

For completeness, the belief mass for unknown

$$m_s^i(\Theta) = 1 - m_s^i(\{S, D\}) - m_s^i(F) \tag{2.9}$$

is simply the residual uncertainty. Figure 2.2 shows the laser scan points and the scan grid belief masses in two scenarios. The green channel denotes the free evidence, and purple, i.e., the red and the blue channel, denote the evidence for $\{S, D\}$.

Radar Scan Grids

Similar to laser scanners, radar sensors also yield the range and the bearing to the targets, although the measuring principle is quite different. Radar sensors are less accurate and unlike laser scanner data, which is usually modeled as point cloud, radar sensor data is often obtained as a list containing the n strongest reflections or as a polar image. The latter is what is received by the radar sensor used here.

The polar amplitude image can already be used as occupied mass of the scan grid. It is transformed into a Cartesian grid using texture mapping as done in [71] and mentioned before. Free space evidences still need to be calculated, though. Sometimes, the occupancy probability, which includes the probability of free space, is directly calculated from the reflection intensity [31, 72]. Hence, every cell that is above the noise level corresponds to an occupancy probability > 0.5 and every cell below the noise level to a probability < 0.5. However, with this model, even cells that lie behind objects, where the radar signal does not pass through, such as walls, result in free space, although they can actually not be observed. Therefore, an approach similar to what is used for laser range finders is chosen. In every row of the polar reflection image, it is searched for the first obstacle, and free space is assumed up to this cell. Additionally, the free space evidences decrease with increasing distance to the sensor to model the decreasing likelihood for detecting a target. This is different to the model for laser scanners. The measurement ray of a lidar cannot pass through obstacles, ignoring glassy objects for the applications in this work, and therefore free space up to the reflection is equally likely, independent of how far the obstacle is away. Similar to the examples with laser scanner data, Figure 2.2 shows radar scan grids in the same scenarios.

Scan Grid Fusion

The information of individual sensors can be fused to increase robustness and to increase the field of view. Sensor data fusion will, however, only be briefly touched as it is not topic of this thesis. Sensor data can be fused at different levels [80]. In the context of grid-based representations they are the raw data level, the scan grid level, and the map grid level. Fusion at scan grid level has the advantage that the raw sensor data is already put into the same format, i.e., the grid structure, while the data is still raw in that it is not filtered, as is the case for map grids. Additionally, the computational overhead is kept low, since the data is fused before updating the map and therefore only one map needs to be managed.

However, unlike scan grids that only hold the static environment, the scans in this thesis also hold the dynamic environment. Therefore, they can only be fused if the sensors are synchronized, i.e., if the sensors take the measurements at the same time in order to capture the same state of the environment. Three simple cell-level fusion methods are given in the following, which combine the cells that correspond to the same location of the world independently of neighboring cells.

Let the index j iterate over all scan grids. The most defensive approach is to fuse the scan grids according to the maximum occupancy

$$\begin{aligned} j' &= \arg\max_j m^i_{s_j}(\{S, D\}) \\ \underset{A\subseteq\Theta}{m^i_s(A)} &= m^i_{s_{j'}}(A) \end{aligned} \tag{2.10}$$

and to choose the free space evidence correspondingly. This way, if an object is detected by any sensor, it definitely is in the fused scan grid, as is any noise, however. In terms of accuracy of the location of the objects, the output corresponds to the accuracy of the least-accurate sensor, at least at the locations that are measured by it. In general, a grid is not an ideal structure for fusion in terms of location accuracy. Other than the combination of Gaussian distributions, such as in a Kalman filter, where the resulting variance will be smaller than, or in the worst case equal to, the variances of the individual distributions [157], in a grid, since every cell is treated independently, this is not the case. The fused result can never be more accurate than the accuracy of the best sensor with standard fusion techniques.

An alternative way to fuse scan grids is by taking the maximum evidence of either occupied or free, i.e.,

$$\begin{aligned} j' &= \arg\max_j \left[\max_{A\in\{F,\{S,D\}\}} m^i_{s_j}(A) \right] \\ \underset{A\subseteq\Theta}{m^i_s(A)} &= m^i_{s_{j'}}(A). \end{aligned} \tag{2.11}$$

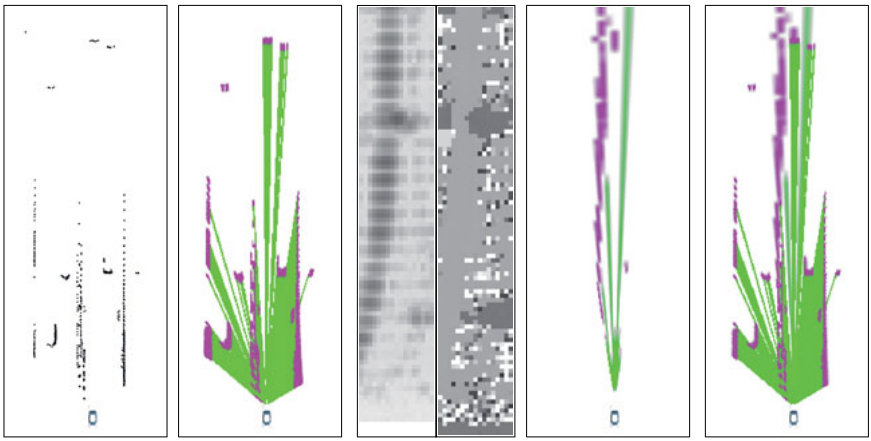
If the free space evidences of the accurate sensors are designed to be higher than the occupancy evidences of the less accurate sensors, then at the non-occluded parts of the objects, the accuracy in the fused result is retained.

In between is a combination of the belief masses using combination rules, such as the Dempster rule of combination

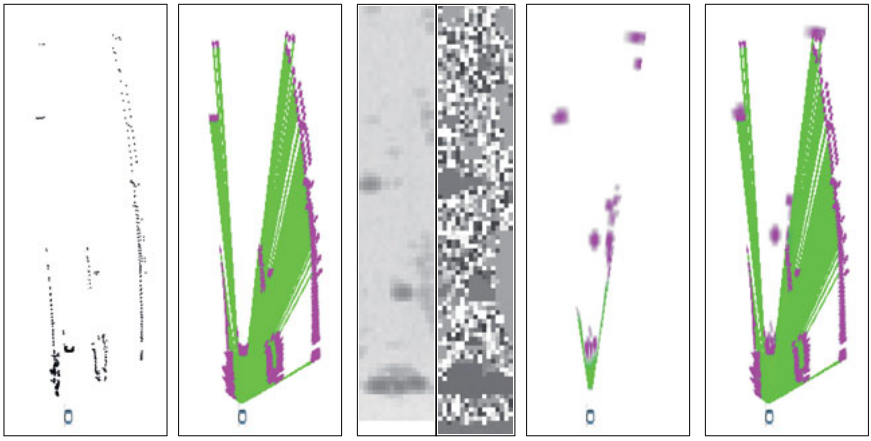
$$\begin{aligned} m^i_s(\{S, D\}) &= m^i_{s_1}(\{S, D\}) \oplus^{\mathrm{D}} \cdots \oplus^{\mathrm{D}} m^i_{s_n}(\{S, D\}) \\ m^i_s(F) &= m^i_{s_1}(F) \oplus^{\mathrm{D}} \cdots \oplus^{\mathrm{D}} m^i_{s_n}(F), \end{aligned} \tag{2.12}$$

which derive a new belief based on the mass evidences given by each sensor.

Figure 2.2 shows results of the scan grid fusion using Dempster's rule of combination. The upper row in Figure 2.2 depicts a situation, in which the radar sensor receives a high amount of reflections from a guardrail to the left of the ego vehicle, which cannot be perceived that far by the laser scanner. Although the radar sensor has a high uncertainty in its location, due to the combination, the strong free space evidence of the laser scanner reduces the extent of the radar occupied masses, which are weaker at the border of the reflections. The lower row shows a scenario, in which the radar sensor detects a moving object, which is occluded in the laser scan by another vehicle.



(a) Laser points. **(b)** Laser scan grid. **(c)** Radar intensity, Doppler (polar). **(d)** Radar scan grid. **(e)** Dempster fusion.



(f) Laser points. **(g)** Laser scan grid. **(h)** Radar intensity, Doppler (polar). **(i)** Radar scan grid. **(j)** Dempster fusion.

(k) Camera image of upper scenario. **(l)** Camera image of lower scenario.

Figure 2.2: Scan grid fusion in two different scenarios, upper and middle row.

Grid Coordinate Systems

The filtered map grids are of constant size. Therefore, if the robot is moving, two grids from two different time steps have to be aligned so that the information about the same world locations is combined. Details about the coordinate systems can be found in Appendix A.3. Note that all velocities in this chapter are absolute, i.e., not relative to the ego velocity.

2.3 The Particle Map

In this section, the particle map (PMap) is presented. It is a grid-based velocity estimator. The particle map was inspired by ideas from [35]. This work, however, presents a different formulation and a different underlying model, which is presented in Section 2.3.1. The main components such as sampling, Section 2.3.2, weighting and resampling, Section 2.3.3, are done in different ways, as highlighted in Section 2.1.2. The approach is given and demonstrated with a laser scanner and a radar sensor. The Dempster–Shafer belief mass derivation, which is input to the DSTMap presented in the next section, is described in Section 2.3.4.

2.3.1 Estimating Cell Velocity Distributions using Particle Filters

The particle map is a velocity map ν_t that represents velocity distributions at particular time instances t. Similar to occupancy grid mapping, the goal is estimate the posterior over all maps

$$p(\nu_t \mid z_{1:t}, x_{1:t}) \tag{2.13}$$

given the measurements $z_{1:t}$ and the robot poses $x_{1:t}$. Due to the grid representation the map is represented by a set of cells $\nu_t = \{\nu_t^i\}$, where every cell has attached to it a random vector $V = \begin{pmatrix} V_x & V_y \end{pmatrix}^{\mathrm{T}}$ in order to represent two-dimensional velocity distributions. Although the cells contain velocity distributions, i.e., dynamic information, they do not move in space, but the grid is a fixed discretization of the world.

As in occupancy grid mapping, the posterior from (2.13) has an intractable dimensionality. Similarly, the problem is simplified by assuming that the cells are independent within a single time instance t allowing the posterior to be approximated as the product of its marginals

$$p(\nu_t \mid z_{1:t}, x_{1:t}) = \prod_i p(\nu_t^i \mid z_{1:t}, x_{1:t}). \tag{2.14}$$

Differently to occupancy grid mapping, a time index t is introduced due to a drop of the static world assumption and the cell independence assumption is only valid within t, since dynamic objects are expected to move over multiple cells over time. Put differently,

it is assumed that the velocity distribution $p(\nu_t^i \mid z_{1:t}, x_{1:t})$ of cell i is independent of $p(\nu_t^j \mid z_{1:t}, x_{1:t})$ of cell j at time t. Between different time steps, however,

$$\begin{aligned} p(\nu_t^i \mid z_{1:t}, x_{1:t}) &= \int p(\nu_t^i \mid \nu_{t-1}, z_{1:t}, x_{1:t})\, p(\nu_{t-1} \mid z_{1:t-1}, x_{1:t-1})\, \mathrm{d}\nu_{t-1} \\ &= \int p(\nu_t^i \mid \nu_{t-1}, z_{1:t}, x_{1:t}) \prod_i p(\nu_{t-1}^i \mid z_{1:t-1}, x_{1:t-1})\, \mathrm{d}\nu_{t-1} \end{aligned} \tag{2.15}$$

the full map posterior is used.

Now, to estimate the velocity posterior from (2.13), particle filters are used. Particle filters, as described in [157], are a non-parametric version of the Bayes filter. In a particle filter, the posterior probability distribution is represented by a finite set of particles $\mathcal{X}_t = \{\chi_{t,[k]}\}$, instead of, e.g., a Gaussian distribution such as in a Kalman filter. Therefore, arbitrary multimodal distributions can be represented. Particle filters have proven to be very efficient in low-dimensional problems, such as in the localization problem and they have also been used for Simultaneous Localization And Mapping (SLAM) [111].

Due to the independence assumption from (2.14), the particles represent low-dimensional velocity hypothesis, i.e., a $v = \begin{pmatrix} v_x & v_y \end{pmatrix}^{\mathrm{T}}$ of a particular cell i, rather than full maps. Each particle is, at a given time t, at one particular continuous position in the grid. The particles themselves are 4-D vectors

$$\chi_{[k]} = \begin{pmatrix} x_{x,[k]} & x_{y,[k]} & v_{x,[k]} & v_{y,[k]} \end{pmatrix}. \tag{2.16}$$

It will be referred to $v_{[k]} = \begin{pmatrix} v_{x,[k]} & v_{y,[k]} \end{pmatrix}^{\mathrm{T}}$ as the velocity component of $\chi_{[k]}$, and to $x_{[k]} = \begin{pmatrix} x_{x,[k]} & x_{y,[k]} \end{pmatrix}^{\mathrm{T}}$ as its position.

Between consecutive time steps the particles move according to a motion vector and a certain motion model through the grid. They are not fixed to the cells, in which they are created. However, at one particular time instance, every particle does belong to one particular cell. It is then one particular hypothesis of the velocity distribution of that cell. The map posterior at time t

$$p(\nu_t \mid z_{1:t}, x_{1:t}) = \prod_i p(\nu_t^i \mid z_{1:t}, x_{1:t}) = \prod_i \sum_{\chi_{t,[k]} \in \text{cell } i} w_{[k]}\, \delta\big(\nu_t^i; v_{t,[k]}\big) \tag{2.17}$$

is therefore represented by a product of cell velocity distributions, which are in turn represented by a sum of weighted Dirac delta distributions δ that are the particles.

In the following sections, the three main components of the particle filter, i.e., sampling, weighting and resampling are given.

2.3.2 Particle Creation and Sampling

Sampling is the process of recursively generating a new set of particles $\overline{\mathcal{X}}_t$, i.e., a new set of hypothetical states, based on the set of particles $\mathcal{X}_{t-1}$. In many robotic applications, such as in this work, this step involves moving the old set of particles according to their

state hypotheses and a certain motion model. Formally, the new particle set is sampled from

$$\overline{\mathcal{X}}_t \sim p(\nu_t \mid \mathcal{X}_{t-1}). \tag{2.18}$$

Since the velocities are represented in global grid coordinates, as described in Appendix A.3, the ego velocity is not needed in this process.

The particles are moved according to a simple constant velocity model

$$\begin{aligned} x_{t,[k]} &= x_{t-1,[k]} + \Delta t\, v_{t-1,[k]} + \mathcal{N}\big(0, \mathrm{diag}\big(\sigma_{n_x}^2, \sigma_{n_x}^2\big)\big) \\ v_{t,[k]} &= v_{t-1,[k]} + \mathcal{N}\big(0, \mathrm{diag}\big(\sigma_{n_v}^2, \sigma_{n_v}^2\big)\big) \end{aligned} \tag{2.19}$$

with added Gaussian noise for the position, having standard deviation σ_{n_x}, and for the velocity, having standard deviation σ_{n_v}.

Initial Position Sampling

Initially, the particles need to be created so that they can be moved, weighted, and resampled. Typically, as in Monte-Carlo localization, the initial particle distribution is uniform throughout the state space. Applying this strategy, though, means to sample the entire 4-D space, requiring a high number of particles. In addition, it is expected that once the particles have converged towards the current scene and a fixed number of particles is used, newly-appearing objects are hard to detect. This is known as the particle deprivation problem [157], since for a particle filter it is important to keep the particles spread out evenly and densely so that there actually are particles in the vicinity of the true states.

In areas, where no measurements occur, no velocities can be inferred and usually most of the cells from a grid map are either free-space, occluded-space or otherwise not-observable-space. Therefore, particles are only created in those cells, initially placed in the cell center, where measurements occur. Previously created particles are, however, allowed to exist even in areas where there is currently no measurement, in order to compensate for missed detections and occlusion, as described in Section 2.3.3. Note that the generation of particles, as well as their deletion through the weighted resampling, is a continuous process. Whenever the mass $m_s(\{S, D\})$ from the scan grid is greater than 0 in a particular cell, and no particles from the previous time step landed in that cell, new particles are created. The number of particles that are created in the cell i equals

$$\big|\mathcal{X}_0^i\big| = \big\lfloor n_\chi^{i,\max}\, m_s^i(\{S, D\})\big\rfloor, \tag{2.20}$$

where $n_\chi^{i,\max}$ is the maximum number of particles in a cell. Hence, newly-appearing objects can quickly be detected. Additionally, the total number of particles is not fixed but adapts according to the density of the scene guaranteeing that there are always enough particles to estimate the dynamics of the objects in the environment, while at the same time keeping the total number of particles low.

Initial Velocity Sampling

Apart from sampling the position, i.e., the cells in which new particles are created, the velocity distribution from which the particles are drawn is also of major importance for the performance of the estimator and for the world that can be modeled. In [35], uniform sampling from the 2-D space of velocities is used. The following is however observed.

Observation 2.1 *With uniform velocity sampling it is impossible to exactly represent the static environment, since the probability*

$$P\big(V = \begin{pmatrix}0 & 0\end{pmatrix}^{\mathrm{T}}\big) = 0 \tag{2.21}$$

of sampling an exact static particle is zero.

Therefore, it is proposed to add a Dirac delta distribution centered at $\begin{pmatrix}0 & 0\end{pmatrix}^{\mathrm{T}}$ to the uniform distribution $\mathcal{U}$ and to sample from

$$p(\nu_0^i \mid z_0^i) = w_D\, \mathcal{U}\left(\begin{pmatrix}-v_{\max}\\ -v_{\max}\end{pmatrix}, \begin{pmatrix}v_{\max}\\ v_{\max}\end{pmatrix}\right) + w_S\, \delta \begin{pmatrix}0\\0\end{pmatrix}, \tag{2.22}$$

where $w_D + w_S = 1$ represent the priors about the amount of dynamic and static cells respectively. With uniform sampling alone, the particles will never converge at static obstacles. The filter is continuously generating particles with random non-zero velocities that eventually move out of the cells corresponding to the static objects. This is demonstrated later in Chapter 6.

It is noted that the introduction of pure static particles comes at a price, since less particles are available for sampling the velocity of dynamic objects. However, on the one hand, for creating a map of the environment, it is even more important to robustly determine whether or not a cell is static or dynamic, than additional precision in the estimated velocity. And on the other hand, static particles do not need to be moved. They therefore stay where they are created, do not need to be re-mapped onto the new cells, and can also be stored efficiently, since per cell they are all equal.

If a sensor is used, that is able to measure velocities or parts of it, this information can be used during initial sampling.

Initial Velocity Sampling with a Radar Sensor

A radar sensor, compared to, e.g., a laser scanner, is able to measure the radial component of the velocity of the dynamic objects relative to the ego velocity. The radial velocity is the 1-D projection of the true velocity vector of the measured point in space onto the line that connects the sensor and the measurement. It provides a good estimate of the true velocity in cases, where an object moves parallel to the sensor axis, but gives less to none information about the true velocity for an object moving perpendicular to the sensor axis. Note, the measurement of a perpendicular moving point in space relative to the line connecting the sensor and the measurement is 0.

The radial velocity measurement z_{v_R} of the radar is only used, if the intensity, i.e., the reflectional component of the measurement is above a certain threshold. Otherwise, the corresponding velocity measurement cannot be trusted and initial sampling is done according to (2.22). If the intensity is above the threshold, the initial velocity sampling distribution of cell i is calculated according to

$$p(\nu_0^i \mid z_0^i) = p(\nu_0^i \mid z_{v_R}^i) = \begin{cases} (1-w_\varepsilon)\, R_\alpha \mathcal{N}\left(\begin{pmatrix} z_{v_R}^i \\ 0 \end{pmatrix}, \begin{pmatrix} \sigma_{v_R}^2 & 0 \\ 0 & \sigma_{v_T}^2(z_{v_R}^i) \end{pmatrix}\right) + w_\varepsilon\, \delta \begin{pmatrix} 0 \\ 0 \end{pmatrix} & \text{if } |z_{v_R}^i| > \varepsilon \\ w_D\, R_\alpha\, \eta \begin{pmatrix} \mathcal{N}(z_{v_R}^i, \sigma_{v_R}^2) \\ \mathcal{U}(-v_{\max}, v_{\max}) \end{pmatrix} + w_S\, \delta \begin{pmatrix} 0 \\ 0 \end{pmatrix} & \text{else} \end{cases} \tag{2.23}$$

with normalizer

$$\eta = \left(\iint_{[-\infty,\infty]\times[-v_{\max},v_{\max}]} \begin{pmatrix} \mathcal{N}(z_{v_R}^i, \sigma_{v_R}^2) \\ \mathcal{U}(-v_{\max}, v_{\max}) \end{pmatrix} \mathrm{d}v_R\, \mathrm{d}v_T \right)^{-1} \tag{2.24}$$

and explained in the following.

The accuracy of the sensor in measuring the correct radial velocity is modeled as Gaussian distribution with mean $z_{v_R}^i$ and standard deviation σ_{v_R}. It is assumed that the ego velocity is already compensated, i.e., that z_{v_R} holds the absolute values. Due to the measurement of the radial component only, there are infinitely many possible velocity vectors. Figure 2.3 depicts the situation and shows the line $v \cdot (z_x/||z_x||) = z_{v_R}$ of possible corresponding velocity vectors v according to the position of the measurement z_x. To ease notation it is assumed that the sensor is in the origin of the coordinate system.

If the measured radial velocity is smaller in absolute value than ε, the distribution is modeled similar to (2.22) with a static part, weighted by w_S, and a dynamic part, weighted by w_D. The radial component is, however, Gaussian distributed according to the measurement. Since nothing is known about the tangential velocity, it is modeled as uniform distribution. R_α denotes a rotation matrix of α degrees, where α is the angle between the x-axis and z_x.

The situation is similar, however slightly different, when the measured radial velocity is above a certain threshold. The larger the measured radial velocity z_{v_R}, the less likely it is that the norm of the true velocity $||v^*||$ strongly deviates from $|z_{v_R}|$ ignoring sensor inaccuracies and failures. This is motivated due to the fact that $|v_R^*|$ is always an underapproximation of $||v^*||$, and the proportional deviation between $|v_R^*|$ and $||v^*||$ reflects the proportional increase of $||v^*||$ over $|v_R^*|$. If $|v_R^*|$ is already high, an assumed maximum speed limits the distribution of velocities in the tangential direction. Using the same argument, the higher the measured radial velocity, the more likely it is that the measured point in space moves in a similar direction as the ego vehicle. Therefore, the standard deviation in the tangential direction

$$\sigma_{v_T}(z_{v_R}^i) = a\, e^{-b\,|z_{v_R}^i|} \tag{2.25}$$

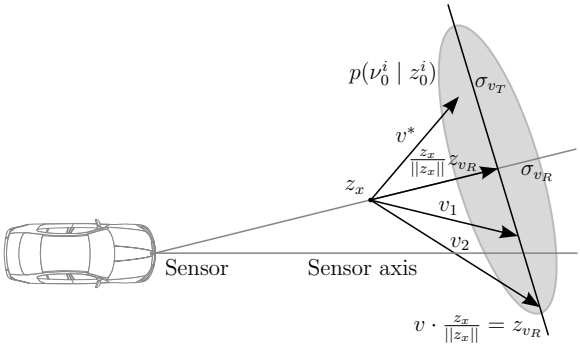


Figure 2.3: Radar sensor model used for the velocity component in particle creation.

is modeled inversely proportional to the measured radial velocity using an exponential function.

Due to resolution inaccuracies of radar sensors, compared to laser scanners, a small fraction of static particles is added, even if a high radial velocity together with a high reflection intensity is measured. In the experiments, it was observed that the reflections of moving vehicles are sometimes larger and more intense than the reflections of the static environment, which lead to the creation of only dynamic particles in cells containing static obstacles, such as road boundaries next to dynamic objects. Next, the weighting and resampling steps are given.

2.3.3 Particle Weighting and Resampling

The weighing and resampling in particle filters are the mechanisms of how the filter converges to its estimate. In every step, each particle gets assigned a weight, which represents how well that particular particle fits the data. This weight is then used to generate the new particle population, where the probability of drawing a particle corresponds to its weight. Resampling is also known as *importance sampling*. The aim is to approximate a target distribution p_{target}. Often, it cannot be sampled from p_{target} directly, but only from a different probability density function p_{proposal}, the proposal distribution. For the proposal distribution, the following must hold

$$p_{\text{target}} > 0 \quad \rightarrow \quad p_{\text{proposal}} > 0 \tag{2.26}$$

To account for the difference in the distribution from which is sampled and the one that is estimated, each particle $\chi_{[k]}$ is weighted according to

$$w_{[k]} = \frac{p_{\text{target}}\big(\chi_{[k]}\big)}{p_{\text{proposal}}\big(\chi_{[k]}\big)} \tag{2.27}$$

Dividing the target distribution for the presented particle filter

$$p_{\text{target}} = p(\nu_{1:t} \mid z_{1:t}, x_{1:t}) \tag{2.28}$$

by the proposal distribution

$$p_{\text{proposal}} = p(\nu_{1:t} \mid z_{1:t-1}, x_{1:t-1}) = \underbrace{p(\nu_t \mid \nu_{t-1})}_{\overline{\mathcal{X}}_t \text{ from } \mathcal{X}_{t-1}} \underbrace{p(\nu_{1:t-1} \mid z_{1:t-1}, x_{1:t-1})}_{\mathcal{X}_{t-1}}, \text{ i.e.,} \tag{2.29}$$

$$\begin{aligned}
\frac{p_{\text{target}}}{p_{\text{proposal}}} &= \frac{p(\nu_{1:t} \mid z_{1:t}, x_{1:t})}{p(\nu_t \mid \nu_{t-1})\, p(\nu_{1:t-1} \mid z_{1:t-1}, x_{1:t-1})} \\
&\overset{\text{Bayes}}{=} \eta \frac{p(z_t \mid \nu_{1:t}, z_{1:t-1}, x_{1:t})\, p(\nu_{1:t} \mid z_{1:t-1}, x_{1:t})}{p(\nu_t \mid \nu_{t-1})\, p(\nu_{1:t-1} \mid z_{1:t-1}, x_{1:t-1})} \\
&= \eta \frac{p(z_t \mid \nu_{1:t}, z_{1:t-1}, x_{1:t})\, p(\nu_t \mid \nu_{t-1}, z_{1:t-1}, x_{1:t})\, p(\nu_{1:t-1} \mid z_{1:t-1}, x_{1:t})}{p(\nu_t \mid \nu_{t-1})\, p(\nu_{1:t-1} \mid z_{1:t-1}, x_{1:t-1})} \\
&\overset{\text{Markov}}{=} \eta \frac{p(z_t \mid \nu_t, x_t)\, p(\nu_t \mid \nu_{t-1})\, p(\nu_{1:t-1} \mid z_{1:t-1}, x_{1:t-1})}{p(\nu_t \mid \nu_{t-1})\, p(\nu_{1:t-1} \mid z_{1:t-1}, x_{1:t-1})} \\
&= \eta\, p(z_t \mid \nu_t, x_t)
\end{aligned} \tag{2.30}$$

yields the measurement model. Assuming independence, the weight of particle $\chi^i_{t,[k]}$, associated to cell i at time t, therefore corresponds to

$$w_{t,[k]} = \eta\, p(z^i_t \mid \chi^i_{t,[k]}, x_t). \tag{2.31}$$

Cell-based Weighting

As discussed before, velocities cannot be measured directly with many sensors, such as laser scanners, and hence it cannot be decided whether one particular sample, within one time instance, fits the data better than any other sample in the same cell. However, since the particles are allowed to move on the grid over time, they can still be weighted on a cell level. A particle that lands in cells which have high associated cell weights, at the times the particle was in each of those cells, fits the real world dynamics well and also has a high chance of being drawn in the resample procedure. On the other hand, if a particle moves through cells with low associated weights, it has a low probability of being drawn.

The weight of cell i

$$w^i_t = \sum_{\chi^i_{t,[k]}} w^i_{t,[k]} = n^i_\chi\, w^i_{t,[k]} = \eta\, p(z^i_t \mid \overline{\mathcal{X}}^i_t, x_t) \tag{2.32}$$

equals the sum of the weights of all particles in the cell and corresponds to the probability of the measurement z^i_t given the set of particles in that cell $\overline{\mathcal{X}}^i_t$. The weight is equal for all n^i_χ particles. Since the actual particles in the cell, i.e., the velocity hypotheses, are

irrelevant for the measurement, only the number of particles $\left|\overline{\mathcal{X}}_t^i\right|$ is of interest. Due to the random initial generation of particles, the number of particles that reached cell i from the previous time step, does not say anything about how good the velocity hypotheses of the particles are. Even one particle may *guess* the true velocity correctly and is also a hypothesis for occupied. Similar to the initial particle generation from (2.20), where the number of created particles corresponds to the evidence of $\{S, D\}$, the cell weight

$$w_t^i \propto m_s^i(\{S, D\}) \tag{2.33}$$

is therefore proportional to the scan grid mass of $\{S, D\}$ of cell i and defines the desired number of particles in the cell after resampling, $n_\chi^{i,\text{des}}$.

Shown above, and differently to other particle filter applications, the number of particles is not constant, but continuously adapted through particle creation and cell-based weighting and resampling. If the number of particles needs to be reduced according to the predicted set of particles and the cell weight, particles undergo a particle deletion filtering. Otherwise, a single missed detection or occlusion in a single frame leads to an elimination of all particles.

If $n_\chi^{i,\text{des}} < \left|\overline{\mathcal{X}}_t^i\right|$, and the particle $\chi_{t,[k]}^i$ associated to cell i is to be removed, which is equivalent to not being drawn by the resampling step, it still has a probability of surviving

$$p_{\text{surv}}\big(\chi_{t,[k]}^i\big) = \max\big(p_{\text{surv}}^{\max} - m_s^i(F), p_{\text{surv}}^{\min}\big) \tag{2.34}$$

controlled by a maximum survival probability $p_{\text{surv}}^{\max}$, and a minimum survival probability $p_{\text{surv}}^{\min}$. The probability p_{surv} is inversely proportional to the measured free space, since it is likely that particles that move into measured free space, correspond to random noise and need to be eliminated. On the other hand, in areas that are occluded in the current scan, the survival probability is higher, in order to continue the tracking.

Intra-cell Weighting with a Radar Sensor

Similar to particle generation, with a radar sensor, the particles can be weighted directly according to the measured radial velocity. The weight

$$w_{t,[k]} \propto \mathcal{N}\big(z_{v_R}^i; v_{R,[k]}, \sigma_{v_R}^2\big) \tag{2.35}$$

of particle $\chi_{[k]}$ is modeled proportional to a Gaussian distribution with mean

$$v_{R,[k]} = \text{proj}_x\big(R_{-\alpha}\, v_{[k]}\big), \tag{2.36}$$

standard deviation σ_{v_R} and evaluated at the measured radial velocity $z_{v_R}^i$. Note, that the intra-cell weighting is done in addition to the cell-based weighting, i.e., the number of particles is still controlled with the cell weights w_t^i. This time, however, the weights of the particles are not equal but calculated according to (2.35).

After having calculated the weight of each particle, resampling is applied using the low variance sampler described in [157]. It exhibits the desired property of maintaining the same particle population if all weights are equal. Now that the particle filter based velocity

estimation is given, it is shown next, how evidences for the masses of static and dynamic are derived.

2.3.4 Belief Mass Derivation

In the previous sections, the velocity estimator is presented. It is a grid-based tracker that works on the cell level and estimates, for each cell, a velocity probability distribution. It already provides useful information about the environment and may fully or partially replace classical model-based trackers. For computing maps of the static world in dynamic environments, a classification between static and dynamic on the cell level is needed.

In [35], this classification is binary and based on the particles. In particular, a cell is static, if the absolute values of both components of the mean velocity are lower than twice the standard deviation of the velocities of all particles in the cell. Otherwise, the cell is dynamic. Intuitively, those cells are declared static, whose velocity distributions, starting from uniform distributions, either failed to converge or have not yet converged to peak distributions. They are detected indirectly, as where the particle filter fails to capture the environment.

Since here, pure static particles are sampled, in addition to uniform velocity sampling, the static as well as the dynamic particles can directly be detected based on their velocity. Let $\mathcal{X}^i = \mathcal{X}_S^i \cup \mathcal{X}_D^i$ be the set of all particles of a particular cell that have survived at least t steps, in order to incorporate the fact that velocities are estimated indirectly over multiple time instances. Then

$$\begin{aligned} \mathcal{X}_S^i &= \left\{ \chi_{[k]} \mid \chi_{[k]} \in \mathcal{X}^i \ \wedge \ ||v_{[k]}|| \leq \varepsilon \right\} \\ \mathcal{X}_D^i &= \left\{ \chi_{[k]} \mid \chi_{[k]} \in \mathcal{X}^i \ \wedge \ ||v_{[k]}|| > \varepsilon \right\}. \end{aligned} \tag{2.37}$$

Contrary to a binary classification, it is aimed for a continuous evidential classification using the environment model described in Section 2.2.1. The evidential belief masses

$$\begin{aligned} m_p^i(\{S\}) &= \frac{\left|\mathcal{X}_S^i\right|}{n_\chi^{i,\max}} \\ m_p^i(\{D\}) &= \left(1 - \frac{\sigma^{\text{orient}}}{\sigma_{\max}^{\text{orient}}}\right) \frac{\left|\mathcal{X}_D^i\right|}{n_\chi^{i,\max}} \\ m_p^i(\{F\}) &= \min\left(m_s^i(\{F\}),\ 1 - m_p^i(\{S\}) - m_p^i(\{D\})\right) \\ m_p^i(\{S,D\}) &= \max\left(0,\ m_s^i(\{S,D\}) - m_p^i(\{S\}) - m_p^i(\{D\})\right) \\ m_p^i(\Theta) &= 1 - \sum_{A \subset \Theta} m_p^i(A) \end{aligned} \tag{2.38}$$

are calculated based on the set cardinality of the corresponding subset of $\mathcal{X}^i$ and based on the scan grid m_s. The ratio of the dynamic particles in the belief mass calculation for the set D is weighted with the inverse of the standard deviation of the orientation of all dynamic particles in the cell. This is similar to the observation from above, in that

a uniform distribution does not support the hypothesis dynamic, since it corresponds to the initial sampling distribution. The mass for free is directly taken from the scan grid. However, due to the non-zero particle survival probability from (2.34), it may need to be adapted according to the masses $m_p(S)$ and $m_p(D)$. The mass $m_p(\{S, D\})$ corresponds to the residual mass of the scan grid, $m_s(\{S, D\})$, after subtracting $m_p(S)$ and $m_p(D)$. Intuitively, since $m_s(\{S, D\})$, which is the scan evidence for occupied, controls the number of particles in the cells $\left|\mathcal{X}^i\right| = \left|\mathcal{X}^i_S \cup \mathcal{X}^i_D\right|$, its mass is partially transferred to the masses of its subsets, i.e., the evidence is distributed to the more specific sets S and D. Without particle survival probability and according to (2.20) and (2.33), it can be observed that $m_p(S) + m_p(D) \leq m_s(\{S, D\})$.

Results of the particle map are shown in Section 2.5, and an evaluation and a comparison to the approach of [35] can be found in Chapter 6. Next, it is shown how the derived beliefs are filtered over time to yield a robust environment model.

2.4 The Dempster–Shafer Theory Map

In the previous section, the cell-based velocity estimator is presented, and it is shown how evidences for static and dynamic are calculated using the particle-based velocity distributions. Nevertheless, the estimated evidences may still contain noise, since they are not filtered yet, as done in occupancy grid mapping. Every measurement directly leads to the creation of new random particles. Additionally, since the particles die out if no measurements support them, evidences for static obstacles from previous observations, which are now occluded or out of the sensor range, are lost. Similarly, free space evidences are also not filtered yet. A robust environment model is important in order to be used for demanding applications such as path or trajectory planning.

Section 2.4.1 describes the temporal filtering for the Dempster–Shafer Theory Map (DSTMap) and Section 2.4.2 discusses how the DST model can be reduced to a standard probabilistic occupancy grid of the static environment.

2.4.1 Filtering over Time

The filtered map at time t is calculated from the estimation of the previous time step, m_{t-1}, and the current estimated belief from the particle map, m_p. All evidences are filtered cell-wise, i.e., the evidences at the same locations in the world are combined. Since dynamic objects move, however, this is not possible for the dynamic environment. In fact, the dynamic world is already estimated and tracked in the particle map and, since the belief m_p is input to the filter in every update step, as shown in Figure 2.1, this information is directly used.

Therefore, the dynamic mass of m_{t-1} is moved to the mass of free first, i.e.,

$$\begin{aligned} m'_{t-1}(F) &= m_{t-1}(F) + m_{t-1}(D) \\ m'_{t-1}(D) &= 0 \\ \underset{A\subset\Theta\setminus\{F,D,\emptyset\}}{m'_{t-1}(A)} &= m_{t-1}(A), \end{aligned} \tag{2.39}$$

since dynamic objects move over free space, or more precisely free-from-static space, and previous dynamic masses are not filtered in the DST map. Due to this, evidences for free space not only come from the scan grid but also from previous dynamic evidences. It is interesting to observe, that free space evidences can thus be obtained in occluded areas, if a tracked object moves over these areas.

Then, using the conjunctive rule of combination [143, 144], which is the non-normalized version of Dempster's rule of combination from (2.4),

$$m_1(A) \oplus^{\mathrm{C}} m_2(A) = \sum_{B\cap C=A} m_1(B)\, m_2(C) \tag{2.40}$$

the belief masses are updated according to

$$\begin{aligned} \underset{A\subset\Theta\setminus\{D,\emptyset\}}{m_t(A)} &= \eta\big(m'_{t-1}(A) \oplus^{\mathrm{C}} m_p(A)\big) \\ m_t(D) &= \eta\big(m'_{t-1}(D) \oplus^{\mathrm{C}} m_p(D)\big) + m'_{t-1}(F)\, m_p(D) \\ m_t(\Theta) &= \eta\big(m'_{t-1}(\Theta) \oplus^{\mathrm{C}} m_p(\Theta)\big) \end{aligned} \tag{2.41}$$

with the normalization factor

$$\eta = \begin{cases} \dfrac{1-\vartheta_{\min}}{1-c-m'_{t-1}(\Theta)\, m_p(\Theta)} & \text{if } \dfrac{m'_{t-1}(\Theta)\, m_p(\Theta)}{1-c} < \vartheta_{\min} \\ \dfrac{1}{1-c} & \text{else} \end{cases} \tag{2.42}$$

and the conflict

$$c = \left(\sum_{B\cap C=\emptyset} m'_{t-1}(B)\, m_p(C)\right) - m'_{t-1}(F)\, m_p(D). \tag{2.43}$$

In its essence, the combination from (2.41) is Dempster's rule with two modifications. First, there is a particular term in the sum of the products of the non-intersecting sets that forms the conflict, i.e., $m'_{t-1}(F)\, m_p(D)$. In the presented model, the conflict between the evidence for free space from the previous time step and the current evidence for dynamic is intentional. Similar to (2.39), where previous dynamic evidences are moved to the mass of F, the conflict between the previous free space evidences and the current dynamic evidence is given to $m_t(D)$.

Table 2.1: Comparison between Dempster's rule and Jøsang's cumulative rule.

		m_{t-1}	m_p	$m_{t-1} \oplus^{\mathrm{D}} m_p$	$m_{t-1} \oplus^{\mathrm{J}} m_p$
$m(S)$	=	0.8	0	0.8	0.7059
$m(\{S, D\})$	=	0	0.4	0.08	0.1176
$m(\Theta)$	=	0.2	0.6	0.12	0.1765

And second, it is undesirable if the residual uncertainties, i.e., the mass of the unknown set $m_t(\Theta)$, converges towards zero. Dempster's rule has received critical responses due to non-intuitive results, in particular in cases with high conflict and little to no uncertainty, e.g., [78, 178]. Therefore, the normalization factor is adapted in (2.42) such that the resulting uncertainty after the combination is never smaller than a certain minimum uncertainty $\vartheta_{\min}$. The masses of all proper subsets of Θ are equally lowered, in order to yield the minimum uncertainty $\vartheta_{\min}$. Apart from this modification, the normalization factor corresponds to the normalization as in Dempster's rule. It is also noted that the highest conflict in the presented frame of discernment is expected to come from free and dynamic, which is resolved explicitly and is excluded in (2.43).

Other combination rules for belief fusion have also been proposed [115, 175] and in particular Jøsang's cumulative rule [77, 78]

$$\begin{aligned} m_1(A) \oplus^{\mathrm{J}} m_2(A) &= \frac{m_1(A)\, m_2(\Theta) + m_1(\Theta)\, m_2(A)}{m_1(\Theta) + m_2(\Theta) - m_1(\Theta)\, m_2(\Theta)} \\ m_1(\Theta) \oplus^{\mathrm{J}} m_2(\Theta) &= \frac{m_1(\Theta)\, m_2(\Theta)}{m_1(\Theta) + m_2(\Theta) - m_1(\Theta)\, m_2(\Theta)} \end{aligned} \tag{2.44}$$

which was also experimented with. However, Jøsang's cumulative rule lacks the logical component of Dempster's rule, since it only combines the masses of the same sets, disregarding any supersets. In particular, it does not incorporate the fact that S and D are subsets of $\{S, D\}$, which is shown in the numerical example in Table 2.1. Although the current estimated belief from the particle map, m_p, which has a non-zero belief mass for $\{S, D\}$, does not contradict the belief of m_{t-1} having a non-zero belief mass for S, the resulting mass for static is reduced. Hence, even though a high evidence for the hypothesis static exists and the current estimated belief supports the hypothesis for the superset, information in the more specific subsets are lost. Dempster's rule, on the other hand, produces the desired result and is therefore the rule of choice.

Although the Dempster–Shafer map contains all information about the static environment, Bayesian maps can be seen as the standard when it comes to grid maps. Therefore, it is shown next, how such maps are derived from an FSD Dempster–Shafer model.

2.4.2 Deriving Static Bayesian Maps

Many applications, which use grids of the static environment, work with occupancy probabilities, i.e., a single real number in the interval $[0, 1]$ that captures information about occupied space, free space, and unknown areas. For example, collision detection and cost

evaluation for path and trajectory planning, which is discussed in Chapter 4. Not only is it often easier to handle for subsequent algorithms, but it is also more efficient in terms of memory requirements, if the grids need to be transferred between different systems.

If a map of the static environment is required, it is tempting to simply use the belief mass for static. This, however, yields wrong results, since the masses of all other sets are ignored, such as free space, areas that have not been observed yet, or occupied evidences $\{S, D\}$. According to the transferable belief model [145], a belief at the *credal* level can be transformed into a *pignistic* level, if decisions need to be made, with the *pignistic transformation*. It calculates probability functions from the more general belief functions. For every atom A of the frame of discernment Θ

$$\mathrm{betP}(A) = \sum_{B \subseteq \Theta} m(B) \, \frac{|A \cap B|}{|B|} \tag{2.45}$$

is the probability of hypothesis A based on the belief m. It distributes the masses of the non-atom sets of Θ equally to the probabilities of the atoms.

In the case of the often used free-occupied frame of discernment $\Theta = \{F, O\}$ [113, 125, 176], the occupancy probability

$$\begin{aligned} p(o) &= m(O) + \frac{m(\Theta)}{2} \text{ , and} \\ p(\neg o) = p(f) &= m(F) + \frac{m(\Theta)}{2} \end{aligned} \tag{2.46}$$

can then directly be calculated using (2.45).

However, for an FSD frame, it needs to be considered that occupied is in fact the superset of static and dynamic and directly applying (2.45) means giving 2/3 of the mass of unknown to S and to D. Additionally, occupancy grid maps hold occupancy probabilities for binary random variables and refer to the static environment only. Hence, the mass of D needs to be treated separately and the three-element frame of discernment needs to be mapped to two probabilities, $p(o)$ and $p(\neg o) = p(f)$, as done in (2.46).

Similar to the previous section, evidences for dynamic are also interpreted as evidences for free space. For the aforementioned unequal mass distribution, it is however not possible to add all masses of $\mathrm{betP}(D)$ to $\mathrm{betP}(F)$, since $m(\Theta) = 1$ does then not result in an occupancy probability of 0.5. Hence, it is chosen to equally distribute the superset masses to the free and the occupied probabilities and to assign the mass of dynamic to free:

$$\begin{aligned} p(o) &= m(S) + \frac{m(\{S, D\})}{2} + \frac{m(\Theta)}{2}, \text{ and} \\ p(\neg o) = p(f) &= m(F) + m(D) + \frac{m(\{S, D\})}{2} + \frac{m(\Theta)}{2}. \end{aligned} \tag{2.47}$$

It is noted, that due to the equal mass distribution, the transformation from (2.47) completely ignores evidence for $\{S, D\}$ in the static map and only uses the masses of the atoms. For the applications in this thesis, having occupancy probabilities from dynamic

Figure 2.4: Color coding of the orientation of the estimated velocities. Orientations are relative to the current orientation of the robot (white mark to the right).

objects in the static map is equally undesirable as to miss a static object. If, however, not missing a static object is of higher priority, then $m(\{S, D\})$ can also be transferred in a biased way.

Results from real street scenarios from a vehicle equipped with a laser scanner and a radar sensor are given in the following.

2.5 Results

Finally, this section shows qualitative results in various different scenarios, such as a busy urban intersection with a high amount of occlusion, the successful detection of pedestrians and bicyclists, and results from laser-only, radar-only, and fused data. More results, especially of quantitative nature, are given in Chapter 6. Unless otherwise noted, the results come from laser scanner data. It was concentrated on the laser scanner, as no information about the velocity is available. The approach was implemented on a GPU using Nvidia CUDA and due to good parallel characteristics runs in real-time.

Color Coding

Two color codings are used. The PMap is visualized in the hue, saturation, and value (HSV) color space, where

$$\mathrm{H} = \sphericalangle(\mu_v, \theta), \quad \mathrm{S} = m_p(D), \quad \mathrm{V} = 1 - m_p(S). \tag{2.48}$$

Hence, the hue represents the orientation of the mean velocity vector of each cell velocity distribution relative to the current ego orientation, as shown in Figure 2.4, the saturation represents the dynamic evidence, and the value represents the inverse of the static evidence.

The DSTMap, on the other hand, is visualized in the red, green, blue, and alpha (RGBA) color space, where

$$\mathrm{R} = m_t(S), \quad \mathrm{G} = m_t(F), \quad \mathrm{B} = m_t(D), \quad \mathrm{A} = m_t(\Theta). \tag{2.49}$$

Busy Intersection

The first scenario, given in Figure 2.5, shows a busy urban intersection with both, crossing traffic (upper two rows) and turning traffic (lower two rows). Crossing traffic is the most

difficult scenario for many state-of-the-art algorithms that use cell conflicts between free and occupied for the detection of dynamic cells, as explained in the introductory part of this chapter. This is because, with crossing traffic, the same cell, over which an object moves, is occupied over multiple time steps by different parts of the object. Therefore, a conflict may only arise at the first occupied observation of that cell. This can also be observed in Figure 2.5d and 2.5i, which show standard occupancy grids, as described in [157]. The occupied cells of the dynamic objects are clearly visible, although they do not belong in the map of the static environment. Figure 2.5c and 2.5h show the Bayes map computed from the DST map, described in Section 2.4.2, for comparison. No artifacts from the dynamic objects are present.

Note that this scenario is also difficult for most tracking algorithms, as the crossing vehicles move at speeds of around 50 km/h requiring quick reactions. Additionally, the scene presents a high amount of occlusion.

Laser, Radar, and Fusion

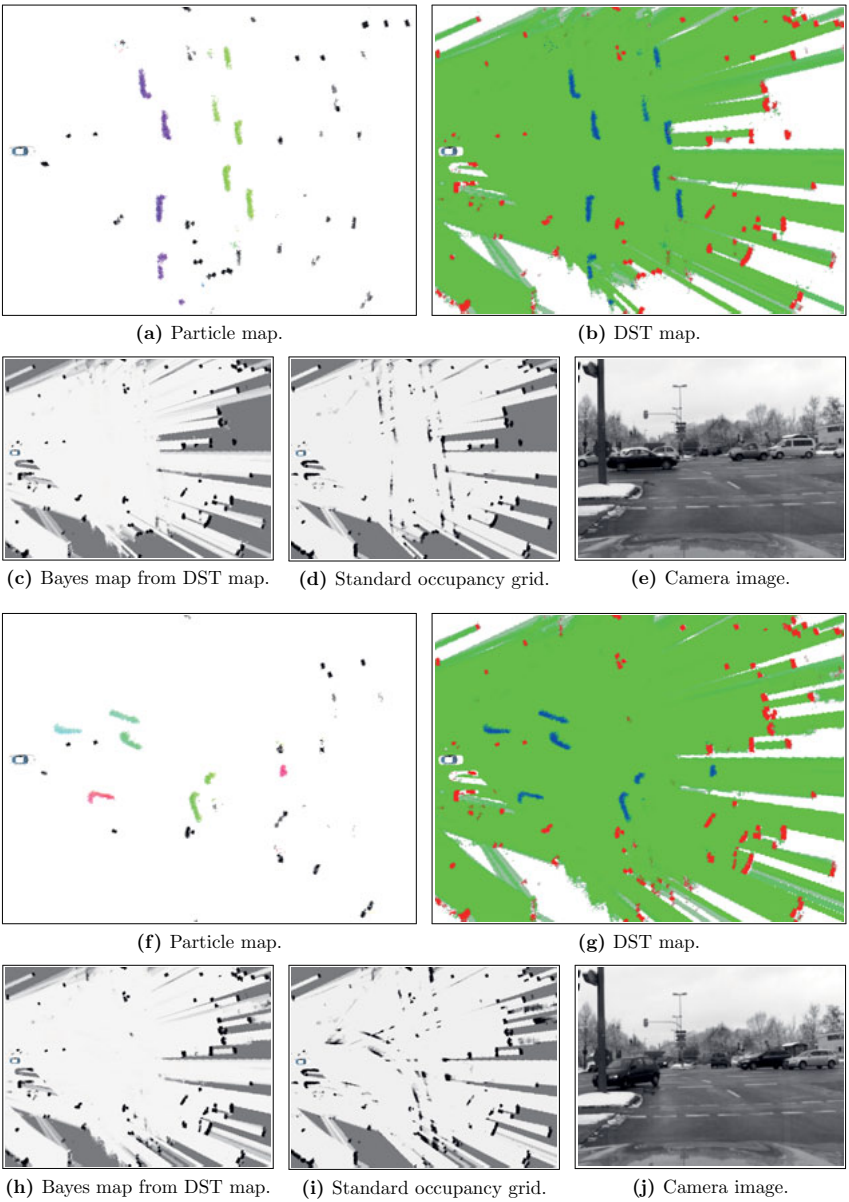
Next, in Figure 2.6, results from an imaging radar sensor are compared to the results from a laser scanner. Also given is the result of a simple cell-based fusion using Dempster's rule. The results are from the same scenario as Figure 2.2 and mostly already discussed in Section 2.2.3. Note that the fusion of laser scanner and radar data can also be done by solely using the scan grid of the laser scanner, and only use the radial velocity measurements of the radar sensor for the initial cell velocity distribution as well as for the intra-cell weighting. This way, the accuracy of the laser scanner can be retained.

Turning Ego Vehicle

Figure 2.7 shows results in a scenario, where the ego vehicle is turning to the right at an intersection following an object. Since the color coding of the orientations are relative to the ego orientation, the color of the tracked vehicle goes from purple to red. Also visible are approaching vehicles that stop in front of a red light.

Pedestrians and Bicyclists

Finally, results from tracking bicyclists, Figure 2.8, and pedestrians, Figure 2.9, are shown. Figure 2.8 also shows the static and the dynamic evidences from the particle map, $m_p(S)$ and $m_p(D)$, in the red and the blue channel in the bottom right image. Also interesting in Figure 2.9 is a vehicle that moves in reverse out of a parking lot.



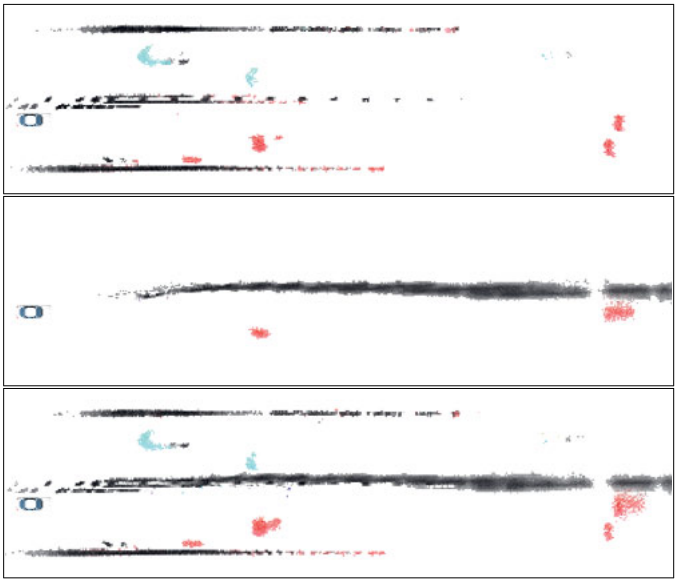
(a) Particle map. **(b)** DST map.

(c) Bayes map from DST map. **(d)** Standard occupancy grid. **(e)** Camera image.

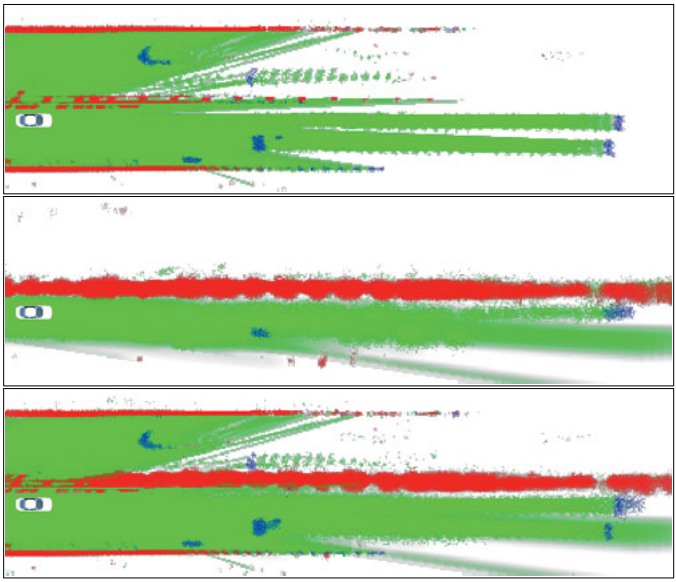
(f) Particle map. **(g)** DST map.

(h) Bayes map from DST map. **(i)** Standard occupancy grid. **(j)** Camera image.

Figure 2.5: Results at a busy intersection with a high level of occlusion. Upper two rows show crossing vehicles, lower two rows show turning vehicles.



(a) Particle map.



(b) DST map.

Figure 2.6: Result from the upper scenario from Figure 2.2. From top to bottom: laser scanner, radar sensor, and laser-radar fusion with Dempster's rule.

Figure 2.7: Ego vehicle turning at an intersection to the right.

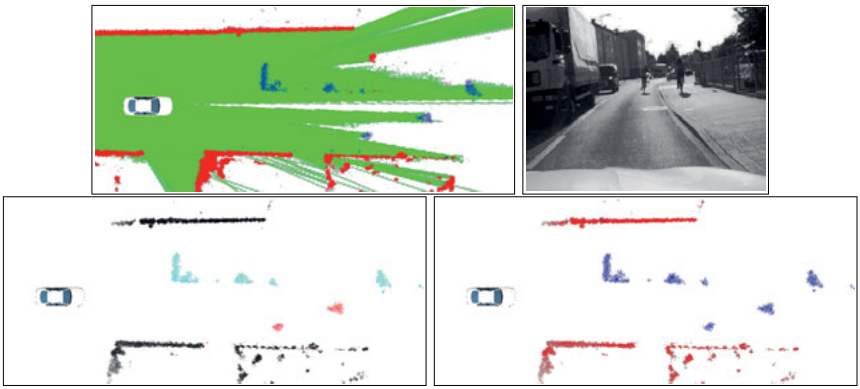


Figure 2.8: Two bicyclists moving in the ego direction and two approaching vehicles.

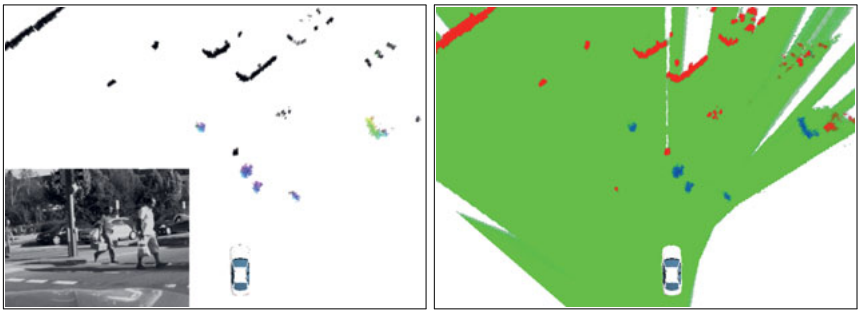


Figure 2.9: Pedestrians at a crosswalk.

2.6 Summary

Mapping and tracking yield the important basis of the environment model. They give the robot a robust representation of the obstacles around it, which is one of the fundamental requirements for tasks such as motion planning. Especially online-generated grid maps, often used for checking collision against the static environment, need to be free of any occupancy values coming from dynamic objects. Since autonomous vehicles move rather fast, compared to other mobile ground robots, the amount of filtering of individual scans is limited, and therefore artifacts from dynamic objects are particularly likely to occur. They need to be treated with explicitly in order for the grid map to be useful for navigation. On the other hand, rather than based on grids, tracking is usually performed under the assumption of certain shapes and a list of objects is estimated.

A novel method termed Grid-based Tracking and Mapping (GTAM) has been presented that simultaneously estimates the dynamic and the static environment from raw, unclassified sensor measurements in a uniform grid-based structure. It overcomes problems due to different forms of representations of the static and the dynamic world. Both are estimated simultaneously without assuming known shapes and without the need for data association between objects and sensor data, as objects do not exist in this representation.

The approach estimates a velocity distribution for each cell using particle filters. Although the representation is a discretized grid, the velocity hypotheses, represented by the particles, are continuous samples. The estimated velocity distributions are used to derive static and dynamic evidences, which are in turn filtered over time, in order to create a robust and complete model of the environment. It represents free space, static occupancy, dynamic occupancy, static-dynamic occupancy, i.e., where the distinction is not known or has not yet been estimated, and the residual uncertainty.

The problem has been formally formulated and details about the particle filter and the map filtering have been given. It has also been shown, how a Bayesian map of the static environment can be derived from the Dempster–Shafer model, which is more widely used. It represents occupied space, free space, and unknown space in form of a single probabilistic variable. This Bayesian map is used in the following chapters to extract information about the environment, such as the principal moving directions discussed next.

3 Detection of Principal Moving Directions

The principal moving directions through the local environment yield the main possible maneuvers of the robot. They represent the road topology as well as a geometric representative of each topological choice. With this information forks in the road and road junctions are detectable, as well as the number of present road courses. Furthermore, they separate the roadway into a left region, containing the left boundary, and a right region, containing the right boundary. The road course estimation, which will be presented in Chapter 5, is also based upon the principal moving directions. In this chapter, the question of how to find them is examined. The presented approach is based on motion planning and therefore considers the geometric and kinematic constraints of the underlying system—an autonomous vehicle. This enables estimations of good quality even with sparse road boundaries like traffic cones, where the road topology is independent of the obstacle topology. Moreover, no assumptions about the number of principal moving directions or the shape of the road courses are made. This chapter is based on work that appears in [6] and [9] published in the context of this thesis.

3.1 Introduction

Motion planning is one of the fundamental problems in robotics. It usually consists of finding a collision-free path or trajectory from one specific start state to one or several specific goal states [96]. In some situations, however, the pose of the goal state, i.e., the location and the orientation, are not known.

In a typical autonomous vehicle system, motion planning consists of several layers, from high-level maneuver planners, also referred to as driving strategy [14, 15, 18], over path planners, such as the one presented in this chapter, to low level trajectory planners [170, 171]. As given in the introductory part, often stored maps form the basis in autonomous vehicle navigation. If it cannot be relied on such an offline-generated map, however, a planner does not receive goal locations or regions to plan towards anymore.

Without goal states the planning problem becomes different, as it is not clear, what kind of paths it is searched for. In this work, the length is used as the goal criterion, as will be described in more detail in Section 3.2. If a path of a certain goal length is found, it is still not guaranteed that its goal pose is desired, as this path, even if it is optimal under some cost function, may not follow the road. It may happen that it leaves the true road course in between two traffic cones—a problem that will be investigated in more detail in Chapter 5. Additionally, it is possible that more than one road course is available and that a higher level decision module wants to be able to choose from them. Therefore, many paths need

to be sampled, where not all of them are going to be relevant. In fact, most of them will be very similar. Hence, they are combined, i.e., clustered, and only one representative out of each cluster is used. Next, related work about path planning and clustering is given.

3.1.1 State of the Art

As discussed in the previous chapter, grids are a common way to represent the environment and are the world model of choice in this work. Therefore, it is focused on planners that work on grid maps. Additionally, since the field of motion planning is vast, it is focused on A* and Rapidly Exploring Random Trees, since these two families are combined in the planner presented in Section 3.2.

A* and RRT Planners

Many computationally hard problems in computer science and related fields are handled with graphs to decrease and manage the complexity. In the simplest case, a grid can directly be represented as a graph by connecting the neighboring cells. If the problem is formulated as a graph, a variety of search algorithms, like Dijkstra's and A*, are available, such as to search for the shortest path. Much work has been done on graph search algorithms in the context of motion planning. The original A* has been extended, e.g., to allow incremental search. Algorithms such as D* [147] are able to quickly re-plan in changing environments and any-time planners like Anytime Repairing A* [104] rapidly find a potentially sub-optimal path and approach the optimal one with more computing time.

Apart from combinations of both, e.g., Anytime Dynamic A* [103], there are also works that focus on the graph itself rather than a specific search algorithm. They target the problem of unnatural and suboptimal paths produced by A* and many of its variants, which is primarily due to the representation of the graph. Non-holonomic systems, such as vehicles, cannot follow grid-like patterns, simply because they are not able to instantaneously change their orientation without moving. Even for robots that are able to, such zick-zack-patterned paths are highly suboptimal as they typically require the robot to stop, to change orientation, and to accelerate again. Theta* [116] modifies how the search graph is generated. Instead of restricting the neighbors of each node in the graph to the neighbors of its grid cell, an edge between two nodes can traverse multiple cells. Hence the move direction from one node to another is not limited to the either 4 or 8 neighbors anymore. Field D* [57] also provides a way to represent an any-angle move between two cells. Cell costs are associated to the cell corners rather than the cell centers and an any-angle transition is calculated by linearly interpolating the cost of the corresponding cell corners.

Planning linearly in an any-angle fashion is still not enough to satisfy the non-holonomic constraints. Different to graphs which are directly defined on the grid, graphs can also be defined completely independent of the underlying object representation. Instead of linear edges between neighboring vertices, the edges can be designed to be feasible by the system. The system state is encoded in the graph node and a system model is used for the generation. Often, every graph node has a fixed number of outgoing feasible edges that may be precalculated. These are referred to as the motion primitives. Motion primitives are short path segments that are sticked together to create a path. In state lattices [128, 129]

the motion primitives are carefully designed to create graphs exhibiting a high amount of cycles in order to reduce the number of leaf nodes and thus the complexity of the graph.

Hybrid-state A* [49] is a variant similar to Field D*, but without the limitation of piecewise linear paths. The state space is discretized into cells, but the graph nodes may land at arbitrary continuous positions within the cells. The continuous positions are stored and transitions to the child nodes are done using this continuous state and by applying the system model or by using motion primitives. Edges are pruned with the discrete cells to reduce the graph complexity. Hybrid-state A* is, however, not guaranteed to find the minimum cost path due to the pruning and a violation of the Markov property, but the output typically lies in the neighborhood of the global optimum, as the authors point out. Compared to state lattices, this approach allows more flexibility of the motions between nodes, but is not as effective in pruning.

Apart from A*, Rapidly Exploring Random Trees (RRT) [95, 96] are another popular family of planning methods. In the RRT algorithm, a tree is built that grows towards random continuous samples from the search space. A path to the goal is found by sampling the goal state with some, usually low, probability and thus by trying to connect the search tree with the goal. Although the original RRT does not use cost values and therefore the resulting paths are typically far from being optimal, extensions targeting this issue have been developed [56, 74, 82, 160].

Not many combinations of A* and RRT exists. In roadmap based methods, RRTs are used to construct the roadmap and A* to compute shortest paths in this graph [21]. In [86], the workspace is decomposed and a heuristic similar to A* is used to construct a roadmap. In [130], a workspace decomposition is used to run a discrete graph-search algorithm like A* to obtain leads for a continuous search [130].

Without goal poses, the planning problem becomes different, as described above. Path planning without known goal poses is related to reachability analysis, which aims at extracting all reachable states from a certain start state within a fixed time frame [13]. Also, there exists a variety of trajectory planners that do not have direct goal poses, such as trajectory planners that create and evaluate trajectory bundles often around a guidance path or street model, e.g., [164, 171] or collision avoidance trajectories planners, e.g., [65]. Other than in offroad environments, if no reference or guidance path is available, such approaches alone are not sufficient for real street scenarios, and the computational complexity increases in a way that exact reachability analysis is not feasible anymore. It is however noted that the local path planner, which is presented in the following can also be viewed as an approximation of the reachable set.

The set of planned and evaluated paths is reduced to the principal moving directions in a second step through path clustering. Related work to path clustering is therefore given in the following.

Trajectory Clustering and Homotopy

As written above, many of the paths generated in sensor-based local planning will be similar and can be combined. The main question is, however, how similarity is defined. This question is tightly coupled to the problem one wants to solve. Clustering of robot trajectories is used in a variety of different tasks, e.g., in learning trajectories for manipulation

activities [124], in motion prediction [148], in street intersection surveillance [16], or in grouping trajectories to detect possible maneuvers [6, 22, 88]. Many well-studied algorithms exist in the clustering literature, such as k-means [51], hierarchical clustering [51], DBSCAN [54], or neural networks [179]. Unlike data points residing in Euclidean space, there is no standard definition of how similar two paths are. Often, geometrically-motivated distance measures between trajectories are used [6, 16, 27, 124], such as the summed Euclidean distance between corresponding trajectory points evaluated at the same time or length parameter [6], the Hausdorff distance [16], or the longest common subsequence [27]. Other approaches cluster the motions that yielded the trajectory to discover common sub-trajectories [98, 148]. In the end, though, they also rely on some sort of continuous similarity measure.

In the field of motion planning, path homotopy is also used to group trajectories [63, 70, 139]. Contrary to traditional clustering algorithms, in homotopy, two paths, sharing the same start and end point, are in the same homotopy class, if there exists a continuous, collision-free deformation between them. In [22], homotopy classes are calculated based on the Cauchy integral theorem. The authors formulate a graph, which allows a direct computation of paths in distinct homotopy classes and also show how to search in this graph. However, since every graph edge requires an integration to be performed and the graph needs to be augmented, the computational performance will most likely not meet the requirements for autonomous vehicles. Additionally, since it is based on homotopy, all paths share the same goal state.

In [88], an equivalence definition between local paths is presented, which is similar to homotopy but does not require the endpoints to be the same for all paths. The authors define two paths to be equivalent, if their *swaths*, i.e., the workspace area swept by the robot, overlap. They use this information for speeding up collision checking. Although the local paths also do not share the same end points, the overlapping criterion requires a dense and uniform path sampling. If paths are calculated online in real-time and are not precalculated as in [88], this can usually not be guaranteed. In addition, in its essence it is similar to a distance-based clustering described above.

Principal Moving Directions

There are also methods for the detection of the principal moving directions that are not directly related to motion planning, but to road detection. Road course estimation is the topic of Chapter 5, where more related work will be presented.

In [106, 108] a colored elevation map is created from lidar and camera data and used to detect and track road networks. A particle filter framework is used to recursively estimate a potential intersection point as well as the parameters of a clothoid model for each branch relative to the ego vehicle center. However, the road topology, i.e., the number of principal moving directions, is inferred from map data and needs to be known. In [20] the road network is detected without map data. A road/non-road probability map is calculated based on the colored elevation map. Using a given road width, road template masks are correlated with the elevation map in order to estimate an orientation-dependent road center probability map. Hypotheses of road intersection centers are created along clothoids starting from the ego vehicle position and evaluated by casting rays in all directions. Valid

rays are clustered and used as observations for an extended Kalman filter. The use of a colored elevation map from camera and lidar data provides the basis to present a wide spectrum of road scenarios, such as rural roads. The road model, however, consists of an intersection point and a line segment for each branch. The shape of the principle directions as well as the horizon of the estimations is thus limited. Arbitrary shaped curves, such as S-shapes popular in road construction sites, cannot be detected. Using motion planning as basis for the detection of the principle moving directions enables an estimation of arbitrary shapes over a long horizon.

In [48] line segments are detected in an obstacle map trough the Canny edge detection and the Hough transform. They are used as input to a Markov random field model. The output of the Markov random field gives a uniform 2-D field of main directions of the road. Here, the aim is, however, the inference of a path or a connected set of points for each road branch, as well as extracting the road topology, rather than a 2-D field of orientations.

3.1.2 Approach and Contribution

This chapter presents two novel contributions. First, a local path planner is presented that is designed for planning without known goal poses or without known street model, i.e., guidance. Due to the combinatorial complexity of the problem, it rigorously exploits constraints and discretizations and restricts the problem to finding paths of a predefined desired length. The problem is defined on a graph and transitions, i.e., edges in this graph, are feasible short motions. The novelty of the planner lies in the seamless combination of A* and RRTs in the graph expansion during the search. Hybrid-state A* [49] is used to quickly find the optimal path through the environment. After having found the optimal path, which can neither be guaranteed to correspond to a legal principal moving direction, as it possibly leads to leaving the street, nor to be the only principal moving direction, such as at road junctions, the planner switches to an RRT search. It does so, however, using the same, partially-explored graph structure. Therefore, no calculations are done twice. The resulting planner

- is optimal under the given discretization, resolution-complete, and focused towards the goal criterion due to A* and the graph formulation,
- is uniformly-exploring once the first goal path is found, i.e., it tends to explore the largest yet unexplored regions, and
- incorporates any-time characteristics allowing to trade-off runtime and dense exploration.

The second contribution regards trajectory clustering. Similar to homotopy and differently to traditional clustering algorithms, trajectory similarity is binary in the proposed approach and does not rely on a distance metric together with distance thresholds. It uses the environment structure, i.e., the objects in the local environment of the robot, to find equivalences between trajectories. Different to homotopy and similar to [88], this method is aimed for local planning and therefore the trajectory endpoints do not need to be all the same. A closed surface between two trajectories is created by sampling inter-trajectories

and by using their endpoints to create a polygon. Equivalence based on the environment structure is then efficiently checked with point-in-polygon tests. The presented method is characterized by:

- providing a binary trajectory equivalence predicate for local trajectories with different trajectory end points, which does not involve distance measures and distance thresholds,
- creating closed surfaces by sampling inter-trajectories, and by
- a computational complexity that is linear in the number of trajectories for non-overlapping clusters and, under certain assumptions, also for overlapping clusters.

Next, the A*-RRT local path planner is presented in Section 3.2 and Section 3.3 describes the clustering. Finally, in Section 3.4 first results are given, which will be complemented in Chapter 6.

3.2 Local Path Planning with Unknown Goal Poses

In this section, the local path planner is presented. First, the problem is formulated in Section 3.2.1. Then, in Section 3.2.2, the structure of the graph is given and it is shown how an admissible heuristic can be designed in the absence of goal poses in Section 3.2.3. Finally, Section 3.2.4 presents the algorithm of the planner.

3.2.1 Problem Formulation

The motion planning problem that is the subject of this section is formulated as follows. Let $q_S \in \mathcal{C}$ be the initial configuration of the vehicle in the configuration space $\mathcal{C}$, l_G the desired goal length in the workspace, and $\tau : [0,\ l] \to \mathcal{C}_{\text{free}}$ a collision-free trajectory with workspace arc length l. The goal is to find a set

$$\mathcal{T} = \{\tau \mid \tau(0) = q_S \wedge l = l_G\}, \tag{3.1}$$

of trajectories

$$\tau(l_G) = q_S + \int_0^{l_G} f_m\left(\tau(l), \omega(l)\right) \mathrm{d}l \tag{3.2}$$

created based on some motion model f_m with action trajectories ω.

There are two desired properties about $\mathcal{T}$ from (3.1). On the one hand, $\mathcal{T}$ is desired to contain the minimum cost trajectory, since without application-specific knowledge, this is assumed to be the best guess about a possible path through the environment. On the other hand, $\mathcal{T}$ is desired to exhibit diversity, so that the information gain between different elements in $\mathcal{T}$ is high and subsequent modules can then choose from a greater variety.

In the formulation of (3.1) and (3.2), the length is used rather than the time, as the focus is to estimate the principal moving directions through the environment and with these the road courses and the road boundaries, which is the topic of Chapter 5. They are independent of a particular time parameterization. It is noted, however, that time

does indeed play a role in the presented system. Since the problem is of global nature in the sense of the search space, rather than local optimization given a reference path, higher order constraints and continuity levels are not of major relevance for the problem. They significantly increase the computational complexity and thus violate the real-time constraints. However, lateral acceleration constraints are exploited to reduce the search space. They directly influence the search graph as described next. A certain safely drivable velocity v_{road} is assumed, which induces curvature constraints on the motion primitives due to an assumed maximum lateral acceleration. Moreover, the length of the primitives is dependent on a certain fixed time interval and the velocity v_{road}. Since the velocity constraints are transformed into curvature and arc length constraints of the primitives, it is referred to as a path planning problem rather than trajectory planning and to $\mathcal{T}$ as a set of paths. However, as a different planner, such as a trajectory planner, can sample $\mathcal{T}$, in subsequent steps, such as the clustering, it will not be clearly differentiated between paths and trajectories.

3.2.2 Velocity-dependent Reachability Graph

Rather than a fixed, carefully-designed lattice consisting of offline-generated motion primitives, as in [128, 129], the search graph is generated online. In many path planning applications, the use of an offline-generated lattice has advantages, since the amount of online computation is significantly reduced. In addition to the primitives, the traversed cells needed for collision checking and cost evaluation can also be pre-calculated saving online computation time. However, if velocity and acceleration constraints are to be considered, the amount of pre-calculation and the size of the look-up tables grow tremendously. Therefore, it is calculated online in this work.

The system model was chosen as the bicycle model, as it is simple and efficient, but still captures the non-holonomic constraints. Let α denote the steering angle of the wheels, d the driven distance, and l_a the axis length, then the new configuration $q' = \begin{pmatrix} x' & y' & \theta' \end{pmatrix}^{\mathrm{T}}$ is calculated from the current configuration $q = \begin{pmatrix} x & y & \theta \end{pmatrix}^{\mathrm{T}}$ as

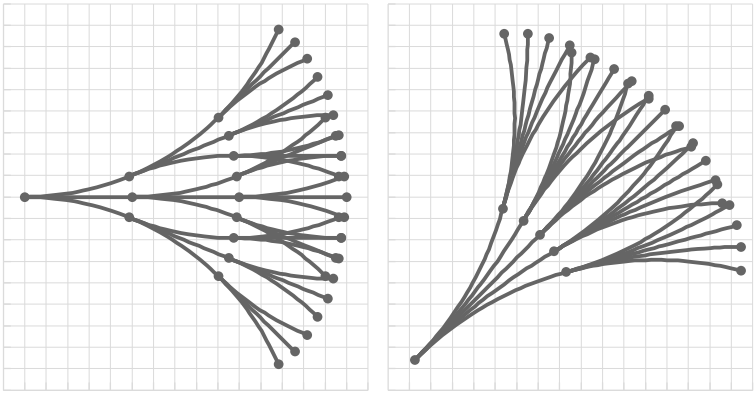
$$f_m(q, \alpha) = \begin{cases} \begin{pmatrix} (x + d\cos\theta) & (y + d\sin\theta) & \theta \end{pmatrix}^{\mathrm{T}} & \text{if } \alpha = 0 \\ \begin{pmatrix} (x - r\sin\theta + r\sin(\theta+\beta)) & (y + r\cos\theta + r\cos(\theta+\beta)) & (\theta+\beta) \end{pmatrix}^{\mathrm{T}} & \text{else} \end{cases} \tag{3.3}$$

where

$$r = l_a / \tan\alpha, \quad \beta = d/l_a \tan\alpha. \tag{3.4}$$

Note that the map grid coordinate system (MGCS), as described in Appendix A.3, is used here, i.e., positive angles are clockwise.

The motion primitives are generated using a maximum steering angle $\alpha_{\max}$. It is dependent on the estimated drivable velocity v_{road} and calculated by assuming a certain maximum lateral acceleration. Let n denote the number of different turning angles without turning angle 0, i.e., going straight. The total number of primitives $n_{\text{prim}} = 2n + 1$ is



(a) Reachability graph with 3 primitives. **(b)** Reachability graph with 5 primitives.

Figure 3.1: Two reachability graphs with different continuous start positions, number of children, maximum turning angle $\alpha_{\max}$, and driven distance d.

odd. The steering angles α, yielding the set of actions U and used for each node expansion of the graph, are then calculated as

$$U = \left\{ -\alpha_{\max},\ -\frac{n-1}{n}\alpha_{\max},\ -\frac{n-2}{n}\alpha_{\max},\ \dots\ ,\ 0,\ \frac{1}{n}\alpha_{\max},\ \frac{2}{n}\alpha_{\max},\ \dots\ ,\alpha_{\max} \right\} \tag{3.5}$$

and the new nodes are generated using (3.3).

In order to keep the computational complexity low and since the paths are not primarily generated to follow them, but rather to detect the principal moving directions, the number of primitives is chosen to be low. It was experimented with $n_{\text{prim}} = 3$ and $n_{\text{prim}} = 5$. Most of the time, however, only 3 primitives per expansion were used.

The second design parameter in (3.3) is the driven distance d, i.e., the length of the primitives. It is also calculated based on v_{road} and a fixed small time duration. Whereas the number of primitives controls the number of children of each node, the length of the primitives controls the number of hierarchy levels, given a fixed goal path length l_G. Figure 3.1 shows two different reachability graphs over the grid.

The motion primitives do not need to land in the neighboring grid cells but usually traverse multiple cells. This way, the resolution of the grid and the resolution of the graph are independent, facilitating the trade-off between collision-checking accuracy and search time due to a graph with fewer hierarchy levels and thus a reduced complexity. However, since a motion primitive traverses multiple cells, all of these need to be considered during collision-checking and cost calculation, as will be described in Chapter 4. Moreover, an additional grid may be used to control the amount of pruning.

3.2.3 Path Cost and Heuristic with Unknown Goal Poses

Typically, one is not only interested in *some* collision-free path, but in one that is *good* or even optimal given some cost criteria. The most famous cost criterion is the path length, as it is used for the well-known shortest path problem. Often, however, and especially for autonomous vehicles, other costs are used either in addition or as replacement. In general, costs can be separated into two categories: path-intrinsic costs and workspace-intrinsic costs. The path-intrinsic costs, sometimes referred to as action costs, are costs due to the shape of the path, or due to the actions that cause the shape of the path, such as making a turn. Examples are cost for curvatures, cost for changes of the curvature, or cost for switching driving direction. Path costs due to the workspace, on the other hand, are the distance to objects, or occupancy probabilities, such that the robot prefers traversing areas that have been observed to be free. The latter costs and their calculation will be examined in detail in Chapter 4.

In A*, the search is based on a heuristic, which directly relates to the costs that are used. The heuristic represents the estimated cost from a node to the goal. For the heuristic to be admissible, it must not overestimate the true cost to the goal. If the path length is used as the cost, then often the straight-line distance to the goal is used as heuristic. Even without goal poses, since the goal path length l_G is known, a heuristic can be derived, as the goal are then paths of a certain fixed length. In order to use arbitrary costs, such as path-intrinsic or workspace-intrinsic costs, some requirements need to be met. Let c_i be the edge cost between two adjacent nodes and l_i the corresponding length of the path segment. Then, by restricting the minimum edge cost between two adjacent nodes to the segment path length, i.e., $c_i \geq l_i$, one admissible heuristic is defined as

$$h(q) = \max(l_G - l_q, 0), \tag{3.6}$$

where l_q represents the path length from q_S to configuration q.

If the total path cost is the sum of the path extrinsic cost and the path intrinsic cost, including the path length, and all costs are positive, then $c_i \geq l_i$ always holds. Furthermore, if all motion primitives have the same length, the hierarchy level of the graph can be used instead of the length by normalizing the primitive length to 1. The heuristic from (3.6) then simplifies to

$$h(q) = \text{depth}(l_G) - \text{depth}(q). \tag{3.7}$$

Next, the A*-RRT planner is presented.

3.2.4 A*-RRT Motion Primitive Path Planner

In order to efficiently create the sample set $\mathcal{T}$ of collision-free paths from (3.1), a novel motion primitive path planner is used. Based on an estimated drivable velocity v_{road}, the turning angles and the length of the primitives are determined, as described in Section 3.2.2. The graph is created, i.e., expanded, and each graph edge is checked for collision and cost online. It is in general not computationally possible to expand and test the whole graph until the desired goal hierarchy level is reached. Therefore, it is crucial in what order the nodes of the graph are expanded.

The proposed planner combines the concepts of two well-known classes of planning algorithms, A* [67] and the Rapidly Exploring Random Tree (RRT) [95]. Due to the heuristic, A* is focused towards the goal. A* is also optimal under the graph discretization, i.e., the first path satisfying the goal criterion that it finds, is the minimum cost path. These properties fit the observations discussed in Section 3.2.1 and therefore this planner starts with an A* search. Hence the set of paths $\mathcal{T}$ from (3.1) always contains the minimum cost path. As described in the introductory part of this section, finding just the optimal path is not sufficient for the problem at hand and thus the search is continued. Usually, however, the cost function is locally uniform and thus similar paths exhibit similar costs. Therefore, if A* is continued to run, the subsequent paths will be similar to the optimal path and it will most likely take a long time to discover new regions of the search space.

RRTs are complementary to the focused search of A*. Uniform-sampling RRTs have the property that nodes with large Voronoi regions are more likely to be expanded than nodes with small Voronoi regions [95]. Therefore, the RRT grows favorably into yet unexplored areas. After the optimal path is found with A*, the planner switches to an RRT node expansion strategy, in order to quickly grow into the yet unexplored regions. This favors diversity in the resulting set of paths $\mathcal{T}$. The planner switches, however, keeping and re-using the same search structures, i.e., the already partially expanded graph during the A* stage. Hence, every state is only visited once.

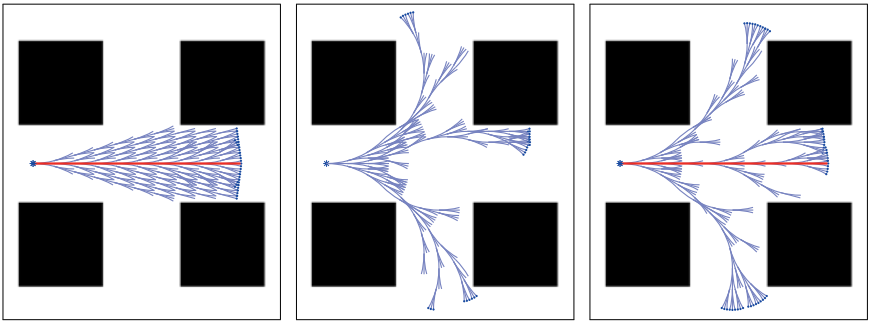
The pseudo-code of the path planner is given in Algorithm 3.1. It uses 3 lists: an open list $\mathcal{L}_{\text{open}}$ containing the next nodes to be expanded, a closed list $\mathcal{L}_{\text{closed}}$ containing the nodes that have already been expanded, and a goal list $\mathcal{L}_{\text{goal}}$ containing the discovered goal states. Once the first goal candidate is found, the algorithm switches the method that yields the next to node to be expanded from popA*() to popRRT().

popA*(): As in conventional A*, the node q with the lowest $f(q) = g(q) + h(q)$ score is chosen for expansion, where $g(q)$ gives the path cost from q_S up to q and $h(q)$ gives the heuristic from q to the goal.

popRRT(): A random sample in the search space is generated and its closest node in the open list $\mathcal{L}_{\text{open}}$ is expanded.

Pruning reduces the complexity of the graph. If a *move* lands in a cell, which has already been expanded, i.e., that is on the closed list $\mathcal{L}_{\text{closed}}$, it is discarded. If a transition ends up in a cell already visited but not yet expanded, i.e., which is on the list of nodes to be processed $\mathcal{L}_{\text{open}}$, the costs of the nodes are compared and the cheaper one is used, as can be seen in line 23-27 of Algorithm 3.1.

Figure 3.2 shows a simulation scenario to demonstrate the effect of the combination of A* and RRT. It compares running A* for 100 iterations, running RRT for 100 iterations, and running the proposed A*-RRT combination for 100 iterations. The start state is marked with a blue star to the left and the extracted goal states are marked by blue dots. The path cost in this example is a combination of the normalized length, as described in Section 3.2.3, and an action cost that penalizes making turns. It can be observed that with A* only paths similar to the minimum cost path are found. From the three possible maneuvers, i.e., turning left, turning right, and going straight, only one was discovered.



(a) A* only. **(b)** RRT only. **(c)** Combination of A* and RRT.

Figure 3.2: Comparison between different expansion strategies. A total of 100 iterations was applied. The minimum cost path, which is marked in red, is found after 10 iterations in (a) and (c).

With the RRT algorithm, the simplest and cheapest path, however, which only consists of going straight, is missing, although all three maneuver-possibilities were discovered. Combining A* with RRTs, both properties are achieved. Still, many of the sampled paths will be similar, and they are therefore combined as shown next.

Algorithm 3.1 A*-RRT Planner with Unknown Goal Poses

Input: Start state q_S, velocity v_{road}, goal path length l_G
Output: Set of goal paths $\mathcal{T}$

$\mathcal{L}_{\text{open}}, \mathcal{L}_{\text{closed}}, \mathcal{L}_{\text{goal}} \leftarrow \emptyset$ // initialize lists of states $\mathcal{L}$
$e.\text{state} \leftarrow q_S$ // create new list element e
$e.[f, g, \text{depth}, \text{parent}] \leftarrow 0$
$i \leftarrow 0$
$\mathcal{L}_{\text{open}}.\text{push}(e)$
while $\mathcal{L}_{\text{open}} \neq \emptyset$ **and** $i < n_{\text{iter}}^{\max}$ **do**
  if $\mathcal{L}_{\text{goal}} = \emptyset$ **then**
    $e_{\text{curr}} \leftarrow \mathcal{L}_{\text{open}}.\text{popA*}()$
  else
    $e_{\text{curr}} \leftarrow \mathcal{L}_{\text{open}}.\text{popRRT}()$
  end if
  $\mathcal{L}_{\text{closed}}.\text{push}(e_{\text{curr}})$
  for each action $u \in U$ **do**
    $e_{\text{next}}.\text{state} \leftarrow \text{move}(e_{\text{curr}}.\text{state}, u, v_{\text{road}})$
    if $\text{noCollision}(e_{\text{curr}}.\text{state}, e_{\text{next}}.\text{state}) \wedge e_{\text{next}}.\text{cell} \notin \mathcal{L}_{\text{closed}}$ **then**
      $e_{\text{next}}.g \leftarrow \text{calculateTotalCost}(e_{\text{curr}}, e_{\text{next}})$
      if $\mathcal{L}_{\text{goal}} = \emptyset$ **then**
        $h \leftarrow \text{calculateHeuristic}(e_{\text{next}}.\text{state})$
        $e_{\text{next}}.f \leftarrow e_{\text{next}}.g + h$
      end if
      $e_{\text{next}}.\text{depth} \leftarrow e_{\text{curr}}.\text{depth} + 1$
      $e_{\text{next}}.\text{parent} \leftarrow e_{\text{curr}}$
      if $e_{\text{next}}.\text{cell} \in \mathcal{L}_{\text{goal}} \cup \mathcal{L}_{\text{open}}$ **then**
        $e_{\text{other}} \leftarrow \text{getNodeInCell}(e_{\text{next}}.\text{cell})$
        if $e_{\text{next}}.g < e_{\text{other}}.g$ **then**
          $\text{exchangeNodes}(e_{\text{next}}, e_{\text{other}})$
        end if
      else if $e_{\text{next}}.\text{depth} \geq l_G$ **then**
        $\mathcal{L}_{\text{goal}}.\text{push}(e_{\text{next}})$
      else
        $\mathcal{L}_{\text{open}}.\text{push}(e_{\text{next}})$
      end if
    end if
  end for
  $i \leftarrow i + 1$
end while
$\mathcal{T} \leftarrow \text{extractPathsFromGoalStates}(\mathcal{L}_{\text{goal}})$

3.3 Environment-based Trajectory Clustering

The local paths, generated as shown in the previous section, are clustered to detect the principal moving directions in the local environment of the vehicle. Contrary to classical clustering algorithms, such as k-means, hierarchical clustering, or DBSCAN, which rely on continuous, distance-based similarity measures, path similarity is binary in the presented approach. It only depends on the objects in the environment, similar to homotopy classes of paths. In homotopy, two paths are path homotopic, if a continuous, collision-free deformation between them exists [114]. As described above, differently to homotopy, the end points of the paths do not always match with local paths. However, if the endpoint requirement is relaxed, all paths are equivalent with the definition of path homotopy. Therefore, a different equivalence definition is introduced in this chapter and used to formulate an efficient clustering algorithm.

Section 3.3.1 gives the equivalence definition, which is formulated as a predicate. Using the predicate defined on two paths, it is then shown in Section 3.3.2 how the paths are clustered. Finally, Section 3.3.3 examines clustering in the case of overlapping clusters.

3.3.1 Equivalence for Local Trajectories

It is started with a formal definition of path equivalence. Then, it is shown how the equivalence between two paths is efficiently checked using polygonal approximations by sampling inter-paths. Although the problem is similar to [88], the approach is fundamentally different. Whereas in [88], it is assumed that the paths are precomputed and sampled so densely, that the workspace volumes swept by the robot, as it traverses the paths, strongly overlap, this approach is designed for online generated paths without such requirements.

Equivalence Definition

Here, it is slightly departed from the use of paths parameterized over the arc length and the use of configurations. In order to present the clustering predicate in a more general notation and to underline that the paths do not need to be all of the same length, time is used instead of length and states instead of configurations.

Given a dynamic system $\dot{x} = f_m(x, u)$ and an action trajectory $\omega : [0, t] \to U$ from an action space U. The action space may be bounded, but must have a continuous interval per dimension. Then, the binary equivalence predicate, i.e., a Boolean-valued, binary function, denoted as pred, is defined as follows.

Definition 3.1 *Two trajectories τ_1 and τ_2, defined in the same time interval $[0, t]$, starting at the same state x_S, with action trajectories ω_1 and ω_2, are in the same cluster, i.e.,*

$$\mathrm{pred}(\tau_1, \tau_2) = \mathrm{true}, \tag{3.8}$$

if all states of all trajectories

$$\mathcal{T}_{\tau_1,\tau_2} = \left\{ \tau \mid \tau(t) = x_S + \int_0^t f_m\left(\tau(t'), \omega(t')\right) \mathrm{d}t',\ \forall \omega \in \Omega \right\} \tag{3.9}$$

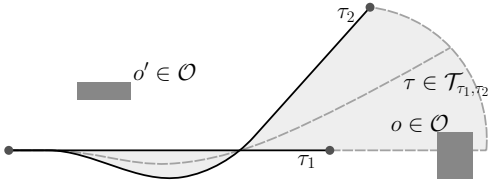


Figure 3.3: The clustering predicate evaluates to false due to the obstacle o.

with the set of action trajectories

$$\Omega = \{\omega \mid \omega(t) = f_w(t)\omega_1(t) + (1 - f_w(t))\omega_2(t),\ \forall f_w \in \mathcal{F}_w\} \tag{3.10}$$

are collision-free. Otherwise,

$$\text{pred}(\tau_1, \tau_2) = \text{false}\,. \tag{3.11}$$

The set of action trajectories Ω must be defined such that $\forall \tau \in \mathcal{T}_{\tau_1,\tau_2}$ the workspace projection $\text{proj}_{\mathcal{W}}(\tau)$ does neither cross the workspace projection $\text{proj}_{\mathcal{W}}(\tau_1)$ nor $\text{proj}_{\mathcal{W}}(\tau_2)$, except at positions where $\text{proj}_{\mathcal{W}}(\tau_1)$ crosses $\text{proj}_{\mathcal{W}}(\tau_2)$. In addition, the weight functions $f_w \in \mathcal{F}_w$, where $f_w : [0, t] \to [0, 1]$ have to lead to feasible action trajectories $\omega \in \Omega$.

Informally, according to Definition 3.1 two trajectories are in the same cluster, if all trajectories *between* them are collision-free. Fig. 3.3 depicts the region of all such states for two sample trajectories. Note that this is different to homotopy, where two paths are equivalent, if there exists *any* collision-free deformation between them. Computation-wise, it is not feasible to generate and evaluate all such inter-trajectories. Therefore, the surface that is spanned by the two trajectories and all the inter-trajectories is approximated and the intersection with the obstacles is evaluated, as shown next.

Polygonal Approximation

Instead of planning and evaluating all inter-trajectories from Definition 3.1 individually, it is chosen to calculate the polygonal hull and evaluate it at once. It is assumed here that the system behaves well, i.e., small changes in the action lead to small changes in the resulting trajectory. Let the trajectory nodes

$$\tau^n = \{\tau_1^n, \ldots, \tau_m^n\}, \quad \tau_j^n \in \mathbb{R}^2 \tag{3.12}$$

be an ordered sequence of m points, which are projected into the workspace from the states of trajectory τ. The trajectory nodes τ^n, in this work, are directly available as the path nodes, as described in Section 3.2.2. Note that the clustering is performed in the workspace $\mathcal{W} \subseteq \mathbb{R}^2$. Then, given two trajectories τ_l and τ_r, where the endpoint of τ_l is to the left of the endpoint of τ_r, a polygon

$$\mathcal{P} = \left\{\tau_{l,1}^n, \ldots, \tau_{l,m}^n, \tau_{c_1,m}^n, \ldots, \tau_{c_k,m}^n, \tau_{r,m}^n, \ldots, \tau_{r,1}^n\right\} \tag{3.13}$$

is created and it is checked whether an obstacle $o \in \mathcal{O}$, or a part of it, lies within $\mathcal{P}$. The vertices $\tau^n_{c,m}$ are trajectory endpoints from a sampled set of trajectories that are in the center between τ_l and τ_r according to Definition 3.1. With the planner described in Section 3.2, the motion model f_m from (3.3) is applied for a discrete number of steps with the actions u from the action set of (3.5), beginning at the start configuration q_S. The polygonal hull is then approximated with the endpoints

$$
\begin{aligned}
\tau^n_{c_1,m} = & \\
& \mathrm{proj}_{\mathcal{W}}\left(f_m\left(\cdots f_m\left(q_S, \frac{k}{k+1}u_{\tau_l,1} + \frac{1}{k+1}u_{\tau_r,1}\right)\cdots, \frac{k}{k+1}u_{\tau_l,m-1} + \frac{1}{k+1}u_{\tau_r,m-1}\right)\right) \\
\vdots & \\
\tau^n_{c_k,m} = & \\
& \mathrm{proj}_{\mathcal{W}}\left(f_m\left(\cdots f_m\left(q_S, \frac{1}{k+1}u_{\tau_l,1} + \frac{k}{k+1}u_{\tau_r,1}\right)\cdots, \frac{1}{k+1}u_{\tau_l,m-1} + \frac{k}{k+1}u_{\tau_r,m-1}\right)\right)
\end{aligned}
\tag{3.14}
$$

of k trajectories. Using the given predicate, the trajectories are clustered according to the method described next.

3.3.2 Clustering with a Binary Equivalence Predicate

This section describes the clustering procedure making use of the binary Boolean-valued equivalence definition given previously.

Algorithm and Computational Complexity

The method is given in Algorithm 3.2. It assumes non-overlapping clusters, i.e., every data point belongs uniquely to one cluster according to the predicate. Hence, the result of the clustering is independent of any permutation of $\mathcal{T}$ and thus the order in which the predicate is applied. Non-overlapping clusters are discussed in the following section.

The computational complexity is $\mathrm{O}\big(n_\tau n_c n_o\big)$, where n_τ is the number of trajectories, n_c the number of clusters, and n_o the number of obstacles. The algorithm is linear in the number of trajectories. Since the number of clusters and the number of obstacles (grid maps are discussed in the following) is usually much smaller than the number of trajectories in real-world applications, the approach scales well. It therefore potentially provides a huge performance increase over other approaches, such as single-linkage agglomerative clustering, which has a computational complexity of $\mathrm{O}\big(n_\tau^2\big)$ [51], or DBSCAN, which runs in $\mathrm{O}\big(n_\tau \log n_\tau\big)$ [54], if an indexing structure is used for the region queries. The computational complexity of the homotopy-like equivalence clustering from [88] is $\mathrm{O}\big(n_\tau^2\big)$.

It is noted that outliers do not exist in this method, since every trajectory is generated using a motion planning algorithm and is thus a valid data point. Hence, a single trajectory can lead to a new cluster, such as with homotopy.

Algorithm 3.2 Environment-based Trajectory Clustering[1]

Input: Set of trajectories $\mathcal{T}$, action trajectories Ω, obstacles $\mathcal{O}$
Output: Clusters C_j, cluster representatives $\mathcal{T}_{\text{rep}}$

```
C_0 ← {τ_0}
T_rep ← {τ_0}
for i ← 1 to |T| − 1 do
  for j ← 0 to |T_rep| − 1 do
    b ← true
    for all obstacles o ∈ O do
      if pred(τ_i, τ_j) = false then
        b ← false
        if j = |T_rep| − 1 then
          T_rep ← T_rep ∪ τ_i
          C_{j+1} ← {τ_i}
        end if
        break
      end if
    end for
    if b = true then
      C_j ← C_j ∪ τ_i
      break
    end if
  end for
end for
```

Clustering in Grid-based Environments using Point-in-Polygon Tests

Grid-based environment representations are challenging for any kind of intersection tests, such as collision checking, discussed in Chapter 4. Contrary to polygon shapes, a single object may be represented by a large number of cells, and an object is not directly identifiable as such. Resolution-exact collision checking therefore needs to examine all individual cells. As Algorithm 3.2 also scales linearly with the number of objects, the computational performance drops significantly, if every cell is regarded as one object, as there are easily thousands of occupied grid cells even in moderately sized grids. Therefore, the occupied cells are grouped into blobs using connected-component analysis. An optimized GPU-based connected-component labeling algorithm [28] is used to group the individual cells to objects. Then, for every blob, one reference point is chosen, which is used as representative for the object. This is similar to [22], but in this approach also small objects, such as traffic cones, are desired to get assigned reference points.

After the extraction of a number of reference points, the predicate is tested. In the implementation, a polygon $\mathcal{P}$ is created from two trajectories, as given by (3.13) and (3.14), and it is checked whether a reference point lies inside of $\mathcal{P}$. The point-in-polygon check is performed with the odd-even-rule, as described in [69]. Note that with the odd-even rule, the polygon is not allowed to intersect itself. The paths must therefore be Jordan curves.

[1] This algorithm is a revised and corrected version from the one presented in [9].

This does not mean, however, that τ_l is not allowed to intersect τ_r or vice-versa. If this cannot be guaranteed, then the non-zero-winding-number-rule [69] can be chosen.

Furthermore, the number of blobs is decreased by applying binary morphological dilation [150] with a circle with diameter equal to the vehicle width. In fact, according to Definition 3.1, since the inter-trajectories have to be collision-free, the check actually needs to be done in the configuration space [96] rather than the work space. However, since the trajectories τ_l and τ_r forming the polygon are guaranteed to be collision-free, and since the inter-trajectories are just used to close the polygon for the predicate evaluation, the approximation is sufficient. Additionally, dilating the obstacle map with a circle equal to the vehicle width is already closer to the configuration space obstacles.

The algorithm was implemented on a GPU. After the next cluster representative τ_{rep} is chosen, polygons are created between τ_{rep} and all other yet unclassified trajectories in parallel. The polygons are also checked in parallel against the reference points. This process is repeated until all trajectories are assigned to a cluster. Thus, the number of GPU passes corresponds to the number of clusters.

3.3.3 Trajectory Clustering with Overlapping Clusters

Until now, it was assumed that the problem exhibits non-overlapping clusters. Even if all paths are of the same length, this can, however, not be guaranteed. The result of Algorithm 3.2 is not independent of permutations of the set of trajectories $\mathcal{T}$, if overlapping clusters are present. However, it is desired that the clustering is unique, given a fixed set of trajectories and a fixed set of obstacles. If multiple possible clustering results exist, the question arises about the optimality of a clustering.

Definition 3.2 *A trajectory clustering exhibiting the following properties*

1. *for every two trajectories τ and τ' from the same cluster C_i the predicate holds, i.e.,*

$$\forall \tau, \tau' \in C_i : \text{pred}(\tau, \tau') = \text{true}\,. \tag{3.15}$$

2. *for every two clusters C_i and C_j there exist two trajectories $\tau_i \in C_i$ and $\tau_j \in C_j$ such that the predicate does not hold, i.e.,*

$$\exists \tau_i \in C_i \; \exists \tau_j \in C_j : \text{pred}(\tau_i, \tau_j) = \text{false}\,. \tag{3.16}$$

is considered to be optimal in order to extract the principal moving directions under a given predicate pred.

A clustering according to Definition 3.2 guarantees that for all trajectories within each cluster the predicate holds, i.e., it exhibits a strong intra-cluster connectivity, while the total number of individual clusters is minimized. This clustering is unique, except for data elements that are within the intersection of two or more clusters. It can be verified that in the case of non-overlapping clusters, Algorithm 3.2 yields a clustering according to this definition. In the case of overlapping clusters, however, it is usually not sufficient to iterate

over all trajectories only once. Instead, every trajectory has to be checked with every other trajectory leading to a quadratic complexity in the number of trajectories.

For many applications, as for the one in this work, this is not computationally-feasible. It can be observed, however, according to Definition 3.2, that

$$\forall \tau \in C_i \setminus \bigcup_{j \neq i} C_j \; \exists \tau' \in \bigcup_{j \neq i} C_j : \mathrm{pred}(\tau, \tau') = \mathrm{false}\,. \tag{3.17}$$

For every trajectory, which is only part of a single cluster C_i and not simultaneously part of a second cluster C_j, and hence not in the intersection $C_i \cap C_j$, there exists a trajectory in C_j for which the predicate does not hold. This leads to the creation of a new cluster and guarantees that all clusters will be found.

As a consequence, if all cluster representatives $\tau \in \mathcal{T}_{\mathrm{rep}}$ are chosen such that

$$\forall \tau \in \mathcal{T}_{\mathrm{rep}} : \tau \in C_i \rightarrow \tau \notin C_j \forall j, \tag{3.18}$$

all trajectories need to be visited only once.

Unfortunately though, without knowing the optimal clustering, this cannot be guaranteed. However, with the use of a heuristic, the probability of picking a representative according to (3.18) can be increased. Since the inter-trajectories are sampled by averaging the controls (3.14), the trajectory with the minimum overall absolute curvature κ

$$\tau_{c_j} = \underset{\tau \in \mathcal{T} \setminus C_0 \cup \ldots \cup C_{j-1}}{\arg\min} \int_0^t |\kappa_\tau(t')| \, \mathrm{d}t' \tag{3.19}$$

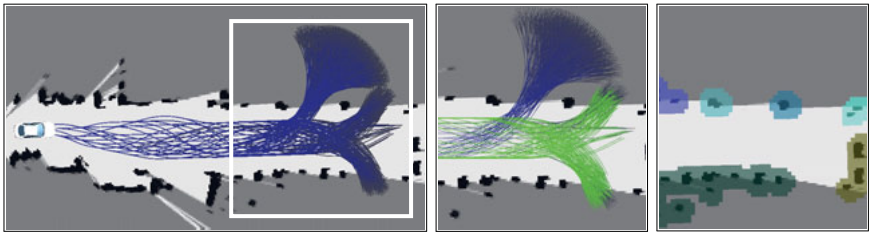
is chosen as representative of cluster C_j. In all tested scenarios, with the above heuristic, a suboptimal clustering due to a false trajectory representative, was not observed.

If trajectories, which may belong to multiple clusters according to the predicate, are assigned to a random one, a non-intuitive clustering may result, as shown in Figure 3.4d. There, the trajectories are assigned to the first cluster, for which the predicate holds. For the problem of extracting the principal moving directions based on a set of trajectories $\mathcal{T}$, this has no influence, since only the cluster representatives $\mathcal{T}_{\mathrm{rep}}$ are used.

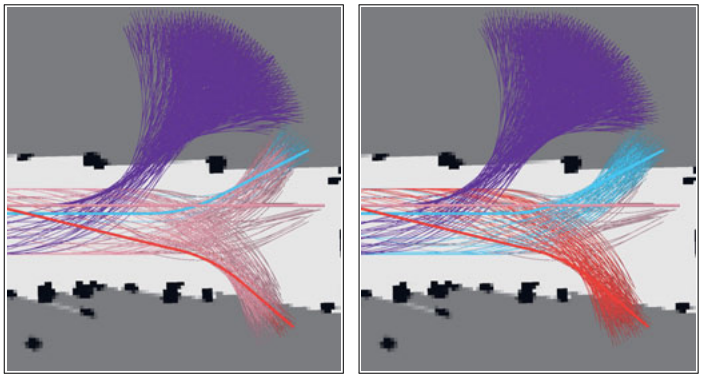
If it does matter, however, how the ambiguities are resolved, nearest-neighbor assignment is one option, which yielded good results in the evaluations. First, all ambiguous trajectories that may belong to more than a single cluster are extracted as the set $\mathcal{T}_{\mathrm{amb}}$, as shown in Figure 3.4b. Then, $\forall \tau \in \mathcal{T}_{\mathrm{amb}}$, the nearest trajectory $\tau' \in \mathcal{T} \setminus \mathcal{T}_{\mathrm{amb}}$ is searched for and τ gets assigned the cluster of τ', shown in Figure 3.4e. The distance measure for the nearest neighbor assignment

$$d(\tau_1, \tau_2) = \frac{1}{m} \sum_{i=1}^{m} ||\tau_{1,i}^n - \tau_{2,i}^n||, \tag{3.20}$$

was the average Euclidean distance between the corresponding path nodes.



(a) Set of paths $\mathcal{T}$. **(b)** Ambiguous paths. **(c)** Blobs.



(d) First-hit assignment. **(e)** Nearest-neighbor assignment.

Figure 3.4: If paths, for which the predicate evaluates to true for more than one cluster (b), are assigned to the first cluster where the predicate holds (d), less-intuitive clusterings may result, compared to assigning them to the nearest cluster (e). Path representatives are shown in bold.

3.4 Results

Finally, results are shown. In Figure 3.5, the paths planned with the path planner presented in Section 3.2 are clustered with the approach proposed in Section 3.3. An occupancy grid is dilated by a circle of vehicle width, blobs are extracted with connected-component labeling, and one reference point per blob is used to cluster the paths with point-in-polygon tests. Note that lateral acceleration constraints are disabled in the path planner for the results in order to yield more challenging scenarios for the clustering approach.

Furthermore, the clustering approach is compared to a standard agglomerative hierarchical clustering and to k-means clustering. The first scenario, given in Figure 3.6, shows an indoor parking lot. Since the sensors cannot observe around turns, the planner needs to be allowed to plan into unknown areas, i.e., where the occupancy probability equals 0.5. The paths that lead to the right of the vehicle are grouped into two clusters with the hierarchical clustering, whereas they are grouped into one cluster, as they represent one topological direction, in the proposed approach. Note that the closest paths between the

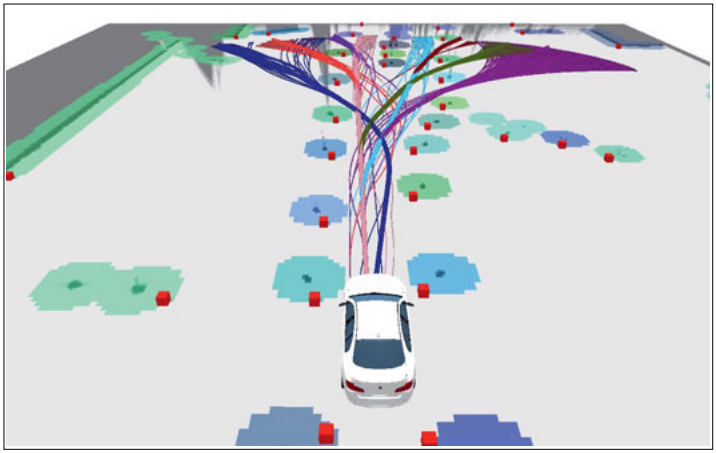


Figure 3.5: Proposed method for the detection of principal moving directions. Equal length paths are clustered based on reference points (red cubes), which are extracted from blobs of an occupancy grid dilated by a circle with diameter equal to the vehicle width.

red and the pink cluster are closer, according to (3.20), than the closest two paths between the purple and the cyan cluster. Hence, no matter how the distance threshold is modified, the same clustering cannot be achieved with hierarchical clustering. Also shown in dark blue are paths from the planner that have not (yet) reached the goal length.

The second scenario, given in Figure 3.7, shows a simulated road construction site with a road junction marked with traffic cones. Both true branches, the pink and the purple cluster, are correctly determined in the proposed clustering, whereas they are in the same cluster, together with an additional path bundle to the left, in the hierarchical clustering. Note that the number of clusters is equal, but the clustering result differs substantially. In this case, with the hierarchical clustering, it may even happen that none of the two valid principal moving directions are detected, because one cluster representative may be chosen out of the path bundle to the left of the pink cluster.

The approach is also compared quantitatively. It is elaborated if the same clustering can be achieved with hierarchical or k-means clustering. The distance similarity was calculated again with (3.20). To this end, the *inter* cluster distances, i.e., the distances of paths from different clusters, and the *intra* cluster distances, i.e., the distances of paths of the same cluster, of the proposed method are analyzed. Examined are the first 200 frames of the scenario from Figure 3.7. In Figure 3.8, the minimum intra cluster path distance, needed for an agglomerative hierarchical cluster approach to yield the same intra cluster connectivity, is plotted. In other words, the minimum threshold value needed, in order to assure that all paths within each cluster from the approach proposed, are grouped together. Also plotted in Figure 3.8, is the minimum inter cluster distance. In other words, the maximum threshold possible so that no two clusters, from the result of the proposed clustering, are

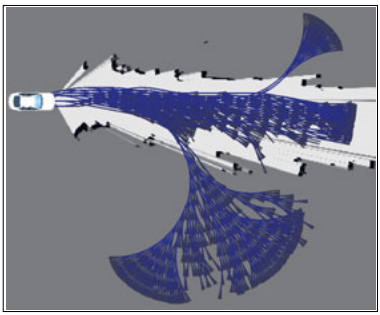
Table 3.1: Computational performance in different scenarios.

	Scenario 1	Scenario 2	Scenario 3
Paths	100	1000	5000
Path nodes	10	20	40
Occupied cells / blobs	1139 / 11	1645 / 16	2322 / 13
Multi-cluster paths	0	0	1016
Nearest neighbor cluster changes	0	0	833
Clusters hierarchical / proposed	1 / 1	4 / 7	5 / 6
Hierarchical clustering (ms)	0.48	12.06	401.77
Proposed clustering (ms)	1.96	3.85	7.61
Proposed with NN assignment (ms)	2.73	5.28	19.88

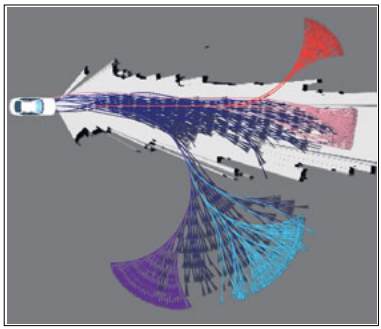
merged together. If the minimum threshold to assure intra cluster connectivity (bright line) is above the maximum threshold possible to assure inter cluster separation (dark line), no matter how the threshold is chosen, the same clustering cannot be achieved.

Furthermore, the approach is compared to k-means clustering. Again, the result of the proposed method is used to calculate the means. All paths are then clustered accordingly. Figure 3.9 gives the percentage of paths that are grouped differently than the proposed clustering. Hence, even under perfect initial conditions, i.e., with given number of clusters k and given means, the paths can rarely be clustered the same way.

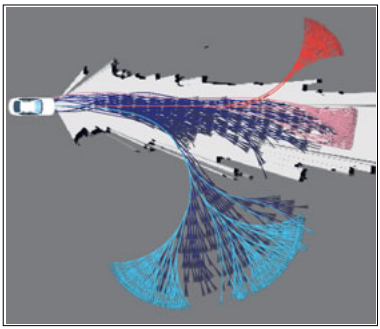
Finally, the computational performance is analyzed in Table 3.1. The hardware is given in Appendix A.2. Both algorithms were implemented with Nvidia CUDA. In the implementation of the hierarchical clustering, the $n_\tau(n_\tau - 1)/2$ distance calculations as well as the sorting was performed on graphics hardware in parallel, while cluster merging, which is linear in the number of paths, was performed on the CPU. In the implementation of the proposed approach, all compute-intensive calculations are calculated on the GPU. The runtime includes the morphological dilation of the occupancy grid and the connected component analysis [28]. As expected, the proposed approach, although slightly slower in simple scenarios, scales well with the number of paths and fits the real-time requirements. With 5000 paths in scenario 3, it is 20 times faster than the hierarchical clustering, if the ambiguous paths are assigned to their nearest neighbor, and even over 50 times faster, if only the principal moving directions are of interest.



(a) Unclustered paths.

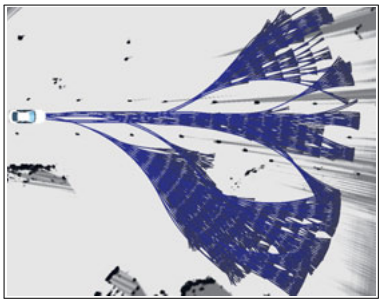


(c) Hierarchical clustering.

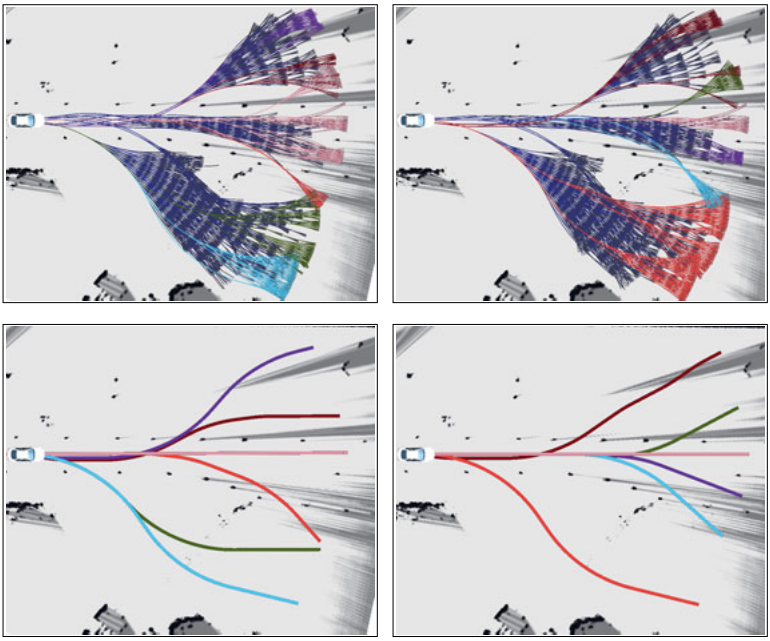


(d) Proposed clustering.

Figure 3.6: Comparison between agglomerative hierarchical clustering based on an optimized distance threshold and the proposed approach. Scenario shows an indoor parking lot with walls and parked vehicles.



(a) Unclustered paths.



(c) Hierarchical clustering.

(d) Proposed clustering.

Figure 3.7: Comparison between agglomerative hierarchical clustering based on an optimized distance threshold and the proposed approach. Scenario shows a simulated road construction site on an open space with traffic cones.

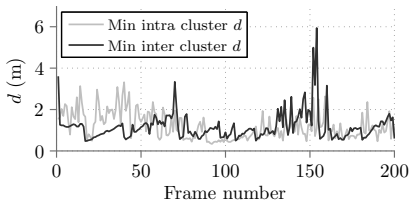


Figure 3.8: Difference between proposed and hierarchical clustering.

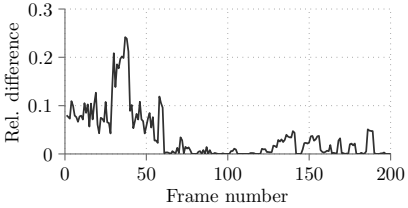


Figure 3.9: Difference between proposed and k-means clustering.

3.5 Summary

In this chapter, the problem of detecting the principal moving directions in the local environment of the vehicle has been investigated. The principal moving directions are important semantic information. With their use, estimating the road courses is greatly simplified, as described in Chapter 5. They separate the roadways into a left region, containing the left boundary, and a right region, containing the right boundary.

The presented approach uses local path planning and path clustering to detect the main directions through the environment. Other than direct estimation techniques on sensor data, the robot shape and simplified dynamic constraints are considered. Therefore, the method works in arbitrarily-shaped road environments and also in the presence of sparse road boundaries, where few true positive data points are available. Additionally, the number of principal moving directions is implicitly estimated by clustering the local paths and does not need to be known a-priori.

Two novel contributions have been presented in this chapter. First, a local, graph-based path planner, which combines A* and RRT, has been proposed. It is designed to efficiently sample a set of paths in the absence of goal poses, where all paths equal a certain predefined length. Due to the combination and the use of motion primitives, the planner is resolution-complete, optimal under the graph discretization, and anytime uniformly exploring.

The sampled set of paths then undergoes clustering to reduce the paths to the principal moving directions. Similar to path homotopy, the clustering depends on the objects in the environment, rather than on distance-based similarity measures, like used in standard clustering algorithms. Differently though, the endpoints of the paths do not need to match. Instead, a closed surface is created by sampling inter-paths and it is checked against the obstacles using point-in-polygon tests. The proposed clustering method is linear in the number of paths. It has been compared to a GPU-based hierarchical clustering and the performance greatly outperforms it by an order of magnitude.

Sampling a set of paths under real-time constraints is computationally demanding. In addition to collision checking, the costs of paths need to be evaluated. In the next chapter collision checking and cost evaluation given workspace cost maps is examined.

4 Configuration Space Costs: Cost Evaluation on Workspace Cost Maps

Collision checking and cost evaluation of paths and trajectories are the major computational bottlenecks in current real world motion planning applications. They may even take up to 99 % of the total time of a planner [136], thus making them the most significant computational problem for real-time motion planners. In particular with grid-based environment representations, the computational burden is huge. Compared to polygonal models, no compact representation of the objects in the local environment of the robot exists. Since computing a *good* path or trajectory is usually important in practical applications, some form of cost representation must be used. Often, costs can be intuitively designed according to the current environment, i.e., the workspace, in form of a map. This chapter shows how workspace-based costs are evaluated incorporating the robot shape in a particular configuration. Moreover, it presents precalculation methods to quickly evaluate both, costs and collisions, providing performance increases for motion planning algorithms. It is based on work that has been presented in [7] in the context of this thesis.

4.1 Introduction

The particular way of collision checking and cost evaluation is dependent on the environment representation. Moreover, it can be divided into checking against the static world, such as static obstacles, and checking against the dynamic world, such as moving objects, both of which are quite different. In collision avoidance with dynamic objects, such as other vehicles, prediction is a key problem. Since for most cases a precise prediction cannot be guaranteed, as the control commands of the other vehicles are not known, over-approximations and defensive collision checking strategies are desired. Precision is usually of minor importance, and the time dimension increases the complexity. Often, it is discretized but there are also approaches providing algebraic solutions [97]. On the other hand, in collision checking against the static environment, precision typically is very important and approximations are restricted. Consider, e.g., automated parking maneuvers or autonomous navigation in cluttered, narrow road environments. This chapter focuses on the problem of evaluating collisions and costs against the static environment.

4.1.1 State of the Art

As described above, current motion planning algorithms spend a substantial computational part with collision detection [96, 103, 141]. It has been studied for decades, and many approaches have been proposed to solve it [94, 96]. Still, computational performance is

of major concern even in state of the art implementations, especially if the obstacles are represented with grid maps, the representation of choice of the recent years as discussed in Chapter 2, and not by polygons as done by many earlier planners. In addition to collision checking, cost evaluations are also computationally expensive, but required if path quality is of concern, which usually is in any real-world application. For resolution-exact collision checking and for workspace cost evaluation, as will be described, every cell under the robot footprint has to be evaluated for every configuration of every path. Just one footprint of the robot, i.e., the cells that need to be checked for a single state, with a 10 cm grid resolution and a 5 m $\times$ 2 m vehicle, comprises a whopping 1000 cells, underlining the huge computational complexity of the problem.

Since evaluating a single configuration for collision and cost may already require examining a large number of cells, precalculation methods have been studied. Depending on the planning complexity, the computational burden can dramatically be reduced by precalculating *configuration space obstacles* [93, 96] based on the workspace obstacles and the robot shape. With configuration space obstacles, the check of a complete configuration reduces to a single point check or array look-up. Often in the literature, circular robot shapes are assumed, as they are rotation-invariant, which greatly reduces the amount of computation. A vehicle, however, cannot be approximated well by one circle, and therefore every possible rotation has to be accounted for. The complexity of the calculation of the configuration space obstacles increases then tremendously.

Other approaches use bounding volume approximations to deal with the complexity. Such methods quickly process a large number of configurations using shape approximations, while only performing a detailed evaluation if the result remains inconclusive. Instead of calculating the configuration space obstacles for all robot orientations, in [102], only two slices of the configuration space obstacles are computed. One slice comes from dilation with a circle with radius equal to the robot inner radius, and similarly, the other is calculated with the vehicle outer radius, in order to reduce the number of exact collision checks needed. While such an approach works well if the robot shape does not deviate too much from a circle and if the environment is not too narrow, it may become ineffective. Consider a vehicle navigating through a single-lane road construction site. The evaluation with the vehicle outer radius will always lead to a collision, while the evaluation with the vehicle inner radius will not, requiring an expensive detailed evaluation of every configuration. In [181], the robot shape is decomposed into a set of overlapping disks, the obstacle map is dilated with the disk, and a collision is checked by evaluating the configurations at the individual disk centers. The authors further propose to decompose the disks into axis-aligned rectangles and to use a summed area table in order to quickly evaluate them. If a high precision is important, however, a high number of disks is required for the vehicle-to-disk decomposition, and in turn a high number of rectangles for the disk-to-rectangle decomposition, rendering the method ineffective.

As aforementioned, calculating costs of individual configurations is also of central interest for many planning algorithms. There are different kinds of costs, as described in Section 3.2.3. The evaluation of workspace-based costs is similar to collision checking in that it also requires evaluating every cell under the robot footprint. There is very little literature, however, on how such costs can be efficiently evaluated. The shape of the robot

is often simply ignored [30] or the costs are already given in the configuration space [74]. Similar to configuration space obstacles, a structure therefore termed *configuration space costs* can be precomputed that allows the cost evaluation of a complete configuration to be done by a single look-up. The only work found that deals with the calculation of the configuration space costs is that of [85], where an algorithm to compute a slice from the configuration space costs is presented that equals a direct implementation of morphological grayscale dilation known from image processing [150].

4.1.2 Approach and Contribution

This chapter gives theoretical and practical insights on how to efficiently check a large number of configurations for collision and cost on grid maps. *Configuration space costs* are introduced and defined. They are a generalization of the commonly used configuration space obstacles. They allow the cost *and* collision evaluation of a complete robot configuration to be performed using a single look-up. Furthermore, it is shown, how the configuration space costs can be efficiently calculated and two algorithms are proposed:

- Fast Approximate Morphological Grayscale Dilation (FAMOD): an approximate algorithm based on convolution, which is independent of the size and the shape of the robot mask, and

- van Herk-Gil-Werman-360 (vHGW-360): a resolution-exact method based on the van Herk-Gil-Werman dilation algorithm, applicable for rectangular robot shapes.

Both methods were implemented on graphics hardware to demonstrate the performance gain for motion planning systems. In addition to the formulation of configuration space costs and their efficient calculation, it is shown how whole paths can be efficiently evaluated without missing a cell or checking a single cell multiple times.

The rest of this chapter is structured as follows. First, in Section 4.2 the fundamentals of collision checking are revisited and it is discussed how binary dilation and convolution relate to each other. These observations are then used to generalize configuration space obstacles to configuration space costs. Section 4.3 and Section 4.4 present FAMOD and vHGW-360 respectively. Since evaluating a path requires examining multiple configurations, this topic is investigated in Section 4.5. Finally, Section 4.6 shows results.

4.2 Grid-based Collision Checking and Cost Evaluation

This section revisits some fundamentals about collision checking, as the rest of this chapter highly depends on them. It then shows how the concepts of collision checking extend to cost evaluation and configuration space costs are introduced.

4.2.1 Collision Checking Fundamentals

A path $\tau : [0, l] \rightarrow \mathcal{C}$ is collision-free, if all of its configurations $q \in \mathcal{C}$ from the configuration space $\mathcal{C}$ are collision-free, i.e.,

$$\forall l' \in [0, l] : \tau(l') \notin \mathcal{C}_{\text{obs}}. \tag{4.1}$$

The configuration space obstacles

$$\mathcal{C}_{\text{obs}} = \{q \in \mathcal{C} \mid S_q \cap \mathcal{O} \neq \emptyset\} \tag{4.2}$$

is the set of all configurations q of the robot region S, which lead to a collision with the obstacles $\mathcal{O}$. In the following, it is focused on a grid-based representation of the obstacles, denoted by a binary grid B, and 3-dimensional configurations q with a 2-dimensional position component x and an orientation component θ.

If the orientation θ of the robot is fixed, or the robot shape is rotation-invariant, $\mathcal{C}_{\text{obs}}$ can be computed using the Minkowski difference $\ominus$ of B and S

$$\mathcal{C}^{\theta}_{\text{obs}} = B \ominus S_\theta = \{b - s \mid b \in B, s \in S_\theta\}, \tag{4.3}$$

for all occupied cells $b \in B$ and all elements of the robot mask $s \in S$, or equivalently with the Minkowski sum $\oplus$,

$$\mathcal{C}^{\theta}_{\text{obs}} = B \oplus -S_\theta = \{b + s \mid b \in B, s \in -S_\theta\}. \tag{4.4}$$

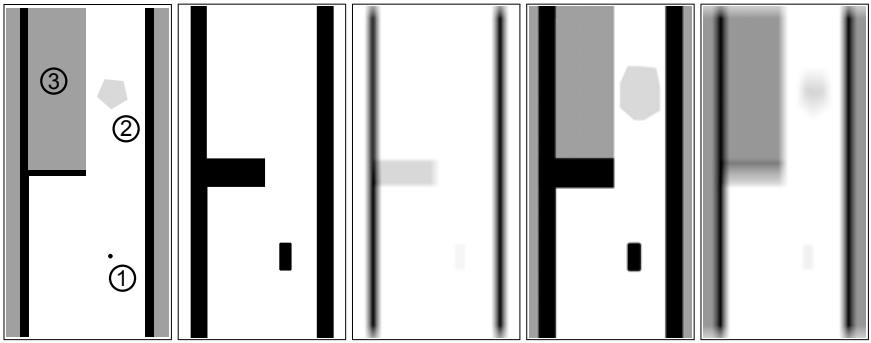
The Minkowski sum may also be calculated by binary morphological dilation known from image processing, as these two operations are equivalent [59, 109]. This is useful, since the obstacles are often represented by discretized grid structures similar to images, such as in occupancy grids. One slice of the configuration space obstacles, which corresponds to one orientation of the robot, is calculated by binary dilation of the grid B with the robot mask S, usually referred to as the structuring element, kernel, or footprint, as

$$\mathcal{C}^{\theta}_{\text{obs}} = B \oplus (R_{180}\, S_\theta) = \bigcup_{b \in B} R_{180}\, S_{b,\theta}. \tag{4.5}$$

The obstacle grid is expanded by placing the origin of the structuring element at every occupied cell of B. The rotation of the binary mask by 180°, given by the matrix R_{180}, is only required if the robot shape is asymmetric. If the vehicle shape is approximated by a rectangle, an approximation that is usually fine enough given the discretization errors of the obstacle representation, the rotation is not necessary. It is thus dropped in the following equations. Hence, an alternative formulation of the configuration space obstacles to (4.2) is given with the use of binary dilation from (4.5)

$$\mathcal{C}_{\text{obs}} = \bigcup_{\theta} \mathcal{C}^{\theta}_{\text{obs}} = \bigcup_{\theta} B \oplus S_\theta. \tag{4.6}$$

Throughout the literature, the mathematical operation for the calculation of $\mathcal{C}_{\text{obs}}$ is consistently denoted as convolution, e.g., [34, 83, 102, 133, 156]. However, although con-



(a) Occupancy grid **(b)** Bin. dilation **(c)** Bin. conv. **(d)** Graysc. dilation **(e)** Graysc. conv.

Figure 4.1: Comparison between binary dilation (b), binary convolution (c), grayscale dilation (d), and grayscale convolution (e) of an occupancy grid (a) with a vertically-oriented rectangular mask. Shown are: a small obstacle ① with $p(o) = 1$, an uncertain area ② with $p(o) = 0.25$, and an unknown area ③ with $p(o) = 0.5$.

volution and dilation are similar, in the latter, a union operator is used, whereas the convolution of two functions f_1 and f_2

$$(f_1 * f_2)(j) = \sum_i f_1(i) f_2(j - i) \tag{4.7}$$

is given by their weighted average. To underline the difference, Figure 4.1 shows both operations on a thresholded occupancy grid. Binary dilation, Figure 4.1b, and convolution, Figure 4.1c, are performed using the same vertically-oriented rectangular shape. Binary dilation correctly results in a collision map that is independent of the number of individual cell collisions under the footprint, which are clearly visible in the output of the convolution due to the sum operator in (4.7). Low occupancy probabilities are visualized by bright gray levels and high occupancy probabilities by dark gray levels. The convolution is visualized by linearly mapping back the range of the output into the original image range.

The output of the convolution, however, is easily transformed to equal the result of binary dilation by applying a simple function h

$$B \oplus S = h(B * S), \text{ with} \tag{4.8}$$

$$h(j) = \begin{cases} 0 & j = 0 \\ 1 & j > 0. \end{cases} \tag{4.9}$$

This observation is often exploited for the calculation of the configuration space obstacles, such as in [64]. It allows reducing the computational complexity to be independent of the size and the shape of the mask due to the convolution theorem [83]. Although differ-

entiating between dilation and convolution may appear merely of formal nature without practical relevance, it will become crucial in the more general case of costs as shown next.

4.2.2 Extending Collision Checking to Cost Evaluation

Evaluating the cost of one particular robot configuration on a grid-based cost map is similar to checking it for collision. It also requires examining all cells under the robot footprint at the current position and with the current orientation and reducing them to a single value. The cost maps that are investigated are grid-based workspace cost maps, i.e., maps that are defined in the work space $\mathcal{W}$ of the robot. Here, 2-D workspaces are assumed as well as non-negative cost values. Workspace cost maps represent the cost of traversing a certain area in the world. They may be arbitrary user-designed maps, such as inverse distances to the nearest obstacles, so that a robot prefers traversing areas of high clearance.

In the literature, no common consensus of how costs from workspace cost maps are reduced to a configuration cost exists. Sometimes the sum is used, e.g., in [129], which possibly comes from the slight misuse of the term convolution for the calculation of the configuration space obstacles. Here, it is proposed to use the maximum operator, and it is the aim of this section to motivate the choice. It is noted again that the observations only target single configuration costs out of workspace cost maps. For calculating the overall path cost, the sum of individual cost values is indeed more reliable than the maximum [74].

In Figure 4.2 the sum and the maximum operator are compared in different situations. In Figure 4.2a the workspace costs under the mask all equal 0.2, and the sum and the maximum result in 1.2 and 0.2 respectively. In Figure 4.2b all cells under the mask equal the lowest cost value 0, while one cell equals the highest cost value 1. The sum and the maximum both result in 1.0. Comparing Figure 4.2a to Figure 4.2b allows making the following observation:

Observation 4.1 *With the sum operator, traversing an area that is uniformly rather cheap, may result in a higher cost than an area with singular high cost values.*

Usually, high cost values represent dangerous areas, such as a collision with an object, while low cost values represent safe areas. With the sum operator, and analogously with the average, no unique distinctions are possible. Another example is depicted in Figure 4.2c. It shows a cost map that represents the inverse distance to obstacles. The cost values range from 0 to 6, where 6 represents a collision and 0 the minimum cost value. The distance metric is the Manhattan distance. The vertical mask has a lower sum value than the horizontal mask even though the vertical shape is closer to an obstacle. Hence, the sum, and thus also the average, do not yield appropriate results. It is noted that the focus is not on calculating collision probabilities, such as in [12], but on arbitrary costs.

In the following, the configuration space costs are defined with the use of the maximum operator motivated above. Again, let $q \in \mathcal{C}$ be a configuration in the configuration space,

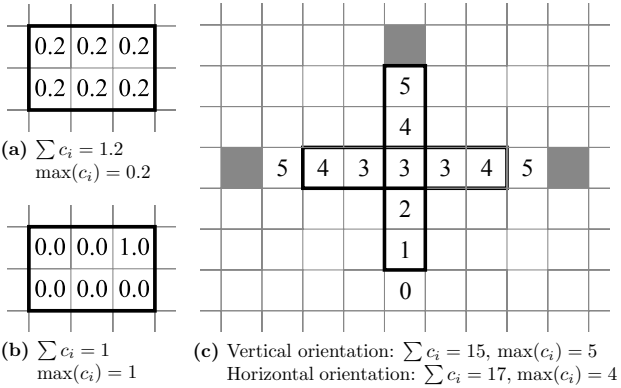


Figure 4.2: Comparison between the sum and the maximum value for the calculation of the cost of a particular configuration under the robot footprint.

S_q the structuring element at configuration q, and M be a grayscale cost map. Then the configuration space costs

$$\mathcal{C}_{\text{costs}} : \begin{cases} \mathcal{C} \to \mathbb{R}_0^+ \\ q \mapsto (M \oplus S_q) \end{cases} \tag{4.10}$$

are defined with the use of morphological grayscale dilation

$$M \oplus S_{j,\theta} = \max(M(j-s) + S_\theta(s) \mid s \in S_\theta). \tag{4.11}$$

Since in this case, the structuring element S is flat, i.e., it is a mask that only holds values of 0, grayscale dilation can be written more compactly as

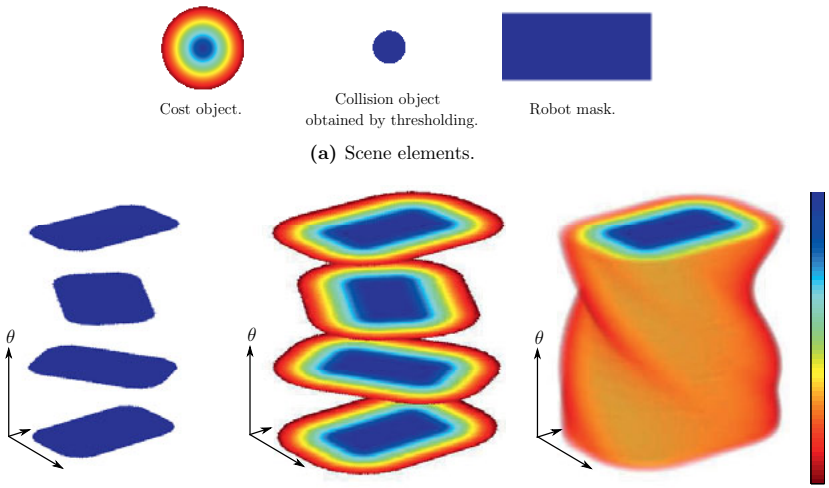
$$M \oplus S_{j,\theta} = \max_{s \in S_\theta} M(j-s). \tag{4.12}$$

Morphological grayscale dilation is a generalization of binary dilation. Apart from the motivation for the maximum operator from above, its use for the calculation for $\mathcal{C}_{\text{costs}}$ therefore comes naturally, since the configuration space costs are a generalization of the configuration space obstacles, which are computed using binary dilation, as discussed above. The highest cost value of $\mathcal{C}_{\text{costs}}$ simply represents a collision. For completeness, the configuration space obstacles

$$\mathcal{C}_{\text{obs}} : \begin{cases} \mathcal{C} \to \mathbb{B} \subset \mathbb{R} \\ q \mapsto (B \oplus S_q), \end{cases} \tag{4.13}$$

are also represented using (4.12), where $\mathbb{B}$ denotes the space of binary numbers.

Figure 4.1d shows the result of the grayscale dilation with the same binary rectangular mask and on the same map as used in the previous section. Also given in Figure 4.1e for completeness, is the grayscale convolution representing the sum operator. Contrary



(a) Scene elements.

(b) Slices at 0, 60, 120, 180 degrees of orientation of $\mathcal{C}_{\text{obs}}$.

(c) Slices at 0, 60, 120, 180 degrees of orientation of $\mathcal{C}_{\text{costs}}$.

(d) Complete $\mathcal{C}_{\text{costs}}$.

Figure 4.3: Visualization of the configuration space obstacles, (b), and the configuration space costs, (c) and (d), of a circular cost object and a rectangular robot footprint, (a). The cost object is expanded with the robot mask rotated at different orientations θ.

to the binary case of $\mathcal{C}_{\text{obs}}$, the differentiation between convolution and dilation is crucial. Due to the sum operator, it is not possible to differentiate between few cells with high cost values or many cells with low-to-medium values. In Figure 4.1e the small circular obstacle, which has the highest workspace cost value, leads to lower configuration cost values, visualized by brighter gray values, than the pentagon-shaped uncertain area and the unknown area. Figure 4.3 shows a visualization of $\mathcal{C}_{\text{obs}}$ and $\mathcal{C}_{\text{costs}}$ for different orientations of a rectangular robot. In the next sections, two efficient algorithms for the computation of the configuration space costs are presented.

4.3 Fast Approximate Calculation of the Configuration Space Costs for Arbitrary Footprints with FAMOD

This section presents Fast Approximate MOrphological Dilation (FAMOD), a method that uses convolution to calculate the configuration space costs. Due to the convolution theorem, the complexity of a convolution reduces to a single multiplication in the frequency domain. By using convolution to calculate the grayscale dilation, the method is thus independent of the size of the robot mask. Furthermore, FAMOD can be used with arbitrarily-shaped masks, a property of great utility, as for certain shapes, such as rectangles, efficient algorithms already exist, as will be shown in the next section.

4.3.1 Calculation of Grayscale Dilation with Convolution

Morphological dilation is a *nonlinear* image filter. Nonlinear filters, by definition, are those that cannot be represented by a pure convolution [150]. The max operator is responsible for this non-linearity. However, motivated by (4.8) and (4.9), it is shown how dilation can still be computed with convolution in the case of a binary image.

In the following, k denotes the number of obstacle cells of the binary mask $S = \{0,1\}^*$, which holds the robot shape. The minimum and maximum possible value of the result of the convolution of an arbitrary binary grid B with S are

$$\begin{aligned} \min(B * S) &= 0 \quad \text{and} \\ \max(B * S) &= k, \end{aligned} \tag{4.14}$$

if the binary values of B are $\{0,1\}$. With arbitrary values a and b where $a < b$, the lowest and highest possible values are

$$\begin{aligned} \min\big(B^{\{a,b\}} * S\big) &= ka \quad \text{and} \\ \max\big(B^{\{a,b\}} * S\big) &= kb. \end{aligned} \tag{4.15}$$

Using (4.14) and (4.15), a mapping is now derived to re-extract the maximum cost value from the result of the convolution, i.e., from the sum operator. Let the cost map exhibit n different cost values $c_0, \ldots, c_{n-1}$ where $c_0 < \ldots < c_{n-1}$, and let ε denote a small positive number. By transforming the original cost values from the cost map according to the recurrence relation

$$\begin{aligned} g(c_0) &= c_0 \\ g(c_i) &= kg(c_{i-1}) + \varepsilon \end{aligned} \tag{4.16}$$

and by re-transforming the result of the convolution according to the function

$$h(j) = \begin{cases} c_0 & g(c_0) \leq j < g(c_1) \\ \vdots & \\ c_{n-2} & g(c_{n-2}) \leq j < g(c_{n-1}) \\ c_{n-1} & j > g(c_{n-1}) \end{cases} \tag{4.17}$$

morphological grayscale dilation of an arbitrary grayscale image M with a flat structuring element S

$$M \oplus S = h(\, g(M) * S\,) \tag{4.18}$$

is calculated with the convolution operator.

Algorithm 4.1 gives details about how the configuration space costs are calculated with FAMOD. The structuring element is rotated to n_{slices} different rotations, which correspond to the different slices of $\mathcal{C}_{\text{costs}}$.

Algorithm 4.1 FAMOD

Input: Map M, structuring elements $S[\,]$ corresponding to different orientations
Output: Configuration Space Costs $\mathcal{C}_{\text{costs}}[\,]$

1: *// offline*
2: **for** $i \leftarrow 0$ **to** $n_{\text{slices}} - 1$ **do**
3: $\quad S_{\text{FFT}}[i] = \text{FFT}(S[i])$
4: **end for**
5: *// online*
6: $T_1 \leftarrow g(M)$
7: $T_2 \leftarrow \text{FFT}(T_1)$
8: **for** $i \leftarrow 0$ **to** $n_{\text{slices}} - 1$ **do**
9: $\quad T_3 \leftarrow \text{modulate}(T_2, S_{\text{FFT}}[i])$
10: $\quad T_1 \leftarrow \text{iFFT}(T_3)$
11: $\quad \mathcal{C}_{\text{costs}}[i] \leftarrow h(T_1)$
12: **end for**

4.3.2 Practical Considerations

Practical considerations and implementation details are discussed in this section.

Output Range Approximation

It is noted that the function g from (4.16) grows exponentially and the total number of cost values, i.e., the output range, is therefore limited. Depending on the size of the mask and on the range of the cost values in the cost map, it may become necessary to reduce the output range, due to limitations of the used data types. If occupancy grids are used as cost maps, the limitation does not affect a cost-based planner too much. Occupancy values typically converge rather quickly towards 0 or 1 for observable areas, and one is primarily interested in differentiating between free cells, occupied cells, and unsure cells. Hence even with 3 classes, a cost map from an occupancy grid can be approximated well. Apart from a uniform quantization of the original cost values, they may also be quantized non-uniformly to better extract the relevant information. With occupancy grid cost maps, it is sometimes desired to distinguish cells with the value 0.5, as it represents not yet observed cells for most cases. Hence, one class can be designed to exactly represent 0.5, while the other classes may cover larger intervals.

GPU Implementation Details

Dilation and convolution parallelize well, as for every cell of the map the same operations are performed, which are independent of each other. Therefore, these operations are well suited for an implementation on graphics hardware. As mentioned on multiple locations in this thesis, Nvidia CUDA is used to program the GPU. The CUDA Fast Fourier Transform library cuFFT [121] was used in the implementation. The functions g and h from (4.16) and (4.17) are not well suited for an efficient implementation on a graphics card, however,

as they require loops and conditionals inside the CUDA kernel. However, with $c_0 = 0$ the recurrence relation from (4.16) is solved by deriving the generating function.

$$\begin{aligned} g(c_i) &= k\, g(c_{i-1}) + \varepsilon \\ &= \varepsilon\, k^{i-1} + \varepsilon\, k^{i-2} + \cdots + \varepsilon\, k + \varepsilon \\ &= \varepsilon \sum_{j=0}^{i-1} k^j \\ &= \varepsilon \frac{1-k^i}{1-k} \end{aligned} \tag{4.19}$$

The final result is calculated by solving the output j of the convolution for the cost index i

$$\begin{aligned} i &= \left\lfloor \log_k \left[1 - \frac{1-k}{\varepsilon} j \right] \right\rfloor \\ h(j) &= c_i \end{aligned} \tag{4.20}$$

without the need for conditionals, replacing (4.17).

Next, another method to calculate $\mathcal{C}_{\text{costs}}$ is presented that does not have limitations in the output range, but requires the robot shape to be rectangular.

4.4 Efficient Exact Calculation of the Configuration Space Costs for Rectangular Footprints with vHGW-360

In addition to circular robot shapes, rectangular shapes are common, such as exhibited by many wheeled robots, like vehicles. Due to the symmetry of rectangular shapes, the computation of $\mathcal{C}_{\text{costs}}$ is done more efficiently, as shown in the following.

4.4.1 Reducing Computations by Exploiting Symmetry

Rectangular forms exhibit beneficial symmetric properties, which can be exploited in multiple ways to increase the performance. First of all, not all orientations have to be actually computed. If the origin of the structuring element is in the center, the dilation of a mask rotated by an angle θ is equivalent to the dilation of a mask rotated by $\theta + 180$, reducing the number of dilations to a half. If the mask is quadratic, the number of dilations even reduces to a quarter. Moreover, the associativity property of dilation further reduces the number of operations. Hence, if the structuring element

$$S = S_1 \oplus S_2 \tag{4.21}$$

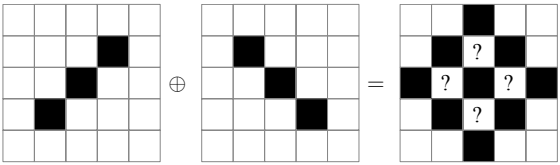


Figure 4.4: Errors in the form of missing cells for structuring element decomposition of non-axis-aligned rectangular shapes.

is the dilation of two masks S_1 and S_2, then the dilation of the map M with the 2-D mask S

$$M \oplus S = (M \oplus S_1) \oplus S_2 \tag{4.22}$$

is equivalent to two dilations with 1-D masks, S_1 and S_2. This reduces the number of operations per cell from $n^S_{\text{rows}} n^S_{\text{cols}}$ to $n^S_{\text{rows}} + n^S_{\text{cols}}$, if n^S_{rows} and n^S_{cols} are the number of rows and columns of S corresponding to the width and the length of the robot shape. The associativity of dilation is an often used property in practical implementations when dealing with large masks [38]. If the mask is, however, not axis-aligned, the decomposition of a 2-D dilation into two 1-D dilations leads to errors, as shown in Figure 4.4. Hence, not the mask is rotated into different orientations θ representing the different slices of $\mathcal{C}_{\text{costs}}$ but the grid is rotated, i.e.,

$$\mathcal{C}^{\theta}_{\text{costs}} = M \oplus R_{\theta}(S) = R_{\theta}(R_{-\theta}(M) \oplus S). \tag{4.23}$$

Although (4.23) requires two grid rotations for every dilation, again due to symmetry considerations, the total number of rotations is reduced. The result of the dilations of orientation θ and $\theta + 90$ are equal except that the width and the length of the robot shape are exchanged. They are thus computed simultaneously with one pair of rotations. Even though a significant number of grid rotations is still needed to use structuring element decomposition, it is still of great advantage, since specialized algorithms for the dilation with horizontal and vertical 1-D masks exist, as shown next.

4.4.2 The vHGW Algorithm

Several algorithms have been proposed for the efficient calculation of the special case of morphological dilation with an axis-aligned 1-D structuring element, e.g., [47, 99, 162]. One of the most famous methods is the van Herk-Gil-Werman (vHGW) algorithm [60, 163]. In addition to its popularity, it has been shown to perform well on graphics hardware [50, 159].

The reason for the popularity of vHGW is its property of being independent of the size of the structuring element. It needs a constant $3 - 4/n^S_{\text{pixels}}$ comparisons per pixel of the image M, where n^S_{pixels} denotes the number of pixels of the structuring element. It is required to be uneven. To this end, M is partitioned into non-overlapping blocks of size n^S_{pixels}. For every block, a larger window of size $2n^S_{\text{pixels}} - 1$ is centered on the block. It is

Algorithm 4.2 vHGW-360

Input: Map M, robot length and width in cells n^S_{rows} and n^S_{cols}
Output: Configuration Space Costs $\mathcal{C}_{\text{costs}}[\,]$

for $i \leftarrow 0$ **to** $n_{\text{slices}}/2 - 1$ **do**
  $T_1 \leftarrow \text{rotate}(M, 180i/n_{\text{slices}})$
  $\mathcal{C}_{\text{costs}}[i] \leftarrow \text{vHGW}(T_1, n^S_{\text{rows}})$
  $\mathcal{C}_{\text{costs}}[(n_{\text{slices}}/2) + i] \leftarrow \text{vHGW}(T_1, n^S_{\text{cols}})$
end for
for $i \leftarrow 0$ **to** $n_{\text{slices}} - 1$ **do**
  $T_1 \leftarrow \text{rotate}(\mathcal{C}_{\text{costs}}[i], 90)$
  if $i < n_{\text{slices}}/2$ **then**
    $T_2 \leftarrow \text{vHGW}(T_1, n^S_{\text{rows}})$
    $\mathcal{C}_{\text{costs}}[i] \leftarrow \text{rotate}(T_2, -(180i/n_{\text{slices}}) - 90)$
  else
    $T_2 \leftarrow \text{vHGW}(T_1, n^S_{\text{cols}})$
    $\mathcal{C}_{\text{costs}}[i] \leftarrow \text{rotate}(T_2, -(180i/n_{\text{slices}}))$
  end if
end for

used to calculate two cumulative maximum arrays A_l and A_r independently of all other blocks. Let j denote the center pixel of a particular block of the linearized image M, then

$$\begin{aligned} A_l[i] &= \max(A_l[i-1], M[j-i]) \text{ and} \\ A_r[i] &= \max(A_r[i-1], M[j+i]) \text{ with} \\ A_l[0] &= A_r[0] = M[j], \end{aligned} \tag{4.24}$$

$\forall i = 0 : n^S_{\text{pixels}} - 1$. The output then results in

$$(M \oplus S)[k+i] = \max(A_l[n^S_{\text{pixels}} - i], A_r[i]), \tag{4.25}$$

where k denotes the starting index of the block.

Now that the 1-D vHGW algorithm, structural element decomposition of a 2-D rectangular shape, and the use of symmetry to reduce computations have been presented, vHGW-360 is given. Algorithm 4.2 shows the outline of vHGW-360 aimed to calculate the configuration space costs. Compared to FAMOD, it is exact in the computed output values, but can only be used if the robot shape is rectangular.

4.4.3 Practical Considerations

The implementation of the 1-D vHGW dilation on the GPU is similar to the work of [159], as it showed good performance. It is important to consider how the map resides in memory and how it is accessed. Global memory accesses with CUDA are much faster if they are coalesced [120]. Since a 1-D dilation is required in both, the horizontal and the vertical direction, one of the two will be substantially slower. In [159], it is therefore proposed

to apply an optimized matrix transpose before and after the dilation and to perform the operation only in the faster direction. The same strategy is applied here, but rotations are used instead of transposes, since by considering their order, the total number of rotations is reduced again. This is already integrated in Algorithm 4.2.

In the next section, the problem of evaluating complete paths given the configuration space costs is discussed.

4.5 Evaluating Continuous Paths on Discrete Grids

Having computed the configuration space costs from a workspace cost map, checking individual configurations for collision and cost is done with simple look-ups. However, the question still remains of how to optimally check whole, continuous paths given $\mathcal{C}_{\text{costs}}$, in order to calculate the total path cost.

4.5.1 Calculating Path Costs with the Configuration Space Costs

As described in Section 4.2 and given in (4.1), assuring that a path or trajectory $\tau \in \mathcal{T}$ from a set of paths of trajectories $\mathcal{T}$ is collision-free, requires determining that all of its configurations $q \in \mathcal{C}$ are collision-free. Likewise, the cost of a path requires evaluating the costs of all of its configurations.

With the integral as past cost measure, the path cost is calculated with the configuration space costs $\mathcal{C}_{\text{costs}}$ as

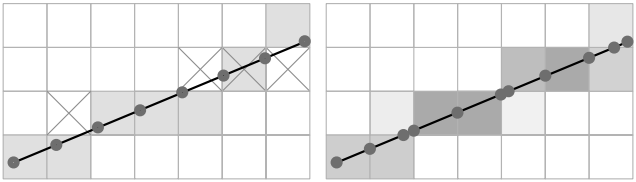
$$\mathcal{T}_{\text{costs}} : \begin{cases} \mathcal{T} \to \mathbb{R}_0^+ \\ \tau \mapsto \displaystyle\int_0^l \mathcal{C}_{\text{costs}}(\tau(s))\,\mathrm{d}s \end{cases} \tag{4.26}$$

Since in most cases, as is here, the configuration space costs are discrete, such as in the form of a grid map, discrete look-ups at positions s_k have to be performed to calculate the cost of a path

$$\mathcal{T}_{\text{costs}} : \begin{cases} \mathcal{T} \to \mathbb{R}_0^+ \\ \tau \mapsto \displaystyle\sum_k \mathcal{C}_{\text{costs}}\big(\delta(\tau(s_k))\big)\big(s_{k+1} - s_k\big) \end{cases} \tag{4.27}$$

where $\delta(q)$ discretizes the continuous configuration q to its corresponding cell in $\mathcal{C}_{\text{costs}}$.

The first and simplest choice of the individual look-up positions is probably equidistant sampling. There are, however, clear drawbacks. One the one hand, a high sampling rate is required in order not to miss any potential cell of the configuration space costs, and on the other, with a high sampling rate, cells are checked multiple times, as shown in Figure 4.5a.



(a) Regular sampling. **(b)** Weighted intersection sampling.

Figure 4.5: Regular sampling, (a), leads to missing cells or duplicates marked with X, while weighted intersection sampling, (b), yields the desired result.

The look-up positions are therefore calculated according to the following definition:

Definition 4.1 *An optimal evaluation of a continuous path τ on a discrete structure requires evaluating τ at a set P of look-up parameters s_k such that*

$$\forall s \in [0, l]\ \exists s_k \in P : \delta(\tau(s)) = \delta(\tau(s_k)) \wedge \forall s_k, s_j \in P : \delta(\tau(s_k)) \neq \delta(\tau(s_j)) \tag{4.28}$$

and weighting each configuration cost with the traversed length.

A path evaluation according to Definition 4.1 assures that all cells traversed by the path or trajectory are determined with the minimum number of total samples needed. It is shown next, how this can be achieved for linear motion primitives.

4.5.2 Determining Look-Up Positions

Determining the cell positions traversed by a motion primitive is strongly related to rasterization and to volume visualization in the field of computer graphics [69]. Computer graphics algorithms, such as Bresenham's line drawing algorithm, have been used before to determine the cell positions for collision checking on a 2-D grid for a point-shaped robot [29]. Incorporating the robot geometry requires extensions to higher dimensions. Additionally, Bresenham's algorithm is entirely integer-based and thus not applicable for planners that work in the continuous domain, such as the one presented in Chapter 3. Whereas in discrete domains, it sometimes is reasonable to precompute the cells and the corresponding segment lengths traversed by a primitive, in the continuous domain, it is in general not, underlining its importance.

Rendering in volume visualization is often done by determining the voxels traversed by a ray, and similarly it is also often desirable not miss any voxel and to use every voxel only once. Therefore, the process is adopted to determine the grid cells in the configuration space costs. To this end, the parametric equation of a line $x(t) = p + tv$ for a point p, a vector v, and the parameter t, is solved separately for t:

$$t_x(x) = \frac{1}{v_x}x - \frac{p_x}{v_x}, \qquad t_y(y) = \frac{1}{v_y}y - \frac{p_y}{v_y}, \qquad t_\theta(\theta) = \frac{1}{v_\theta}\theta - \frac{p_\theta}{v_\theta}. \tag{4.29}$$

Starting from the initial cell, the cell intersections of the next cells are evaluated with (4.29) for (x, y, θ), and $t_{\min} = \min(t_x, t_y, t_\theta)$ gives the next cell to check in a 3-D configuration space. The minimum value determines the dimension for which the first ray intersection occurs. Figure 4.5b shows the discussed intersection sampling.

According to Definition 4.1, the extracted configuration costs need to be weighted with the length of the traversed path segments. If the vector v is normalized, i.e., $||v|| = 1$, the weight directly corresponds to $t_{\min}$. However, the incompatibility between Euclidean and angular units makes the definition of a path length, and thus the normalization, difficult and ambiguous. For autonomous vehicles, it is chosen here to completely discard the orientation θ from the path length, since it is not possible to move in θ direction without moving in x or in y direction. Therefore, v is normalized such that $\sqrt{v_x^2 + v_y^2} = 1$. Algorithm 4.3 shows the method.

For non-holonomic robots, line segments are typically not sufficient and non-linear primitives, such as circular arcs, as used in the search graph described in Section 3.2.2, are present. Unfortunately, Algorithm 4.3 involves considerably more computational resources for circular arcs. On the one hand, solving for the parameter requires expensive arcsin and arccos computations inside the loop or the use of look-up tables. On the other hand, the position of the next cells cannot be determined as simple since ambiguities arise. Alternatively, if computational performance is of prior concern, it is still possible to sample the motion primitives at equidistant parameters, to connect the samples with line segments, and to apply Algorithm 4.3. Results are given in the following.

4.6 Results

Finally, this section shows results of the configuration space costs on different costs maps and shows the usefulness for planning algorithms. Moreover, the performance of the proposed algorithms of their calculation is evaluated. While FAMOD is able to calculate the configuration space costs efficiently for arbitrary robot shapes, its output range is limited. The vHGW-360 algorithm, on the other hand, does not have output range limitation, but requires rectangular masks.

Configuration Space Costs on Cost Maps from Inverse Obstacle Distance

It is started with qualitative results. Figure 4.6 shows a standard occupancy grid in a parking lot from a laser scanner. Although this grid may directly be used as cost map, it is post-processed in this application. First, a binary obstacle grid is created by thresholding. Then, a distance transform is applied on the obstacle map, i.e., for every cell the distance to the closest obstacle cell is calculated. And finally, the distance transform is inverted so that high costs represent low distances, and zero distances correspond to the highest cost value. Finding a minimum cost path in the configuration space using the given cost map will therefore automatically result in path that has high clearance, and collisions can directly be detected by the highest configuration cost value. One slice of the configuration space costs is shown, which corresponds to the current orientation of the ego vehicle.

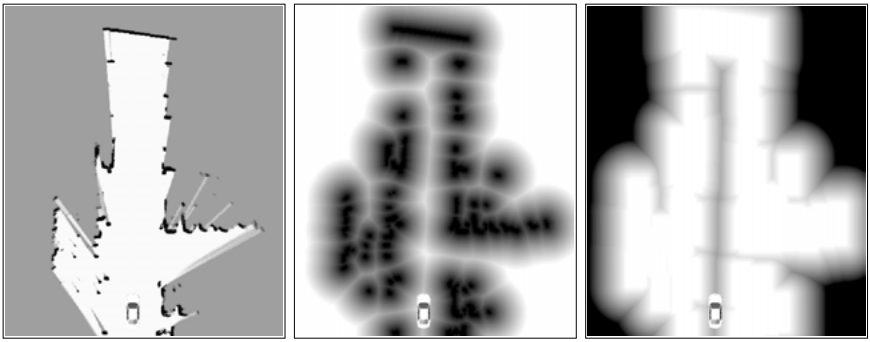
Algorithm 4.3 Check Cells Between q_0 and q_1

Require: $\Delta x, \Delta y, \Delta\theta \neq 0$
$(x, y, \theta) \leftarrow \text{round}(x_0, y_0, \theta_0)$
$(x_{\text{end}}, y_{\text{end}}, \theta_{\text{end}}) \leftarrow \text{round}(x_1, y_1, \theta_1)$
$(v_x, v_y, v_\theta) \leftarrow (\Delta x, \Delta y, \Delta\theta)/\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$
$(d_x, d_y, d_\theta) \leftarrow (1/2)(\text{sgn}\Delta x, \text{sgn}\Delta y, \text{sgn}\Delta\theta)$
$(t_x, t_y, t_\theta, t_{\min}) \leftarrow (-1, -1, -1, 0)$
while $(x, y, \theta) \neq (x_{\text{end}}, y_{\text{end}}, \theta_{\text{end}})$ **do**
  if $t_{\min} = t_x$ **then**
    $x \leftarrow x + 2d_x$
  end if
  if $t_{\min} = t_y$ **then**
    $y \leftarrow y + 2d_y$
  end if
  if $t_{\min} = t_\theta$ **then**
    $\theta \leftarrow \theta + 2d_\theta$
  end if
  $t_x \leftarrow (x + d_x)/v_x - x_0/v_x$
  $t_y \leftarrow (y + d_y)/v_y - y_0/v_y$
  $t_\theta \leftarrow (\theta + d_\theta)/v_\theta - \theta_0/v_\theta$
  $t_{\text{prev}} \leftarrow t_{\min}$
  $t_{\min} \leftarrow \min(t_x, t_y, t_\theta)$
  Check cell (x, y, θ) and weight by $t_{\min} - t_{\text{prev}}$
end while

Configuration Space Costs on Cost Maps from Offline Map Data Combined with Online Map Data

Cost maps can also be derived from offline map data and combined with online sensor data. In Figure 4.7, a cost map is derived from a navigation map of a parking lot and combined with the information of an online occupancy grid. Visible are parking spots, stored walls of the offline map, as well as obstacles from the online map. In addition, the current route from a high-level navigation planner is used to assign low costs to the current route, while assigning high costs to other route elements. Such workspace cost maps are useful for planning as they also incorporate logical information from traffic rules, and due to the given definition of the configuration space costs as well as the given algorithms for their calculation, they can quickly be incorporated. They enable efficient planning even in complex environments and under a high planning complexity, such as in parking scenarios. Also shown in Figure 4.7 are two slices of $\mathcal{C}_{\text{costs}}$.

The configuration space costs given such cost maps is also used in the BMW Research Fully Automated Remote Valet Parking Assistant shown at the Consumer Electronics Show (CES) 2015 [2, 23]. In this research prototype system, the vehicle is able to navigate without a driver through a parking garage and to park itself into an empty parking spot. If the driver wants the car back, it can simply be called over an app on a smartwatch, and the car picks up the driver fully automated. To this end, the system uses a hierarchy of



(a) Occupancy grid. **(b)** Distance transform. **(c)** Slice of $\mathcal{C}_{\text{costs}}$.

Figure 4.6: Slice of $\mathcal{C}_{\text{costs}}$ (c), corresponding to the orientation of the ego vehicle, of a cost map that comes from thresholding an occupancy grid (a), and calculating and inverting the distance transform (b).

different planners, with one level being a path planner that solves the complex combinatorial problem for navigation in narrow possibly cluttered environments [73]. It relies on the configuration space costs to be able to plan a cost-optimal path given a cost map, as shown in Figure 4.7a, in the configuration space.

Performance Evaluation of the Proposed Algorithms

Now, the computational performance of FAMOD and vHGW-360 is analyzed and compared to standard grayscale dilation algorithms. The hardware and software platform is described in Appendix A.2. Both, FAMOD and vHGW-360 were implemented on the GPU using Nvidia CUDA. All timing measurements of the GPU include copying of the output from graphics memory back to host memory, in order to incorporate the fact that most motion planners are CPU-based. The time for kernel generation is excluded.

Figure 4.9 gives the computing time for a single grayscale dilation of a 512×512 grid with a non-axis-aligned quadratic mask of different sizes of the following algorithms: FAMOD, vHGW-360, Nvidia's GPU dilation function from the NPP library *nppiDilate_8u_C1R* [123], Matlab's *imdilate* (CPU), and a naive CPU implementation in C++. FAMOD has the lowest computing times and is independent of the size of the mask, as expected. The vHGW-360 algorithm, on the other hand, contrary to a theoretical constant complexity per pixel, increases slightly, although for most cases negligibly, with the size of structuring element. This phenomenon is also exhibited in the implementation of LTU-CUDA [158]. Nvidia's nppiDilate_8u_C1R seems to implement a direct approach on the graphics card in the version tested, and thus shows a strong increase with the width of the mask. It is, however, substantially faster than the tested CPU algorithms, i.e., Matlab's imdilate, and an implemented serial naive C++ CPU implementation.

Next, in Figure 4.10 the total computing times for the calculation of $\mathcal{C}_{\text{costs}}$ with different number of slices, i.e., different angular resolutions, are given. The mask is of size 25×11

(a) Cost map. **(b)** Slice of $\mathcal{C}_{\text{costs}}$ at 90°. **(c)** Slice of $\mathcal{C}_{\text{costs}}$ at 150°.

Figure 4.7: Cost map from a combination of stored logical map data and obstacle data (a) and two slices of the configuration space costs (b) and (c) [73].

Figure 4.8: BMW Fully Automated Remote Valet Parking Assistant [2, 23]. Images courtesy of BMW Group.

representing a vehicle of size $5\,\text{m} \times 2\,\text{m}$ at a cell resolution of $0.2\,\text{m}$. Since vHGW-360 requires uneven mask sizes, the width is increased by one cell. Note that the x-axis denotes the number of actual dilations. For a rectangular mask, the number of orientations, i.e., slices of $\mathcal{C}_{\text{costs}}$, is therefore twice as much. The computational advantage of FAMOD slightly increases further over vHGW-360, as the function $g(M)$ and its Fourier transform have to be calculated only once for all dilations.

Furthermore, the total computing time for a varying number of cost evaluations is analyzed. In Figure 4.11 a direct C++ CPU implementation of cost evaluations, that does not use any preprocessing except for the rotated masks, is compared to the time it takes to first compute $\mathcal{C}_{\text{costs}}$ using FAMOD, copy the result from graphics to host memory, and to then perform the evaluations. Table 4.1 shows the exact timings t for the precomputations and the number of checks n_{checks} it requires to outperform a direct CPU configuration evaluation without preprocessing. In the applications of this thesis, a 5° angular resolution is used, which corresponds to 36 dilations with a rectangular vehicle approximation. With this number of dilations it only requires around 6000 configuration checks to outperform a direct evaluation. Assuming a path length of 500 cells, it only requires checking as little as 12 paths to overcome the time for the precomputation. In many motion planning ap-

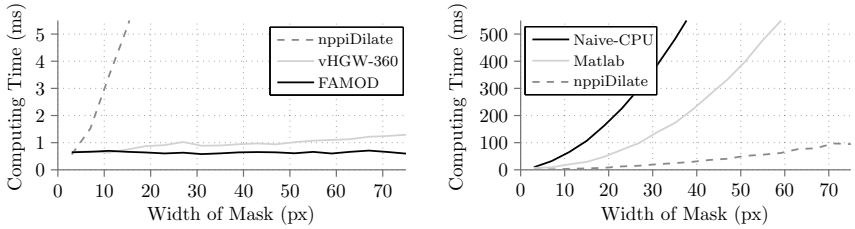


Figure 4.9: Computing time for a single grayscale dilation of a 512×512 grid with a $30°$ rotated quadratic mask of varying size. Shown are from fastest to slowest: FAMOD (GPU), vHGW-360 (GPU), Nvidia's nppiDilate (GPU), Matlab's imdilate (CPU), a naive serial CPU implementation. Nvidia's nppiDilate is shown in both plots for comparison.

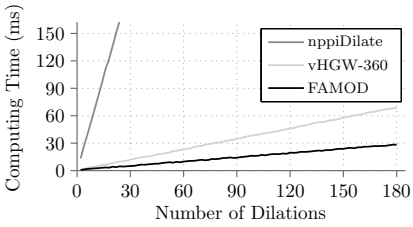


Figure 4.10: Computing time for different numbers of consecutive grayscale dilations of a 512×512 grid with a rectangular 25×11 structuring element.

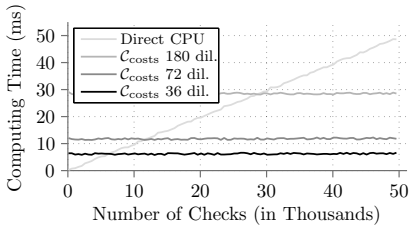


Figure 4.11: Comparison between a direct check on the CPU and precomputation of $\mathcal{C}_{\text{costs}}$ with FAMOD on a 512×512 grid with a randomly chosen 25×25 mask.

plications, as in this work, the number of paths easily grows in the thousands, underlining the benefit of the approach.

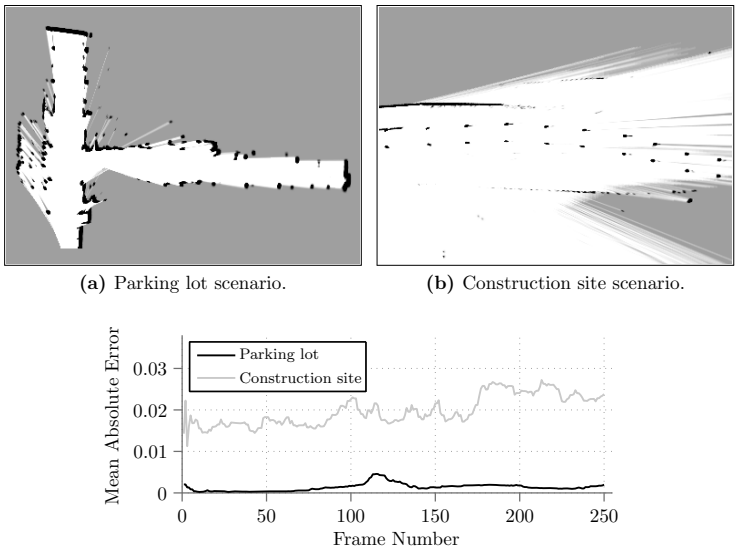
Additionally, Table 4.1 gives the computation time for $\mathcal{C}_{\text{obs}}$ using binary convolution on the GPU. It can be observed that the performance difference to FAMOD is under 5%. The number of checks to break even is not given, since a direct CPU implementation skips evaluating the rest of the cells as soon as a collision occurs making the performance dependent on the grid.

Output Range Accuracy of FAMOD

Finally, the errors due to the output range approximation of FAMOD are analyzed. The algorithm was evaluated on cost maps from occupancy grids recorded in two different scenarios: a parking lot representing a typical corridor scenario and a road construction site with non-continuous road boundaries. The number of output cost values of FAMOD was set to the lowest number possible still differentiating it from binary dilation, i.e., 3. The classes of cost values are distributed uniformly in the range of the original cost map. They

Table 4.1: Precomputation times for the GPU algorithms and break-even points for a direct configuration check on the CPU.

	36 dilations		72 dilations		180 dilations	
	t (ms)	n_{checks}	t (ms)	n_{checks}	t (ms)	n_{checks}
FAMOD	6.01	6.1K	11.56	11.8K	28.22	28.7K
vHGW-360	14.19	14.4K	27.87	28.3K	69.03	70K
nppiDilate	245.6	249K	490.7	494K	1226.3	1.25M
Bin. dilation by conv.	5.75	–	11.06	–	26.95	–



(a) Parking lot scenario.

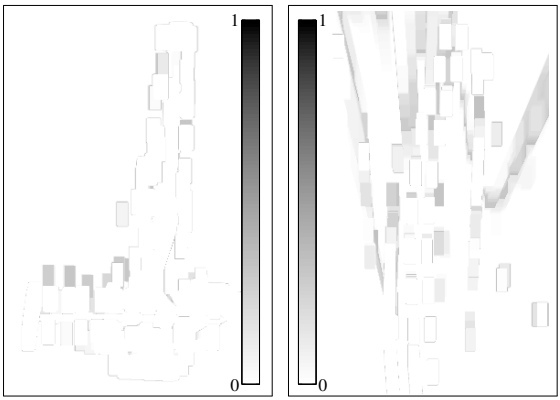
(b) Construction site scenario.

(c) Error.

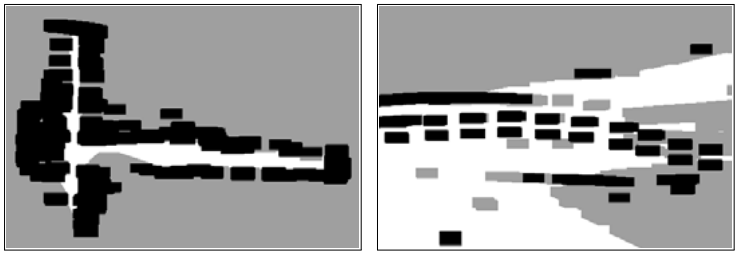
Figure 4.12: Mean absolute difference between exact grayscale dilation and FAMOD using 3 classes (c) in two different scenarios.

map to the minimum occupancy value, the maximum occupancy value, and to unknown, i.e., $p(o) = 0.5$.

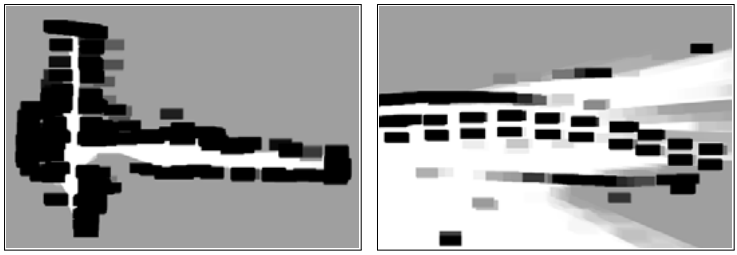
Figure 4.12 gives the mean absolute error over all cells normalized to the range $[0, 1]$. The clear edges of the parking lot scenario work in favor of the accuracy, but even in the construction site scenario, the error stays low. Figure 4.13 gives a qualitative comparison on one sample map of each scenario and shows the absolute difference.



(a) Absolute difference



(b) FAMOD with 3 classes.



(c) Exact dilation.

Figure 4.13: Comparison between FAMOD and an exact dilation of the parking lot scenario (left column) and the construction site scenario (right column) with a rectangular vehicle mask. Figure (a) is rotated counterclockwise by $90°$.

4.7 Summary

Collision checks and cost evaluations are the computationally most expensive part in many motion planning algorithms. Checking a single robot configuration in a grid-based world representation requires examining every cell under the robot footprint in that particular configuration. Often it is natural to design costs in the workspace of the robot, such as distances to obstacles or areas that are safe to traverse. Similar to collision checking, the robot shape also needs to be considered during cost evaluation.

This chapter has presented fundamentals on how to evaluate robot configurations on workspace cost maps by incorporating the robot shape. Configuration space obstacles, which allow collision checking of a whole configuration to be done with a single look-up, have been generalized to configuration space costs, which allow the check for both, cost and collision, of a whole configuration to be performed likewise.

Two methods have been proposed to efficiently calculate the configuration space costs: FAMOD, which uses convolution to calculate grayscale dilation, and vHGW-360, which is based on the van Herk-Gil-Werman dilation algorithm. Both algorithms show fast computation times, which potentially speed-up motion planners by orders of magnitude. While FAMOD is independent of the shape of the robot, the output range is limited and will therefore yield approximate results. The vHGW-360, on the other hand, is specifically designed for rectangular masks.

Furthermore, this chapter has discussed the problem of how a continuous motion primitive or a whole path may correctly and efficiently be checked using the configuration space costs. It is important neither to miss a cell nor to check cells multiple times. Similar to the previous chapters, the huge performance gain of parallel computations on graphics hardware has again been demonstrated, as FAMOD and vHGW-360 were implemented in CUDA. Having presented the major components, which may be individually and independently used, the next chapter gives the road course estimation. It uses all of the components given so far.

5 Road Course and Road Boundary Estimation

The road course with its road boundaries is an essential component of many advanced driver assistance systems and of autonomous vehicles. It represents the road shape ahead of, and relative to, the ego vehicle and thus defines where the vehicle is required to move. Although it is commonly stored in a map during an offline process [1], an online sensor-based estimation is necessary for robust applications. Road changes, such as due to road construction sites, require a high map maintenance, and there is no guarantee that the stored information is valid. Additionally, some road boundaries simply cannot be stored in an offline map, since they only constitute to the road boundary for a certain limited time frame, such as parked vehicles in urban areas. The presented approach enables a sensor- and feature-independent multi road course estimation that works with arbitrary road boundaries. The estimated principal moving directions, presented in Chapter 3, yield a first estimate of the number of individual road courses and provide boundary separators. This chapter is based on work published in [5, 10] in the context of this thesis.

5.1 Introduction

Road surface, road-boundary, and lane detection are established areas of research [33, 87], and there exists a considerable amount of proposed methods in the literature designed for different applications and for different sensors. An overview of a variety of existing approaches is found in [66]. In this section, it is focused on similar methods to the one that is proposed.

5.1.1 State of the Art

The majority of road boundary estimation methods uses features, with camera-based lane marking recognition systems being the most prominent [84, 100, 177]. Lane marking recognition is one of the primary components of any autonomous vehicle driving on public streets and is the basis of many driver assistance systems, such as lane departure warning systems. It is therefore indeed mandatory as component. However, lane marking recognition systems reach their limits in more complex, ambiguous, and unstructured environments, such as urban areas or road construction sites. Sometimes lane markings are invalid, conflicting with temporary markings, or simply not provided, such as in many single-lane roads.

Other camera-feature-based road boundary detection methods search for the boundaries of the road [68, 75, 92, 107]. They usually work well on rather uniform roads with contin-

uous boundaries. Real street environments, however, often exhibit small holes and repair spots and non-continuous road boundaries, such as traffic cones or parked vehicles.

Apart from cameras, other sensors have been used to estimate the road boundaries, such as laser scanners [66, 89, 126, 172], and specific methods have been developed. Due to the scanning principle, often discontinuities between neighboring scan points are used to detect curbs and obstacles. As with camera features, these methods are designed for one sensor, and in the absence of the exploited features, they will fail.

Methods that do not rely on certain features or are not designed for one specific sensor are applicable in a more general and wider spectrum. A sensor-independent world representation, as mentioned on multiple locations throughout this thesis and discussed in detail in Chapter 2, are occupancy grids. In occupancy grids, the sensor information is already filtered over time and therefore noise measurements are already suppressed. Additionally, the information about previously observed regions is maintained in a local frame, as well as previously observed but currently occluded obstacle information. Furthermore, occupancy grids *should* only hold obstacle information about the static environment and thus be free from moving objects, which typically do not constitute to the road boundaries. Although standard occupancy grids are not designed for dynamic environments [157], there are methods, such as the one proposed in Chapter 2, that deal with this issue. Hence, they provide a sensor-independent representation of the environment that already copes with many fundamental but important data pre-processing steps. It is noted that occupancy grids have also been used with features [90].

The majority of grid-based road boundary estimation methods start by separating the environment into two regions: one that holds the occupied grid cells of the left boundary, and one that holds the ones of the right boundary [19, 37, 91, 149, 168]. Each boundary is then estimated individually. The separation is done by casting a ray in the current vehicle orientation [19, 91, 168] or by using a motion model to calculate a predicted vehicle trajectory [37]. In [168], the separator ray is divided into individual segments that are moved individually to the left and to the right to detect the boundary points. The boundaries are then estimated with parabolic curves. In [37], two search windows as well as assumptions about the shape and the structure of the road boundary are used to extract the boundary cells. A clothoidal model is tracked with an extended Kalman filter for each of the two boundaries. In [91], the road boundaries are estimated by optimizing with the Levenberg-Marquardt algorithm. The part to the rear of the ego vehicle is optimized with a clothoid model, whereas the part to the front is optimized with a quadratic model.

A separation with the predicted vehicle trajectory does not always work, however, although it is crucial for all of the above techniques. If the separation fails, the estimation automatically fails as well. With non-continuous road boundary elements, such as given by traffic cones, the predicted vehicle trajectory easily passes in between two obstacles, as shown in Figure 5.1. In such a case, the true road course cannot be detected. In addition, since only the predicted vehicle trajectory is used, roads with multiple road courses, such as forks in the road, branches, or road junctions, cannot be represented.

There are also approaches that do not rely on a separation of the environment. In [71], a quadratic model is directly fitted into the occupancy grid by optimizing over histograms of different curvatures and headings with the Nelder-Mead-Simplex algorithm. However, the

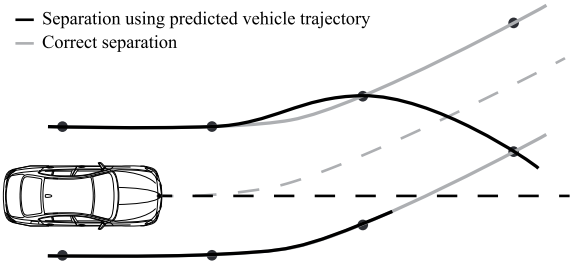


Figure 5.1: Estimation errors, if the predicted vehicle trajectory is used as boundary separator.

boundaries must be symmetric and the number of occupied grid cells that represent the road boundary must be high, compared to the occupied cells from non-boundary objects. In [106, 108], a particle filter framework is used to detect and track the parameters of a road network model in a colored elevation map. The road model exhibits an intersection center and a clothoid model for each branch. Combining camera and vision data in a colored elevation map enables an estimation in many road scenarios, such as rural roads. The topology of the road, i.e., the number of branches, needs to be known in advance, however. Quadratic and clothoid models restrict the estimation horizon in complex driving environments, such as urban roads or road construction sites, and such optimization techniques usually require prior knowledge, such as the road topology, in order to cope with the complexity.

5.1.2 Approach and Contribution

This chapter presents a novel grid-based road course estimation approach. Rather than relying on the predicted vehicle trajectory, it uses motion planning to detect the principal moving directions through the environment. It therefore overcomes the shortage of most existing grid-based approaches of not finding the correct road course in scenarios such as Figure 5.1. Moreover, it overcomes the limitation of finding only a single road course, as well as estimation errors over long planning horizons due to the predicted trajectory, such as with S-shaped curves. The number of road course hypotheses in scenarios with non-continuous road boundaries is greatly reduced by exploiting non-holonomic constraints, using collision-checking, and by assuming a certain drivable velocity, such as from speed signs. In summary, the proposed approach

- estimates the road boundaries indirectly by using motion planning to find the principal moving directions through the environment,
- is able to detect and represent roads with multiple road courses, such as forks in the road and road junctions,
- can handle arbitrary road shapes, such as one sided road narrowings and S-shaped curves,

- works with arbitrary road boundaries, continuous ones, such as guardrails, and non-continuous ones, such as parked vehicles and traffic cones, and
- is independent of specific features or particular sensors.

The rest of this chapter is structured as follows. Section 5.2 gives an overview of the approach and shows how the individual components, many of which have already been presented in the previous chapters, work together. Section 5.3 then describes how the boundaries are extracted and estimated based on the principal moving directions. Section 5.4 and Section 5.5 discuss how the road courses are validated and tracked, and finally Section 5.6 shows results.

5.2 Overview

An overview of the road course estimation system is given in this section, and it is referred to the individual sections and chapters for further details.

The outline is presented in Algorithm 5.1. The aim is to estimate a set of road courses $\mathcal{R}$, where each road course $\rho(s) = \begin{pmatrix}\beta_l(s) & \beta_r(s)\end{pmatrix}^{\mathrm{T}}$ is defined by exactly two *semantic* continuous boundaries, $\beta_l(s)$ and $\beta_r(s)$. The length parameter s is used in this chapter instead of l as previously used, to reduce ambiguities with the index for left. The road boundaries may either be given explicitly, such as with guardrails or tunnel walls, but may also be given implicitly with non-continuous boundary elements, such as traffic cones. With continuous boundaries, the semantic boundary equals the obstacle elements, and the vehicle cannot leave the road course without a collision. With non-continuous boundaries, however, the semantic boundary gives the desired road course, even if it is possible to move out of it without a collision. The latter scenarios are typically the harder ones for road boundary estimation methods.

The approach works on an occupancy grid that holds the static environment. In Chapter 2 a novel approach has been presented to create static grids in dynamic environments. Hence, with each new measurement, the grid is updated and provided to the subsequent modules. Similar to other grid-based approaches, the following observation is exploited:

Observation 5.1 *Given an arbitrarily shaped curve $\tau(s)$ that lies within the road course ρ. Then, the complexity of estimating the corresponding road boundaries, $\beta_l(s)$ and $\beta_r(s)$, from the obstacles $\mathcal{O}$ is greatly facilitated, compared to a direct boundary estimation in $\mathcal{O}$.*

As mentioned above, this approach uses motion planning to find these boundary separators, which divide the environment into left and right, rather than simply relying on the predicted vehicle trajectory. More specifically, it relies on the principal moving directions through the environment. In Chapter 3 a novel method has been presented to efficiently sample local paths and to cluster them in order to extract these main maneuvers through the environment. The complexity of the planning problem and the number of principal moving directions with non-continuous road boundaries is greatly reduced by exploiting velocity constraints, in addition to constraints from the obstacles and the motion model. It is assumed that a rough estimate of the velocity v_{road}, which can be safely driven on the

Algorithm 5.1 Road Course Estimation using Motion Planning

Input: Estimated drivable velocity v_{road}, vehicle pose x_t, measurement z_t
Output: Set of road courses $\mathcal{R}_t$

$t \leftarrow 0$
$m_0, \mathcal{R}_0 \leftarrow \emptyset$
while running **do**
  $m_t \leftarrow \text{updateStaticMap}(m_{t-1}, z_t, x_t)$
  $m_c \leftarrow \text{generateCostMap}(m_t)$
  $\mathcal{C}_{\text{costs}} \leftarrow \text{calculateConfSpaceCosts}(m_c)$ // Algorithm 4.1 or 4.2
  $\mathcal{T}_{\text{rep}} \leftarrow \text{findPrincipalMovingDirections}(x_t, v_{\text{road}}, \mathcal{C}_{\text{costs}})$ // Algorithm 3.1 and 3.2
  $\mathcal{R}_{\text{curr}} \leftarrow \emptyset$
  for all separators $\tau \in \mathcal{T}_{\text{rep}}$ **do**
    $\mathcal{B}_{\{l,r\}} \leftarrow \text{extractBoundaryCells}(\tau, m_t)$
    $\beta_{\{l,r\}} \leftarrow \text{estimateSemanticBoundary}(\mathcal{B}_{\{l,r\}})$
    $\rho \leftarrow \text{validateRoadCourse}(\beta_l, \beta_r)$ // estimate $p(\rho \mid \xi)$
    $\mathcal{R}_{\text{curr}} \leftarrow \mathcal{R}_{\text{curr}} \cup \rho$
  end for
  $\mathcal{R}_t \leftarrow \text{updateTracker}(\mathcal{R}_{\text{curr}}, \mathcal{R}_{t-1})$
  $t \leftarrow t + 1$
end while

road, exists. Such an estimate may be provided by a traffic sign recognition system, since on many roads, the maximum allowed velocity still allows safely following it. There are also traffic signs indicating dangerous curves, i.e., curves with high curvatures, where the velocity needs to be reduced. It can also come from rough navigation maps. Curvatures on highways are different than curvatures on rural roads or on urban streets. It may also be estimated from other traffic participants driving in the vicinity of the ego vehicle.

Collision checking and cost evaluation is the most computation-intensive part of the planner. Therefore, the configuration space costs, as introduced in Chapter 4 are precalculated. A cost map is derived from the static occupancy grid and used as input for the calculation of $\mathcal{C}_{\text{costs}}$. Depending on the scenario, the static occupancy grid is either used directly as cost map, or a distance transform is applied on the obstacles and inverted.

Every found principal moving direction $\tau \in \mathcal{T}_{\text{rep}}$ is regarded as one road course hypothesis ρ. Hence, for every τ the set of corresponding occupied grid cells, $\mathcal{B}_l$ and $\mathcal{B}_l$, which represent its left and right boundary, are extracted. Then, the continuous semantic boundaries, $\beta_l(s)$ and $\beta_r(s)$, are estimated, which constitute the road course ρ. Finally, each road course is validated based on the estimated boundaries, since even with the kinematic and velocity constraints it may happen that a principal moving direction does not correspond to a valid road course. The road course, as well as its probability of being valid given certain criteria ξ, is given to the tracker to yield a robust and consistent representation. Next, the road boundary estimation is presented.

5.3 Path-based Road Boundary Estimation

After motion planning and path clustering, the road boundaries are estimated with the path representatives of the clusters, and road course hypotheses are generated, as described in this section.

5.3.1 The Effects of Path Clustering

Every path from the motion planner is one road course separator, i.e., it divides the environment into two parts, where one holds, amongst other obstacles in the map, the left boundary, and one the right boundary. From the computational point of view, it is not reasonable to use every output path of the path planner as base for a road course hypothesis, i.e., to apply the subsequent steps as given in Algorithm 5.1. It also does not yield a greater variety in the final estimations, since many paths are similar. Contrary even, the data association in the tracker has then to resolve a large number of ambiguities, which most likely leads to a decrease in the performance.

In Chapter 3, a path clustering approach has been proposed to reduce the set of paths to the principal moving directions. Here, it is now discussed if this clustering assures the desired properties for the road course estimation presented in this chapter. Two observations are made about the effects of the clustering on the system:

Observation 5.2 *If paths, which lead to different estimated road courses, are in the same cluster, only one road course hypothesis is generated. The other road courses are lost.*

And similarly:

Observation 5.3 *If paths, which lead to the same estimated road course, are in different clusters, multiple road course hypotheses are generated that are equivalent. Complexity is unnecessarily increased and ambiguities for the tracker are introduced.*

Hence, it directly follows that:

Corollary 5.1 *A path clustering is optimal for a motion-planning-based road course estimation, if every cluster holds all paths that lead to the same estimated road course, whereas different clusters hold paths that lead to different road courses.*

A direct implementation of Corollary 5.1, however, requires estimating the boundaries with every original path and comparing for similarity. This conflicts the motivation from above about complexity and ambiguity reduction. Since the boundaries are extracted based on the paths, it can be observed that if the paths separate the obstacles in the environment equally, the resulting road course hypotheses are also equal. Furthermore, if two paths separate the environment differently, there must be an obstacle between them. In this case, according to the clustering predicate from Definition 3.1, they are then in different clusters. Similarly, if the paths separate the environment equally, then there is no

obstacle in between them, and they are combined to the same cluster as of Definition 3.1. Therefore, the proposed clustering from Chapter 3 yields the desired properties.

5.3.2 Road Boundary Estimation

This section now presents how the boundaries are estimated based on the cluster representatives.

Boundary Element Extraction

Given a path τ with path nodes $\tau^n \in \mathbb{R}^2$ as described in Section 3.2.2, the occupied cells $o \in \mathcal{O} \subseteq \mathbb{R}^2$ to the left and to the right of τ

$$\begin{aligned} \mathcal{O}_l &= \left\{ o \in \mathcal{O} \mid \det\left(\tau_i^n - \tau_{i-1}^n, o - \tau_i^n\right) > 0 \right\} \\ \mathcal{O}_r &= \left\{ o \in \mathcal{O} \mid \det\left(\tau_i^n - \tau_{i-1}^n, o - \tau_i^n\right) < 0 \right\} \end{aligned} \tag{5.1}$$

are determined with the closest two path nodes τ_i^n and τ_{i-1}^n from every $o \in \mathcal{O}$. Note the definition of the grid coordinate system given in Appendix A.3.

In the occupancy grid, there are typically more occupied cells than just the road boundaries. They come from other static obstacles in the environment. Since τ separates the road course, it is assumed that the road boundary elements are close to it. Hence, the left and right obstacle elements, $\mathcal{O}_l$ and $\mathcal{O}_r$, are refined to extract the road boundary elements

$$\mathcal{B}_{\{l,r\}} = \left\{ o \in \mathcal{O}_{\{l,r\}} \mid \forall o_j \arg\min(||o_j - \tau_o^c||) = o \right\}, \tag{5.2}$$

where τ_o^c denote the closest path cell from o. The distance from (5.2) must also be smaller than some maximum distance $d^{\max}$.

Furthermore, obstacles that are *behind* the end of the path, are excluded, since τ does not separate them, and the left/right classification is therefore ambiguous. In order to detect them, a circle with radius $d^{\max}$ is centered at

$$\begin{pmatrix} x_{\text{end}} & y_{\text{end}} \end{pmatrix}^{\mathrm{T}} + d^{\max} \begin{pmatrix} \cos\theta_{\text{end}} & \sin\theta_{\text{end}} \end{pmatrix}^{\mathrm{T}}, \tag{5.3}$$

where $\begin{pmatrix} x_{\text{end}} & y_{\text{end}} & \theta_{\text{end}} \end{pmatrix}^{\mathrm{T}} = \tau(l_G)$ is the end configuration of the path. Figure 5.2 shows the boundary cell extraction.

Semantic Boundary Estimation

Often, a continuous boundary representation is required. Especially in the case of road boundaries that consist of sparse road boundary elements, the semantic continuous boundary is useful. The extracted set of road boundary elements $\mathcal{B}_{\{l,r\}}$ allows a variety of methods to be applied, since they can be given in an ordered sequence based on the separator path. Even a simple polygonal chain is possible. For increased robustness, penalized regression splines [135] are used here. They allow controlling the smoothness over the flexibility of the curve and can thus be adapted to different scenarios and application requirements. The

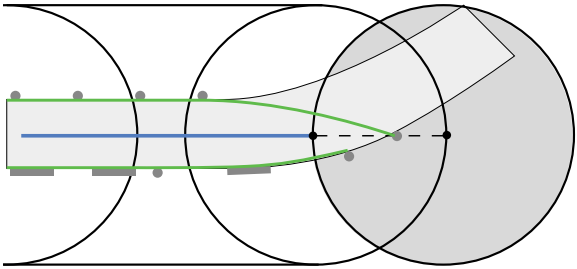


Figure 5.2: Extraction of the boundaries. The closest cells are searched for within some maximum distance (empty circles). Cells behind the path (blue) are excluded from the search (gray circle), since left/right classification is ambiguous, leading to incorrect boundary estimations (green).

estimated semantic boundaries together with the extracted boundary elements are then used to validate the road course hypotheses, which is described next.

5.4 Road Course Validation

Even though several constraints are exploited by the motion planner, it is not guaranteed that all principal moving directions correspond to valid road courses. Especially in scenarios with sparse boundaries, it is possible that a collision-free and feasible path is found that leaves the true road course in between two boundary elements. Therefore, all road course hypotheses are validated based on their shape in every time step, as described in Section 5.4.1. In addition, their plausibility is filtered over time, as described in Section 5.4.2.

5.4.1 Single Frame Validation

Every road course hypothesis ρ_i is validated in the current frame, i.e., in the current time step, by calculating its probability

$$\begin{aligned} p(\rho_i \mid \xi_1, \xi_2, \xi_3, \xi_4) &= \eta \; p(\rho_i) \; p(\xi_1, \xi_2, \xi_3, \xi_4 \mid \rho_i) \\ &= \eta \; p(\rho_i) \; \prod_j p(\xi_j \mid \rho_i), \end{aligned} \tag{5.4}$$

given certain properties ξ_j observed from valid road courses. They are assumed to be conditionally independent. Four properties were identified and used in the experiments. They are given in the following. It is noted that these properties were derived experimentally from data recorded with a laser scanner in real street environments. No claims about completeness are made. All properties only rely on the shape of the boundaries and do not use color, features, or object classification. The individual properties are evaluated in Chapter 6.

1) Constant Boundary Width

The width of a valid road course typically is rather constant or changes slowly. Strong deviations in the width are therefore a hint for an invalid estimation. Let $d_{\beta,s} = ||\beta_l(s) - \beta_r(s)||$ be the distance between corresponding values at the left and the right continuous semantic boundary, $\beta_l(s)$ and $\beta_r(s)$. Then,

$$p(\xi_1 \mid \rho_i) = f_w\left(1 - \frac{\max_s d_{\beta,s} - \min_s d_{\beta,s}}{d_\beta^{\max}}\right), \tag{5.5}$$

where $d_\beta^{\max}$ is a predefined maximum distance and $f_w : [0,1] \to [0,1]$ is a weighting function. In the implementation a sigmoid function is used.

2) Uniform Road Boundary Element Distribution

The boundary elements are often distributed rather uniformly, such as traffic cones in road construction sites. Furthermore, large gaps in between boundary elements, in comparison to the average distance between the boundary elements in the scene, are also typically not exhibited in real street environments. Let $\mu_b^{\{l,r\}}$ and $\sigma_b^{\{l,r\}}$ be the mean and the standard deviation of the distances $d_{b,i}^{\{l,r\}} = ||b_{i+1}^{\{l,r\}} - b_i^{\{l,r\}}||$ between consecutive boundary elements of the left or right boundary. Then,

$$p(\xi_2 \mid \rho_i) = \min \begin{cases} f_w\left(1 - \dfrac{\max_i d_{b,i}^{\{l,r\}} - \mu_b^{\{l,r\}} - \sigma_b^{\{l,r\}}}{d_b^{\max} - \mu_b^{\{l,r\}} - \sigma_b^{\{l,r\}}}\right) \\ f_w\left(1 - \dfrac{\sigma_b^{\{l,r\}}}{\sigma_b^{\max}}\right) \end{cases} \tag{5.6}$$

The symbols $\sigma_b^{\max}$ and $d_b^{\max}$ denote predefined maximum values. Equation (5.6) is evaluated for the left and the right boundary individually and the minimum value is chosen.

3) Equal Endpoint Tangents

In scenarios where the road boundaries are given by a sparse set of boundary elements, road course hypotheses that depart from the true road course often exhibit boundary estimations, which point apart of each other at the end. In such a case, the separator path leads to the extraction of the backfaces of the true road boundary elements. Let

$$v_l = \frac{\beta_l'(s_{\text{end}})}{||\beta_l'(s_{\text{end}})||}, \quad v_r = \frac{\beta_r'(s_{\text{end}})}{||\beta_r'(s_{\text{end}})||}, \quad v_\tau = \begin{pmatrix} \cos\theta_{\text{end}} \\ \sin\theta_{\text{end}} \end{pmatrix} \tag{5.7}$$

denote the normalized tangent vector at the endpoint of the left and the right boundary, as well as the tangent vector in the endpoint of the separator path $\tau(l_G) = \begin{pmatrix} x_{\text{end}} & y_{\text{end}} & \theta_{\text{end}} \end{pmatrix}^{\mathrm{T}}$. Then,

$$p(\xi_3 \mid \rho_i) = \begin{cases} f_w\left(1 - \frac{\sphericalangle v_l, v_r}{180}\right) & \text{if } (\sphericalangle v_l, v_\tau + \sphericalangle v_r, v_\tau) < 180 \\ 0 & \text{else} \end{cases} \tag{5.8}$$

where $\sphericalangle v_1, v_2 = \cos^{-1}(v_1 \cdot v_2)$.

4) Boundary Length Equals Separator Length

Finally, the lengths of the estimated semantic boundaries are analyzed. Since they are calculated based on a separator path τ, they are expected to exhibit the same length as τ, i.e., the goal path length l_G. Let

$$l_{\beta_l} = \int_{s_0}^{s_{\text{end}}} ||\beta_l'(s)|| \mathrm{d}s \quad \text{and} \quad l_{\beta_r} = \int_{s_0}^{s_{\text{end}}} ||\beta_r'(s)|| \mathrm{d}s \tag{5.9}$$

denote the lengths of the boundaries. Then,

$$p(\xi_4 \mid \rho_i) = \begin{cases} f_w\left(\frac{\min(l_{\beta_l}, l_{\beta_r})}{l_G}\right) & \text{if } \min(l_{\beta_l}, l_{\beta_r}) < l_G \\ 1 & \text{else} \end{cases} \tag{5.10}$$

The single-frame road course probabilities are recursively filtered with a binary Bayes filter, as shown next.

5.4.2 Recursive Bayesian Validation

In order to increase the robustness of the estimated road courses, the probability of each tracked road course is recursively filtered. Road course tracking, and thus also association, is described in the next section. For now assume that the individual single-frame probabilities are correctly associated to the corresponding tracks. Let $\xi_t = \{\xi_{t,1} \dots \xi_{t,4}\}$ and $p(\rho \mid \xi_t)$ denote the probability of road course ρ being valid in the current frame. Then, $p(\rho \mid \xi_{1:t})$ is calculated recursively from $p(\rho \mid \xi_{1:t-1})$ and the current estimation $p(\rho \mid \xi_t)$ using Bayes rule and the log odds ratio [157]

$$\lambda_t(\rho) = \log \frac{p(\rho \mid \xi_t)}{1 - p(\rho \mid \xi_t)} - \log \frac{p(\rho)}{1 - p(\rho)} + \lambda_{t-1}(\rho) \tag{5.11}$$

with

$$\lambda_0(\rho) = \log \frac{p(\rho)}{1 - p(\rho)}. \tag{5.12}$$

The probability of ρ being valid is recovered from (5.11) as

$$p(\rho \mid \xi_{1:t}) = 1 - \frac{1}{1 + \exp(\lambda_t(\rho))}. \tag{5.13}$$

The tracked probability $p(\rho \mid \xi_{1:t})$ is then used by the tracker to determine one of three possible states:

Potential Road Course: Denotes a new track candidate. If the single-frame probability $p(\rho \mid \xi_t)$ exceeds a certain threshold and the road course has not been associated to an existing track, a new track is created. If the track is successfully associated over multiple frames and the filtered probability $p(\rho \mid \xi_{1:t})$ exceeds a second threshold, the state of the track is changed to valid road course.

Valid Road Course: Denotes a stable track. There can be multiple valid road courses in order to represent road junctions or forks in the road.

Primary Road Course: Denotes the selected road course. It only exists if the separator path is used to navigate the vehicle, as described in Section 5.5.2.

The tracker, which is centered on the paths from the path planner, is presented in the following.

5.5 Road Course Tracking

In this section, it is shown how the road course hypotheses are tracked over time to assure consistency in the estimation as well as increased robustness. Section 5.5.1 presents the general idea of the tracker and Section 5.5.2 describes the association.

5.5.1 Tracking Road Courses based on Paths

The obvious and usual way of tracking road course estimations is by tracking the estimations themselves. Often, the parameters of the mathematical model that is used to represent them are tracked with a Bayes filter. The current estimations are predicted, associated, and correct using the current measurements.

If the road courses are estimated on an occupancy grid, i.e., a representation where the sensor measurements are already filtered over time, filtering the road course hypotheses with the occupancy grid as measurement, corresponds to filtering the data twice. Therefore, changes in the shapes of the estimations due to new measurements, may appear too slowly. Here, a different approach is used. Contrary to directly tracking the road courses, tracking is done indirectly based on the separator paths from the motion planner. In addition, the boundaries that form the road courses are extracted from the current grid in every time step. Hence, a quick reaction to changes in the environment is guaranteed, while keeping consistency.

Tracking deals with the problems of track creation, track deletion, track merging, and track splitting. They are all handled based on the result of the data association, which is discussed in the next section.

5.5.2 Path Association

Central to any tracker is the data association problem, i.e., the association of existing tracks to the current measurements or current-frame estimations. Two different approaches are presented in the following.

Nearest Cluster Association

The first variant presented associates the individual road course estimations based on the closest cluster. In every frame, a set of road courses $\mathcal{R}_{\text{curr}}$ is estimated with a set of cluster representatives $\mathcal{T}_{\text{rep}}$, which in turn come from a set of paths $\mathcal{T}_t$ planned from the current position of the vehicle in the grid. Every tracked road course $\rho \in \mathcal{R}_{t-1}$ is then associated to the current frame road courses $\mathcal{R}_{\text{curr}}$ in the following way. The closest path

$$\tau_j = \arg\min_{\tau \in \mathcal{T}_t} \Big(\max_i (||\pi_i^n - \tau_i^n||) \Big) \tag{5.14}$$

from the separator path π of the tracked road course $\rho \in \mathcal{R}_{t-1}$ is searched for in the set of paths $\mathcal{T}_t = \{\tau\}$. If τ_j and π are within the gating window, the road course $\varphi \in \mathcal{R}_{\text{curr}}$ that was estimated based on the cluster of τ_j is associated to the tracked road course ρ. Note that the separator path π needs to be resampled onto the current graph at time t for the computation of (5.14).

Having associated every $\rho \in \mathcal{R}_{t-1}$ to a $\varphi \in \mathcal{R}_{\text{curr}}$, the probability of ρ being a valid road course is updated as described in Section 5.4.2 with the probability of φ, which is calculated as shown in Section 5.4.1. The new tracked road courses $\mathcal{R}_t$ hold the boundaries of the current frame estimations $\mathcal{R}_{\text{curr}}$, thus they always correspond to the newest occupancy grid, which incorporates the newest sensor measurements.

If a tracked road course $\rho \in \mathcal{R}_{t-1}$ cannot be associated, it is deleted, as no similar path is found. To increase the chance that a similar path to the separator path π of the tracked road course $\rho \in \mathcal{R}_{t-1}$ will be found by the motion planner, the node extraction strategy from the A*-RRT planner from Algorithm 3.1, described in Section 3.2.4, is slightly modified. An additional method is added after the first goal path is found that is called with a low probability instead of the popRRT() method:

popTrackedPath(): The endpoint of one of the tracked paths is randomly picked. Similar to the popRRT() method, its closest node in the list of open nodes $\mathcal{L}_{\text{open}}$ is then searched for and chosen next for expansion.

It is noted that the number of node expansion iterations $n_{\text{iter}}^{\max}$ must be appropriate for this strategy to work. If $n_{\text{iter}}^{\max}$ is very low leading to a low number of goal paths $\mathcal{T}$, tracks will continuously be deleted and recreated, unless the gating window is not substantially increased or even disabled. For the applications in this work, it is always assumed that the local environment is rather densely sampled with goal paths, which implies a high number of expansion iterations $n_{\text{iter}}^{\max}$.

Furthermore, if two tracked road courses are associated to the same cluster, then they are merged. If a current frame road course $\varphi \in \mathcal{R}_{\text{curr}}$ cannot be associated, but has a high probability $p(\varphi \mid \xi_t)$, a new track is created with the state of *Potential Road Course*

as described in Section 5.4.2. Tracks are never split though. It cannot be differentiated between a split due to a fork in the road corresponding to a correct split, and a split due to an estimation that leaves the true road course between sparse boundary elements. In the latter case, it is not desired that the filtered probability as well as a potential stable tracking state is copied.

Direct Association

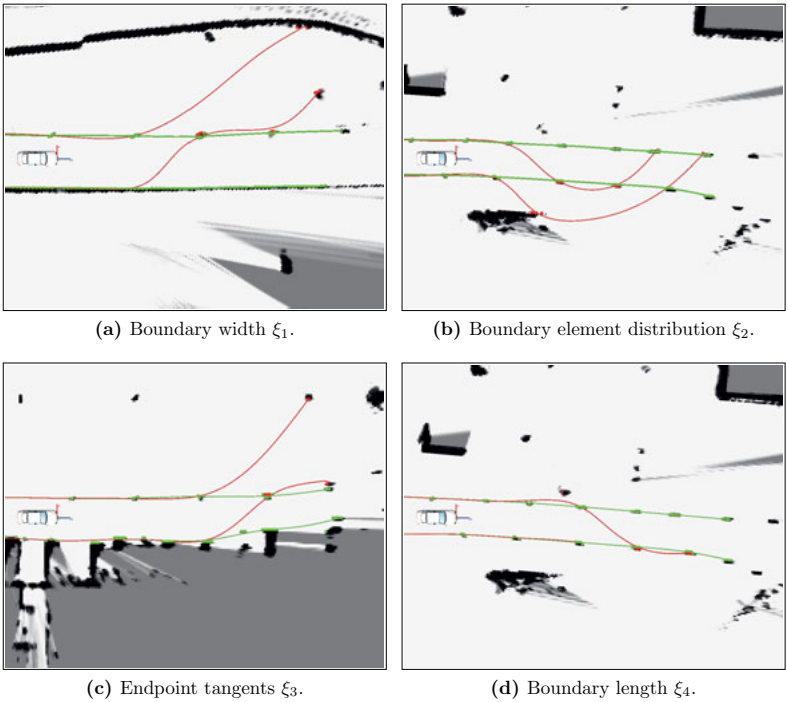
If the vehicle follows one of the paths from the planner, the road courses can directly be associated. In Section 6.2.4 of the next chapter, an experiment is conducted, where a vehicle is maneuvered autonomously solely based on the estimated road course without any prior map information. In this case, the paths from the motion planner, which are used and meant to estimate the road course, are used to navigate the vehicle as well. In a general autonomous vehicle system, one is typically not interested in using the paths for navigation as more information needs to be incorporated, such as other vehicles, lane markings, or traffic rules.

If one of the paths is used for maneuvering the vehicle, it is clear that re-planning in every cycle from the current pose of the vehicle leads to discontinuities and instabilities. Instead, in this case, the search space of the path planner, i.e., the graph, is fixed in the world coordinate system. The coordinate systems are described in Appendix A.3. Contrary to keeping and re-evaluating all paths from the previous time step, in every cycle, a new set of paths $\mathcal{T}_t$ is generated based on the old start pose, which has been used to plan at time $t-1$. This fosters exploration and allows detecting new branches. However, the primary path τ_p of the primary road course that the vehicle is following, as well as all valid tracked road courses, are directly inserted into the set $\mathcal{T}_t$ and hence data association is directly accomplished without any distance based measures. In order to keep the bias of re-inserting the tracked paths low, only the first couple of path nodes are inserted, while the rest of the path is chosen so that the overall path cost is minimized. Once the vehicle traverses the second node of the primary path τ_p, this second node becomes the new start pose of the motion planner, and the search graph is expanded by one level.

Due to the used motion primitives, τ_p is not feasible and not optimized with respect to a comfortable trajectory. One the one hand, the curvature is not continuous at the path nodes, and on the other hand, the number of primitives typically is very low and thus a strong discretization of the steering angles exists. Therefore, the path is smoothed for the aforementioned autonomous navigation experiment. A variety of smoothing, trajectory generation, and optimization techniques may be applied, such as [49, 182]. In Appendix A.4 the smoother that was used in the autonomous drive presented in Section 6.2.4 is described.

5.6 Results

This section gives first qualitative results. A quantitative evaluation of the presented road course estimation system follows in the next chapter. In Figure 5.3 four scenarios are shown. Each image presents an example of one of the four validation criteria presented in Section 5.4.1: the semantic road boundary width in Figure 5.3a, the road boundary



(a) Boundary width ξ_1.

(b) Boundary element distribution ξ_2.

(c) Endpoint tangents ξ_3.

(d) Boundary length ξ_4.

Figure 5.3: Sample valid (green) and invalid (red) estimated road courses for each of the presented criteria. The probabilities are given in Table 5.1.

element distribution in Figure 5.3b, the boundary endpoint tangent in Figure 5.3c, and the road boundary length in Figure 5.3d. Two road course estimations are shown in each figure, one of the valid road course in green and one of an invalid estimation in red that was successfully detected because of the corresponding validation criterion.

Table 5.1 gives the detailed probabilities of both estimations for each figure and for each criterion as well as the joint probabilities. The values were clamped to the interval $[0.15, 0.85]$ to reduce the impact of a single estimation in the temporal filtering.

Table 5.1: Road course plausibility values for Figures 5.3a–5.3d.

	$p(\xi_1 \mid \rho)$	$p(\xi_2 \mid \rho)$	$p(\xi_3 \mid \rho)$	$p(\xi_4 \mid \rho)$	$p(\rho \mid \xi_{1:4})$
Figure 5.3a, valid RC	**0.797**	0.510	0.850	0.850	0.850
Figure 5.3a, invalid RC	**0.163**	0.451	0.668	0.850	0.240
Figure 5.3b, valid RC	0.767	**0.738**	0.822	0.845	0.850
Figure 5.3b, invalid RC	0.677	**0.193**	0.842	0.799	0.555
Figure 5.3c, valid RC	0.628	0.691	**0.830**	0.850	0.850
Figure 5.3c, invalid RC	0.356	0.155	**0.286**	0.850	0.150
Figure 5.3d, valid RC	0.777	0.600	0.823	**0.850**	0.850
Figure 5.3d, invalid RC	0.659	0.550	0.817	**0.267**	0.364

5.7 Summary

The boundaries of the road, which compose the road course, are important extracted and interpreted information about the environment. Contrary to the variety of arbitrary static obstacles in the world, they represent the ones, which define where the road is heading and thus where the vehicle is supposed to maneuver.

A novel approach to road course estimation has been presented in this chapter. It is independent of specific features or particular sensors, but is based on occupancy grids. In addition to an abstraction from the sensor level, the grid representation assures a high robustness over noise and with recent methods, such as the one proposed in Chapter 2, also to be free of dynamic obstacles, which do not constitute to the road boundaries. The method allows an estimation of multiple road courses at road junctions and can cope with arbitrary-shaped and -structured road boundaries over a large planning horizon.

The approach estimates the road boundaries indirectly based on the principal moving directions through the environment. A method for their extraction using motion planning has been presented in Chapter 3. An estimated drivable velocity reduces the search space of the path planner and significantly decreases, together with collision constraints, the number of false positive principal moving directions in the case of sparse boundaries. The principal moving directions are seen as boundary separators and greatly facilitate the estimation of the boundaries.

Given the separators, the boundary obstacle elements are extracted and a continuous semantic representation of the boundaries is obtained. This is especially useful in scenarios with road boundaries consisting of sparse boundary elements, such as often exhibited in road construction sites. Finally, the extracted boundaries are validated and filtered over time, in addition to the applied constraints during the extraction of the principal moving directions. In the following chapter, the proposed methods are evaluated.

6 Evaluation

In this chapter the proposed methods are evaluated. It is noted that each of the previous chapters already presented results. The path planning and clustering have already been evaluated in Chapter 3, as well as the proposed methods for the calculation of the configuration space costs in Chapter 4. Here, it is focused on an evaluation of the grid-based tracking and mapping and the road course estimation system. In addition to the qualitative results given in Chapter 2 and Chapter 5, quantitative evaluations are presented in Section 6.1 and Section 6.2. Furthermore, the road course estimation method was used to autonomously maneuver a vehicle through an unmapped road construction site. Finally, results from the road course estimation in dynamic environments with the static Bayesian map from GTAM are given. Parts of the evaluations of the road course estimation have been presented in [10] in the context of this work.

6.1 Evaluation of the Grid-based Tracking and Mapping

In this section GTAM, presented in Chapter 2, is evaluated. First, in Section 6.1.1 the particle convergence is analyzed and compared to the approach of [35]. Then, in Section 6.1.3 the classification is evaluated in a variety of real street scenarios that were labeled manually. The classification is done for different parameters to show their effect. Finally, in Section 6.1.4 the estimated velocities are evaluated and compared to a commercial object tracking system. If not mentioned otherwise, the evaluation was done with data from a laser scanner. Rather than with data from a radar sensor, no information about the velocity is available.

6.1.1 Particle Convergence with Static Particle Sampling

First, the convergence of the particle filter is analyzed. One of the characteristics of the presented particle filter from Section 2.3 is the initial velocity sampling distribution from (2.22). Contrary to sampling from a pure uniform distribution as in [35], a uniform distribution is combined with a Dirac distribution at $(0, 0)$ to allow the representation of the static environment. To show how this distribution impacts the convergence of the filter, the algorithm was tested in a synthetic static scenario. The exact same scan grid was used as input to the filter in every time step, in order to exclude effects from sensor noise. Let $\mathcal{X}_{\text{del}}$ denote the set of particles that were deleted after the resampling step and $\mathcal{X}_{\text{surv}} = \mathcal{X} \backslash \mathcal{X}_{\text{del}}$ the set of particles that have survived the resampling procedure. Figure 6.1 shows a plot of the *Particle Destruction Rate*

$$\text{PDR} = \frac{|\mathcal{X}_{\text{del}}|}{|\mathcal{X}_{\text{del}}| + |\mathcal{X}_{\text{surv}}|}, \tag{6.1}$$

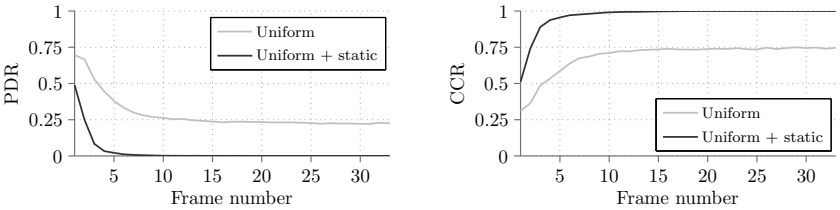


Figure 6.1: Comparison between uniform sampling and uniform + static sampling as proposed. The particle destruction rate (left) and the cell convergence rate (right) are given. The same input is applied in every time step.

i.e., the relative number of deleted particles. Compared are sampling from a uniform distribution only and sampling with (2.22) as proposed. The probability of sampling a static particle was set to 0.3. A high PDR indicates that a large amount of the total particle population moves into areas that do not receive sensor data and is therefore deleted. These particles are an overhead to the system as they do not support the measurements. In addition, if many particles move out of the relevant areas and a particle survival probability $P_{\text{surv}} > 0$ is used, in order to cope with occlusions and missed detections, a high computational overhead is introduced due to a high number of particles in such areas.

Furthermore, the average *Cell Convergence Rate*

$$\text{CCR} = 1 - \frac{|n_\chi^{i,\text{des}} - n_\chi^i|}{n_\chi^{i,\max}}, \tag{6.2}$$

i.e., the relative difference between the desired number of particles in a cell $n_\chi^{i,\text{des}}$ and the actual number of particles n_χ^i for all cells with a non zero cell weight, is given in Figure 6.1. It shows how well the particles have converged towards the static objects.

As expected, with a pure uniform sampling distribution, the filter does not converge towards a stable result. Around 1/4 of the total particle population constantly moves out of the static objects and is therefore deleted in every time step. The result of the CCR is very similar. Also, there exists a mismatch between the desired and the actual number of particles of around 1/4. If static particle sampling is introduced and combined with uniform sampling, the filter quickly converges after a couple of iterations to a stable result.

6.1.2 Parameter Evaluation

Next, the main different parameters are analyzed and evaluated, i.e., the maximum number of particles per cell, the effect of sampling new random particles during resampling, and the probability of sampling static particles.

Evaluation Metric and Ground Truth

The parameters were evaluated on a variety of real street scenarios including urban street scenarios, one way streets, road junctions, as well as highways. Several sample images

from the test data are given in Figure A.3 of the Appendix. The dynamic objects were manually labeled in the data set to generate the static and dynamic ground truth for every input scan grid. In order to evaluate the continuous Dempster–Shafer belief masses from the particle map from (2.38) with the binary ground truth, the classifier

$$\text{classifyGTAM}(\nu^i) = \begin{cases} S & \text{if } m^i(S) = \max\limits_{A \in \{S,D,\{S,D\}\}} m^i(A) \\ D & \text{if } m^i(D) = \max\limits_{A \in \{S,D,\{S,D\}\}} m^i(A) \\ \text{undecided} & \text{else} \end{cases} \tag{6.3}$$

is applied on all cells for which $m(S) + m(D) + m(\{S, D\}) > 0$. It classifies according to the maximum of $m(S)$, $m(D)$, and $m(\{S, D\})$. Then, using the labeled ground truth the rates for *True Dynamic* (TDR), *True Static* (TSR), *False Dynamic* (FDR), *False Static* (FSR), *Undecided Dynamic* (UDR), and *Undecided Static* (USR), defined as

$$\begin{aligned} \text{TDR} &= \frac{\text{true dynamics}}{\text{true dynamics + false statics}} & \text{TSR} &= \frac{\text{true statics}}{\text{true statics + false dynamics}} \\ \text{FDR} &= \frac{\text{false dynamics}}{\text{false dynamics + true statics}} & \text{FSR} &= \frac{\text{false statics}}{\text{false statics + true dynamics}} \\ \text{UDR} &= \frac{\text{undecided dynamics}}{\text{dynamics}} & \text{USR} &= \frac{\text{undecided statics}}{\text{statics}} \end{aligned} \tag{6.4}$$

are calculated. They reflect the observation that an undecided classification, i.e., where the mass of the set $\{S, D\}$ is the highest, is not a false classification. The rates give the relative number of unclassified cells. Note that in the evaluation, the undecided rates will be relatively high, since all cells for which $m(S) + m(D) + m(\{S, D\}) > 0$ are classified. Hence, also cells that are observed to be occupied for the first time are classified. Since the vehicle is moving for most of the time, there are many of such cells. As described in Section 2.3.4, only the particles that have survived at least t time steps are accounted for the classification. Hence, for those cells $m(\{S, D\})$ is the maximum and they will contribute to the UDR and the USR.

Maximum Number of Particles per Cell

The maximum number of particles per cell $n_\chi^{i,\max}$ has a crucial impact on the computational performance, since it controls, together with the cell resolution, the total number of particles. In Table 6.1 the evaluation rates from (6.4) on the test data are given for 5 different values of $n_\chi^{i,\max}$. Rather surprisingly, the rates regarding the classification do not change much with the number of particles. Even with a low number of particles, high true dynamic rates and true static rates are achieved. Note that the undecided dynamic rate and the undecided static rate, as mentioned above, are rather high due to the classifier of (6.3). The influence of the number of particles is further analyzed in Section 6.1.4 regarding the estimated velocities. Overall, setting the maximum number of particles to 32 is the most appropriate setting for the tested scenarios.

Table 6.1: Classification performance with varying number of maximum particles per cell.

	Particles per cell				
	16	32	64	128	256
True dynamic rate (TDR)	0.9628	0.9634	0.9697	0.9692	0.9718
False dynamic rate (FDR)	0.1365	0.0845	0.0835	0.0723	0.0702
Undecided dynamic rate (UDR)	0.4589	0.4710	0.4336	0.4294	0.4088
True static rate (TSR)	0.8635	0.9155	0.9165	0.9277	0.9298
False static rate (FSR)	0.0372	0.0366	0.0303	0.0308	0.0282
Undecided static rate (USR)	0.3244	0.3131	0.3088	0.3058	0.3038

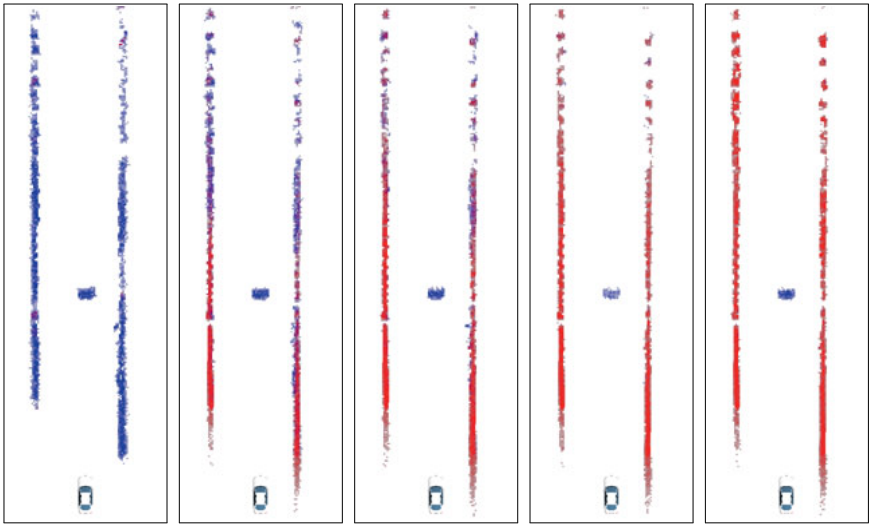
New Random Particles

If resampling, i.e., generating the new particle population from the one at time $t-1$, is only done with the existing particles, i.e., the ones that moved into that particular cell from the previous time instance, the particles quickly converge towards one hypothesis and artifacts may occur. The velocity distribution may be badly represented by the particle population, since all particles converge to one estimate. No particles remain in the other areas of the distribution. This problem is called the particle deprivation problem [157].

Within the presented filter, these artifacts are most clearly visible at objects exhibiting a uniform longitudinal shape parallel to the moving direction of the robot. Such objects, unfortunately, occur often in real street scenarios, e.g., guard rails or tunnel walls. Due to sensor limitations, only a small fraction of relatively constant length of the boundary object is visible. If the robot moves, new parts of the object become visible, while other parts passed by the robot, disappear. Cells of such stationary objects cannot be discriminated from cells of objects that move with the same speed and in the same direction as the robot. The particles therefore quickly converge towards such an estimate, as depicted in Figure 6.2a. All cells corresponding to the stationary road boundary are filled with particles that move similar to the ego vehicle and therefore have a high dynamic mass. If a small fraction of new random particles is inserted into the existing particle population, this problem is mitigated, as shown in Figure 6.2b–d. Even a low fraction of random particles, such as in Figure 6.2b, has a high effect on the resulting estimation. It comes, however, at the price of destroying the existing tracks with new particles that are randomly created and have not yet caught the real environment, such as visible in the dynamic object in Figure 6.2d.

For purposes of computational efficiency, new random particles are only inserted when the number of particles in a particular cell is increased from the number of particles from $t-1$ after the motion update. Due to noise added during the motion update, this is, however, constantly the case for a certain number of particles. A constant number of $\left\lfloor w_{\text{rand}}\, n_{\chi}^{i,\max} \right\rfloor$ random particles is created at most.

If velocities, or parts of it, can be measured, such as with a radar sensor, individual particle weights, i.e., intra-cell weights, can be calculated, as discussed in Section 2.3.3.



(a) $w_{\text{rand}} = 0\,\%$ lidar only. **(b)** $w_{\text{rand}} = 1\,\%$ lidar only. **(c)** $w_{\text{rand}} = 5\,\%$ lidar only. **(d)** $w_{\text{rand}} = 30\,\%$ lidar only. **(e)** w_{rand} from (6.5) lidar + radar.

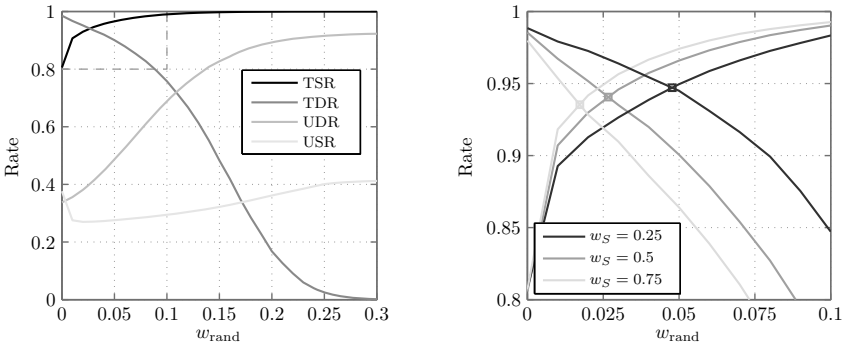
Figure 6.2: Effect of increasing the cell particle population with new random particles instead of just duplicating them to fight particle deprivation. Scene consists of two guard rails to the left and the right of the robot and one moving vehicle to the front. Figures (a)–(d) show results on laser scanner data only, whereas Figure (e) shows position sampling and cell weighting with a laser scanner, and velocity sampling and intra-cell weighting with a radar sensor.

Hence, the choice of the value of the parameter w_{rand} can be determined automatically. The particle weights are used to calculate w_{rand} individually for each cell according to

$$w^i_{\text{rand}} = f_w\Big(1 - \max_k w^i_{[k]}\Big), \tag{6.5}$$

where each $w^i_{[k]}$ is normalized between 0 and 1. As before, the index i denotes the cell index and f_w a weighting function. Hence, if a cell contains one or more particles with a high weight according to the current measurement, then w_{rand} is low or even zero. Dynamic cells from real moving objects are therefore not affected by the random particles, such as in Figure 6.2e. If the maximum particle weight in cell i is, however low, then w_{rand} increases. This prevents the filter from converging against a wrong estimation in static boundaries. In Figure 6.2e, the same scene is shown, where only the laser scanner data was used for the position sampling of the particles and for calculating the cell weights, and the radar sensor was used for the velocity sampling and for calculating the intra-cell weights.

The effect of sampling random particles during resampling was also evaluated quantitatively. Figure 6.3a shows the true dynamic rate, the true static rate, the undecided dynamic rate, and the undecided static rate over different values of w_{rand}. The higher



(a) TSR, TDR, UDR, and USR for $w_S = 0.5$.

(b) TSR and TDR for different values of w_S.

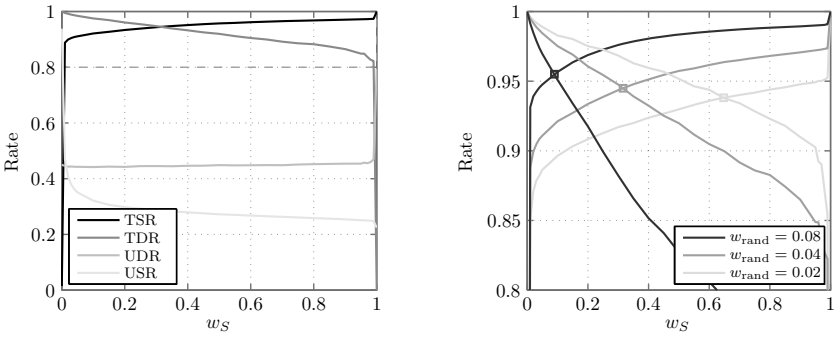
Figure 6.3: Plots over the ratio w_{rand} of sampling random particles during resampling. Left, true static rate, true dynamic rate, undecided static rate, and undecided dynamic rate for the probability of sampling static particles w_S during initial sampling of 0.5. Right, cutout for different values of w_S.

w_{rand}, the higher is the true static rate, since the problem discussed above is mitigated. The true dynamic rate, however, decreases, since the existing tracks are destroyed through the new random particles. Moreover, with increasing values of w_{rand} the undecided rates, in particular the undecided dynamic rate, increase. Less cells are classified, since less cells hold enough particles that have survived at least t steps. The parameters w_{rand} and w_S, i.e., the probability of sampling static particles during initial sampling, analyzed in the following section, are dependent on each other. Therefore, in Figure 6.3b the curves for the TSR and the TDR are given for different fixed values of w_S. The points of interest, i.e., the intersection of the TSR and the TDR, are marked with squares.

Pure Static Particles

Similar to the previous section, the prior w_S of sampling static particles in the initial sampling distribution from (2.22) is evaluated. In Figure 6.4a, the true static rate, the true dynamic rate, the undecided static rate, and the undecided dynamic rate are plotted over different values of w_S. As expected, the higher w_S, the higher is the true static rate, while the true dynamic rate decreases. The undecided rates stay relatively constant with the undecided static rate slightly decreasing with increasing w_S, since more static particles are initially generated and therefore the static world can be better classified.

Similar to Figure 6.3, the top left part of the plot is presented in Figure 6.4b, in addition to the full plot, in greater detail and for different values of w_{rand}. As mentioned above, the two parameters, w_{rand} and w_S influence each other. From Figure 6.3 and Figure 6.4 it follows that it is beneficial to combine a low value of w_S with a high value of w_{rand}, and vice-versa, a high w_S with a low value of w_{rand}, in order to achieve high true dynamic and true static rates. With a pair of low w_S and high w_{rand}, the highest TSR and TDR are



(a) TSR, TDR, UDR, and USR for $w_{\text{rand}} = 0.04$. **(b)** TSR and TDR for different values of w_{rand}.

Figure 6.4: Plots over the probability of sampling static particles during initial sampling w_S. Left, true static rate, true dynamic rate, undecided static rate, and undecided dynamic rate for the ratio w_{rand} of sampling random particles during resampling of 0.04. Right, cutout for different values of w_{rand}.

achieved, although the difference is marginal. The undecided rates are, however, rather high with such a parameter pair. If a high number of actual classifications is desired, then a pair of high w_S and low w_{rand} is recommended. For the application in this thesis, the intermediate way is chosen, i.e., a w_S of 0.5 and a w_{rand} of 0.03.

6.1.3 Classification

In this section, the classification with the proposed initial velocity sampling distribution is compared to the classification of [35] with pure uniform sampling. The evaluation data set was the same as in the previous section.

Evaluation Metric

In order to compare the approach to the classification of [35], a different classifier than (6.3) is used, since the classifier of [35] is binary. Slightly different to the previous section, only the cells in which there exist at least $\lfloor 0.2\, n_\chi^{i,\max} \rfloor$ particles that have survived t steps are classified according to

$$\text{classifyBinaryGTAM}(\nu^i) = \begin{cases} S & \text{if } m^i(S) \geq m^i(D) \\ D & \text{else} \end{cases} \tag{6.6}$$

and compared to the classification of [35], i.e.,

$$\text{classifyDanescu}(\nu^i) = \begin{cases} S & \text{if } \mu^i_{v_x} < w\sigma^i_{v_x} \wedge \mu^i_{v_y} < w\sigma^i_{v_y} \\ D & \text{else} \end{cases} \tag{6.7}$$

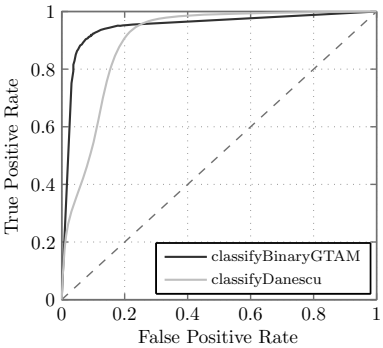


Figure 6.5: Receiver operating characteristic curve of (6.6) and (6.7).

Note that the classification with (6.7) is done with pure uniform velocity sampling as used in [35]. The rest of the particle filter, such as weighting and resampling, is done as given in this work.

Results

In Figure 6.5 the receiver operating characteristic (ROC) curve on the labeled ground truth data is given. The ROC parameter for classifyBinaryGTAM was w_S, i.e., the probability of sampling static particles. It was varied in the range $[0, 1]$. The parameter for classifyDanescu was the weight w from (6.7). It was varied in the range $[0, 20]$. Contrary to the previous section, the classifiers here are binary, and therefore only the true positive rate over the false positive rate is given. Positive, in this case, was the class dynamic, but due to the symmetry of the ROC curve, the class static is simply mirrored and thus not given. Additionally, there are no undecided rates. Every cell with enough old particles is classified with both classifiers. Hence, the number of classifications is equal.

It is noted that for this evaluation, the power of the estimation result was significantly reduced. In fact, there exist continuous masses in the Dempster–Shafer theory of evidence for the class static $\{S\}$, dynamic $\{D\}$, static-dynamic $\{S, D\}$, as well as unknown $\{\Theta\}$, instead of only a binary two-class classifier. The proposed classifier still performs significantly better.

6.1.4 Estimated Velocities

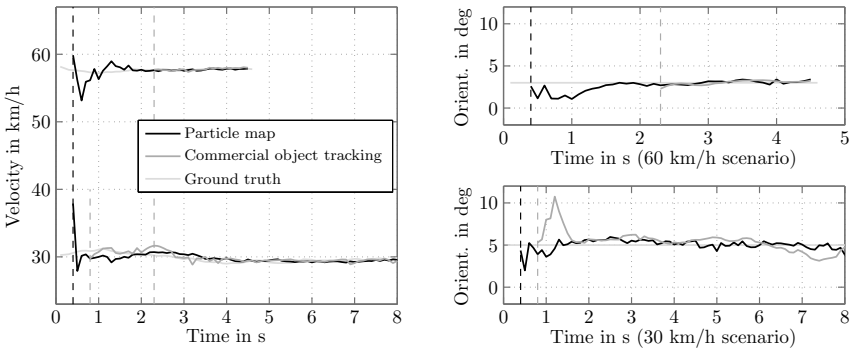
Finally, the estimated velocities are evaluated. They are compared to a commercial object tracking that works with the same sensor. Note, that the grid-based tracking is entirely cell-based and no higher level object representation exists. As creating objects out of the grid representation, such as with a clustering algorithm, was not topic of this work, the cells corresponding to the dynamic object under evaluation were manually clustered by drawing a box around them. The velocity and the orientation of the object were calculated

by taking the weighted average of the mean velocity of all dynamic cells weighted by their dynamic evidences $m^i(D)$.

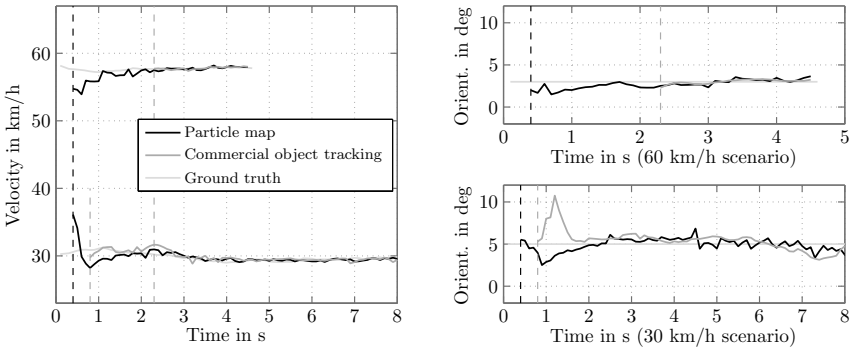
Two scenarios are shown: one in which the dynamic object, i.e., another vehicle, is passing by in parallel, shown in Figure 6.6, and one in which the object passes at around 45 degrees, shown in Figure 6.7. In each scenario the object passes with two different speeds, around 30 and 60 km/h. Moreover, the tests are done with two different maximum numbers of particles per cell, 32 and 128, in order to evaluate if the results are improved if the number of particles is increased. Rather surprisingly, but consistent with the observations from Section 6.1.2, the difference is hardly noticeable in Figure 6.6 and moderate in Figure 6.7. Hence, the computational overhead of increasing the maximum number of particles per cell is usually not worth the improvement.

The difference in the accuracy between the commercial object tracking and the grid-based estimation is also moderate, although the particle map shows slightly more noise, especially in the orientation. Note, that this is a good result for the particle map, since no higher-level object-based information is used. It is expected that an object-to-object association and filtering is able to estimate more stable and accurate results as an independent set of cells. The biggest difference, however, is the reactiveness of the particle map. In this evaluation, particles had to have survived at least 4 time instances, i.e., 0.4 s in this setup, to be accounted for. Hence, every plot shows this constant offset. The commercial object tracking needs substantially longer to detect the dynamic object as such. If a separate object tracking is now used, in order to filter out the dynamic object in an occupancy grid map, there will always be a certain delay in which the dynamic objects will produce artifacts. Note, that although it also takes the particle map 0.4 s with this parameter set to yield velocity and orientation estimations, static and dynamic evidences are directly inferred from these velocities as described in Section 2.3.4. Therefore, the resulting uniform Dempster–Shafer environment representation and thus also the Bayesian occupancy grid do not have this issue.

(a) Camera image.



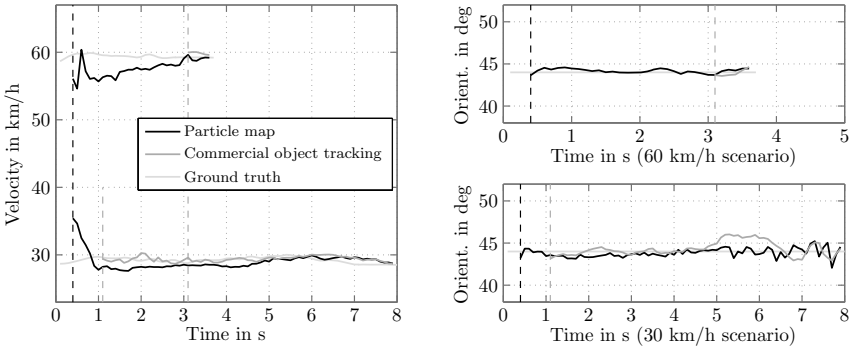
(b) Maximum number of particles per cell $n_{\chi}^{i,\max} = 32$.



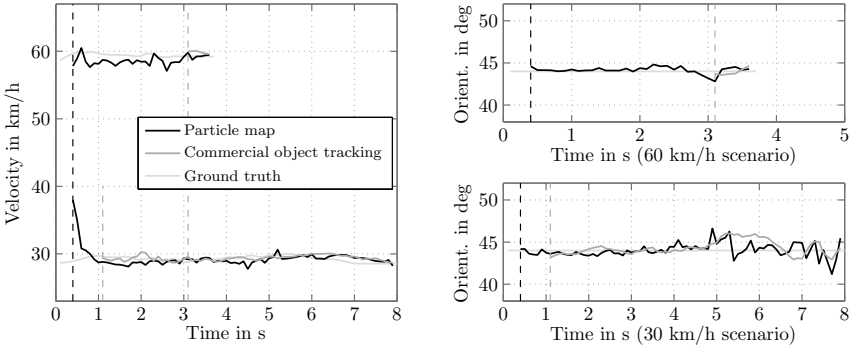
(c) Maximum number of particles per cell $n_{\chi}^{i,\max} = 128$.

Figure 6.6: Evaluation of the estimated velocities in comparison to a commercial object tracking. Estimated velocity (left) and orientation (right) of an object passing the ego vehicle in parallel.

(a) Camera image.



(b) Maximum number of particles per cell $n_{\chi}^{i,\max} = 32$.



(c) Maximum number of particles per cell $n_{\chi}^{i,\max} = 128$.

Figure 6.7: Evaluation of the estimated velocities in comparison to a commercial object tracking. Estimated velocity (left) and orientation (right) of an object passing the ego vehicle at around 45 degree.

6.2 Evaluation of the Road Course Estimation

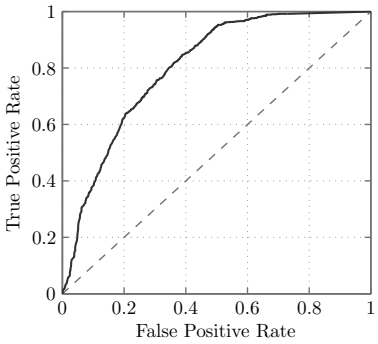
This section presents an evaluation of the road course estimation. In Section 6.2.1 and Section 6.2.2 the validation and the accuracy are analyzed respectively. Both evaluations were conducted on a data set, which includes a variety of boundaries from different real street scenarios, such as highways, urban areas, and road construction sites. The test set includes guardrails, concrete walls, fences, walls of buildings, vegetation, traffic cones and other sparse traffic markers, parked vehicles, and plastic barriers. Several sample images from the test set are shown in Figure A.4 of the Appendix. The ground truth road course in the data set was labeled manually.

Next, in Section 6.2.3 the approach is compared to separating the environment with the predicted vehicle trajectory, as done by many grid-based road course estimation methods as described in Chapter 5. The comparison was done on a road construction site marked with traffic cones. Finally, Section 6.2.4 presents results from the autonomous drive. All evaluations were conducted on a standard occupancy grid from laser scanner data in static environments, in order to focus on the road course estimation alone.

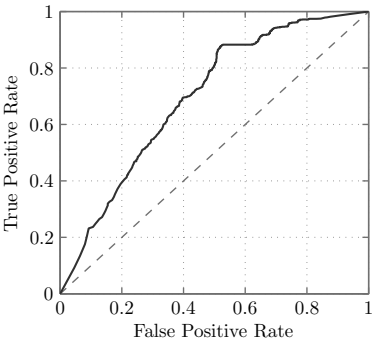
6.2.1 Road Course Validation

Here, an evaluation of the road course validation, presented in Section 5.4, is given. Each criterion $\xi_1, ..., \xi_4$ is analyzed individually. Only the single frame evaluations are given, not the tracked ones, in order to focus on the validation. Figure 6.8 shows receiver operating characteristic curves for the different criteria. They are generated by computing $p(\xi_j \mid \rho_i) > a$ and $p(\xi_j \mid \rho_i) \leq a$ with varying parameter $a \in [0, 1]$ and by comparing to the labeled ground truth. The comparison to the ground truth is done with the separator path of the estimations by checking whether it stays within the ground truth or whether it crosses the ground truth boundaries. The result of the ROC curves was then also used to tune the weighting functions f_w.

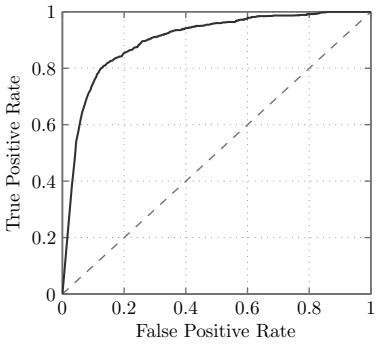
The best overall classification is achieved by the equal endpoint tangent criterion ξ_3. Also, the constant boundary width criterion ξ_1 performed well. The performance of the uniform boundary criterion ξ_2 and the length criterion ξ_4 depend on the particular scenario. Criterion ξ_2 performs well in organized, well-structured environments, such as highway construction sites, but fails if the boundary elements are positioned arbitrarily. The validation with the joint probability from (5.4) performs the best over all scenarios.



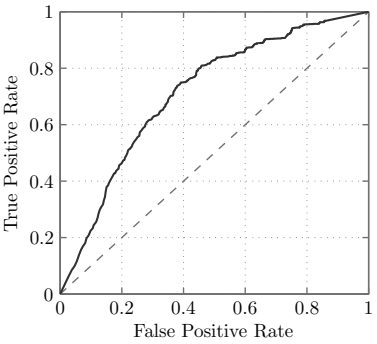
(a) Constant boundary width ξ_1 (5.5).



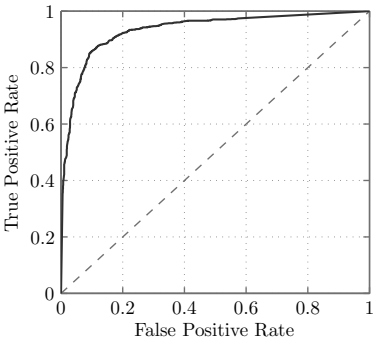
(b) Uniform boundary ξ_2 (5.6).



(c) Equal endpoint tangents ξ_3 (5.8).



(d) Boundary length ξ_4 (5.10).



(e) Joint probability $\xi_{1:4}$ (5.4).

Figure 6.8: ROC curves for the evaluation of the single frame road course validation.

Table 6.2: Accuracy of tracked primary estimated road course for different planning horizons.

	Planning horizon		
	30 m	50 m	70 m
Average deviation (in cells @ 0.1 m)	1.37	2.22	2.58
Average boundary length (in m)	27.7	46.9	62.13
True positive rate	0.99	0.96	0.95

6.2.2 Boundary Estimation Accuracy

In this section, the accuracy of the estimated boundaries of the tracked, primary road course is evaluated. It is compared to the labeled ground truth. Table 6.2 shows the average deviation of the boundaries with three different planning horizons: 30, 50, and 70 meters. It is expected that since the ground truth boundaries are generated manually, additional inaccuracies are introduced. The average deviation is within 1-3 cells. Also given, is the average length of the estimated boundaries. It can be observed that in 70 m, the average estimated length deviates by 8 meter. This is due to limits from the sensor and its mounting position. Increasing the planning horizon does therefore not necessary mean that the road course can be estimated over a greater distance. Furthermore, the true positive rate of the primary tracked road course is shown. It reflects the recursively filtered probability of the road course validation, as well as the dynamic and collision constraints.

6.2.3 Comparison to Predicted Vehicle Path

Next, the approach is compared to separating the environment simply based on a predicted vehicle path, as done by many state-of-the-art grid-based road course estimation approaches, instead of using motion planning. To this end, the motion planner was disabled and replaced by the predicted vehicle path. Two different models are used, a constant heading model, i.e., the vehicle orientation separates the environment, and a constant curvature model based on the current angle of the front wheels. The comparison was done on an S-shaped curve made up of traffic cones at a distance of around 7 m with a road width of around 3.5 m.

Two variants of each type of predicted vehicle trajectory were applied as the approaches in the literature differ. On the one hand, the predicted trajectory was checked for collision in the workspace, denoted by $\mathcal{W}$, i.e., it was followed until the first occupied grid cell was found. And on the other hand, collisions where checked, as in this work, in the configuration space, denoted by $\mathcal{C}$. In Table 6.3, the true positive rate of the different variants are given. As expected, with the collision check in the workspace, the true positive rate is low, since the boundaries are sparse. In the configuration space, using the predicted trajectory, the same true positive rate as with using motion planning is achieved in this particular scenario. In this setup, due to the collision constraints, the predicted trajectory does not pass in between two boundary elements, although in other scenarios this may easily happen. However, Figure 6.9 shows the length of the separator path until the first collision for the approaches in the configuration space. It is clearly visible that with the

Table 6.3: Evaluation against using the predicted vehicle path as boundary separator for an S-shaped curve marked with traffic cones.

	This work	Const. heading		Const. curvature	
	$\mathcal{C}$	$\mathcal{W}$	$\mathcal{C}$	$\mathcal{W}$	$\mathcal{C}$
True positive rate	1.0	0.51	1.0	0.82	1.0

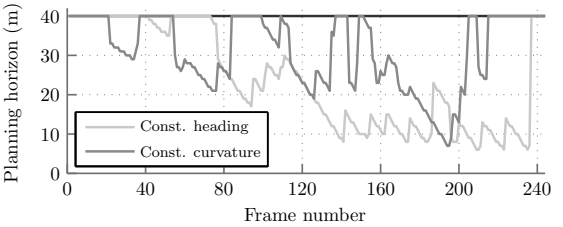


Figure 6.9: Length of the separator path until first collision in the configuration space $\mathcal{C}$ for an S-shaped curve marked with traffic cones. The maximum value, set to 40 m, is continuously reached by the path planner of this work.

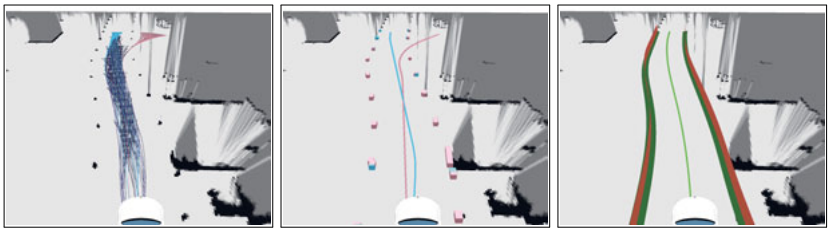
predicted trajectory, the length, which is set to a maximum of 40 m, can rarely be reached, while it is constantly reached when using motion planning.

In addition, it is noted that with using the predicted trajectory branches, junctions, and forks in the road cannot be detected. Moreover, it is strongly relied on a good start state, i.e., a correct orientation of the vehicle and, in the case of the constant curvature model, also a correct wheel angle.

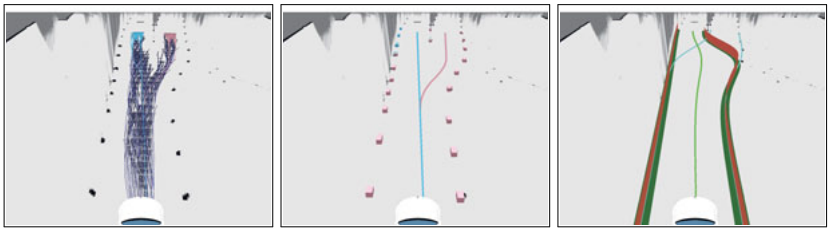
6.2.4 Autonomous Navigation in an Unmapped Road Scenario

The road course estimation system was also integrated into a vehicle and used to control it through an unmapped road construction site. The boundaries were marked with traffic cones, plastic barriers, parked vehicles, vegetation, and walls of buildings. Images of the scene are shown in Figure 6.11. No GPS or prior map of the environment was used, but only the data from a single 4-layer laser scanner.

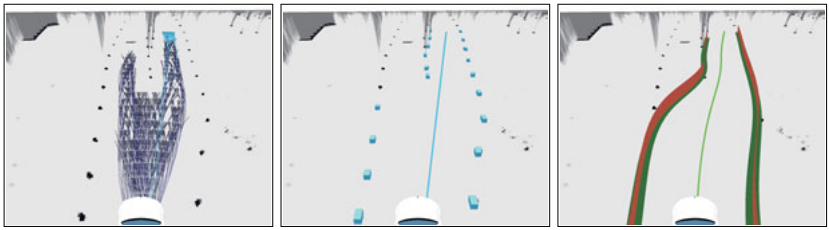
In Figure 6.10 the clustered paths, the path representatives with the extracted boundary cells, and the semantic boundaries with the smoothed trajectory that was the input of the controller are shown. Four different time steps are given. Figure 6.10a shows an S-curve, where two clusters were detected according to the paths that leave the valid road course. Due to the validation and the tracking, such wrong hypotheses are eliminated. Figure 6.10b shows a fork in the road. This time the two clusters correspond to two real road courses and both are successfully detected. A couple of frames later, in Figure 6.10c, the system detects that one of the branches is blocked. The tracked primary road course switched to the branch to the right. Finally, Figure 6.10d shows a strong curve. Figure 6.12 gives the driven trajectory and highlights the positions of the presented time instances.



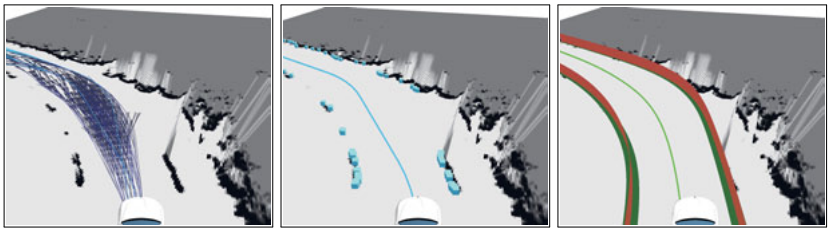
(a) $t = 15.2\,\mathrm{s}$.



(b) $t = 27.4\,\mathrm{s}$.

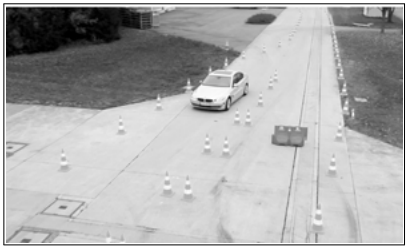


(c) $t = 30.2\,\mathrm{s}$.



(d) $t = 55.7\,\mathrm{s}$.

Figure 6.10: Autonomous navigation through an unmapped street scenario at four different time instances. The left column shows the clustered paths, the middle column the cluster representatives and the extracted boundary cells, and the right column the semantic boundary of the primary and all valid road courses as well as the smoothed trajectory for the controller.



(a) Overview of the setup.

(b) $t = 15.2s$.

(c) $t = 27.4s$.

(d) $t = 30.2s$.

(e) $t = 55.7s$.

Figure 6.11: Camera images from the results from Figure 6.10.

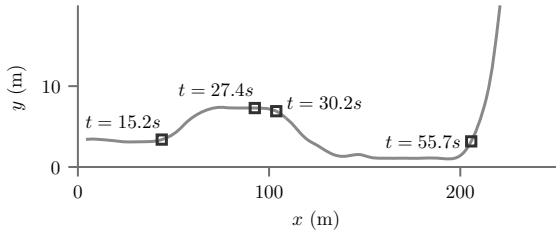


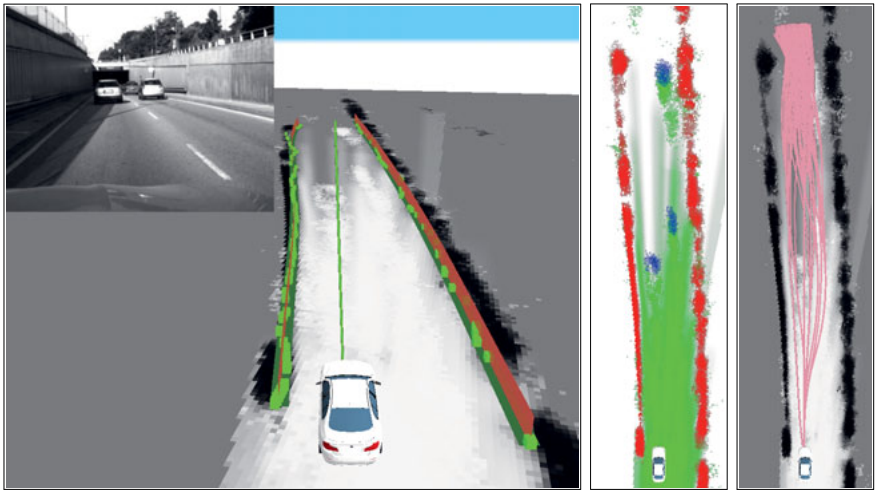
Figure 6.12: The driven trajectory of the scenario from Figure 6.10.

6.3 Qualitative Evaluation of the Road Course Estimation with GTAM

Finally, in this section, the road course estimation is applied on the static Bayesian map from the grid-based tracking and mapping. Whereas the scenarios in Section 6.2 are static, in this section, results from dynamic environments are given.

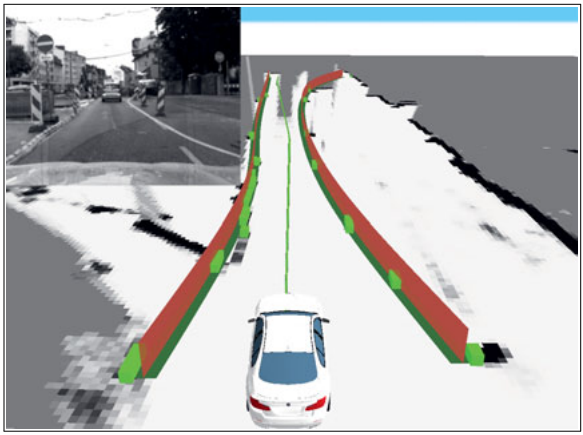
Figure 6.13a shows the used separator path, the extracted boundary cells, and the continuous semantic boundary on the static Bayesian map from GTAM. Figure 6.13b shows the static, the dynamic, and the free space evidences from the Dempster–Shafer map, which forms the basis to compute the static Bayesian map. The 3 dynamic objects, visible in blue, do not lead to occupied cells in the static map and therefore the path planner is able to find collision-free paths that satisfy the goal length, as shown in Figure 6.13c. The grids were generated from radar sensor data.

In Figure 6.14, a scenario is given, which shows a single-lane road construction site, where a vehicle is driving in front of the ego vehicle. Again, Figure 6.14a shows the separator path of the valid tracked road course, the boundary cells, and the semantic boundary, Figure 6.14b the DST map, and Figure 6.14c the clustered paths. Although multiple clusters, and thus principal moving directions, are found in this scenario, the one corresponding to the valid road course was successfully detected due to the validation and the tracking. In the previous scenario, with a radar sensor and with objects moving in radial direction, a static map can also be computed by simply filtering the data with the Doppler measurements. In this case, however, with a laser scanner, the vehicle in front of the ego vehicle leads to occupied cells, as depicted in Figure 6.14e. Thus, the valid road course is blocked in a standard occupancy grid and no collision-free paths can be found by the road course estimation. With the static map from GTAM, the dynamic objects are treated accordingly, as demonstrated in Figure 6.14d, and the valid road course is detected.

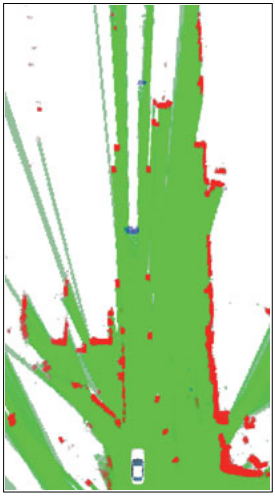


(a) Estimated valid road course. (b) DST map. (c) Paths.

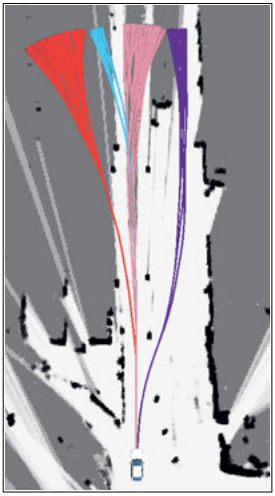
Figure 6.13: The road course estimation on the static Bayesian map from GTAM from radar sensor data.



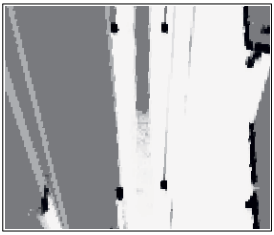
(a) Estimated valid road course.



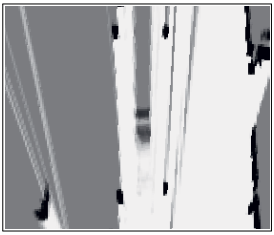
(b) DST map.



(c) Clustered paths on static map.



(d) Static map from GTAM.



(e) Standard occupancy grid.

Figure 6.14: The road course estimation on the static Bayesian map from GTAM from laser scanner data in a dynamic environment.

6.4 Summary

This chapter has given evaluations of the presented algorithms in addition to the results already given in the previous chapters. First, an analysis of the grid-based tracking and mapping has been shown. The convergence of the proposed filter has been analyzed and compared to the approach from [35]. The problem of particle deprivation, which leads to problems in the detection of continuous road boundaries, such as guard rails and tunnel walls, has been discussed and it has been shown how it is mitigated. The main parameters are evaluated, i.e., the maximum number of particles per cell, the ratio of new randomly created particles to fight the aforementioned particle deprivation, and the probability of static particles in the initial particle sampling probability. Plots and tables have been given, which demonstrate their effects. The classification between static and dynamic has also been compared to the one from [35] and an ROC curve has been provided. Finally, the estimated velocities have been evaluated and compared to a commercial tracking algorithm.

In addition to GTAM, the road course estimation has been evaluated. Receiver operating characteristic curves for each criterion of the road course validation, as well as for the combined validation, have been given. The accuracy and the true positive rate of the tracked primary road course have been shown for different planning horizons. The approach has also been compared to relying only on the predicted vehicle trajectory, instead of using motion planning. Moreover, the road course estimation system was integrated into a test vehicle and used to navigate through an unmapped single lane street scenario.

Finally, qualitative results from the road course estimation in dynamic environments with the use of the static Bayesian map from GTAM have been presented.

7 Conclusion

In this thesis new ideas and concepts for tracking and mapping, the extraction of principal moving directions, workspace cost evaluation for motion planning, and road course estimation have been presented and validated in experiments with a vehicle in street environments.

Grid-based Tracking and Mapping

A robust, real-time representation of the local static and dynamic environment is an essential component to enable autonomous navigation. Occupancy grid mapping is widely considered as the standard approach for creating a local map of the static obstacles. However, it assumes the world to be static, which leads to map errors in dynamic environments. If dynamic objects are mistaken for static ones, crucial navigation errors occur, especially for robots moving at high speeds, such as autonomous vehicles. Similarly important is a robust representation of the dynamic world. If different representations are used for the static and the dynamic environment, inconsistencies leading to errors are likely to occur.

Grid-based Tracking and Mapping (GTAM), a novel method that simultaneously estimates the static and the dynamic environment in a grid representation, has been presented. Particle filters are used to estimate a 2D continuous velocity distribution for every cell of the grid according to the input sensor data. New particles are sampled from a combination of a uniform distribution and a Dirac distribution at zero velocity, in order to be able to exactly represent the static environment. The estimated velocities are then used to derive continuous evidences for static and dynamic occupancy in a Dempster–Shafer model. The proposed FSD frame of discernment allows representing uncertainties in the static/dynamic classification, as well as evidence for unclassified occupancy. Moreover, a combination rule for temporal evidence filtering has been given, and it has been shown how the DST model is reduced to a standard Bayesian occupancy grid.

An obvious way to remove moving objects and their artifacts in occupancy grids is to use a standard object tracker and to filter the corresponding measurements. However, object tracking algorithms usually require a certain amount of time to detect and track new objects, during which the corresponding sensor measurements are not removed for the grid update. Moreover, the removal of scan points is a binary operation. Uncertainties in the estimation of the dynamic world cannot be incorporated. If a static object, such as a road boundary, is mistaken for a dynamic one, the whole structure is filtered out and can therefore not be detected by extractors anymore, such as road course estimation. In addition, there is typically no common low-level representation of the static and the dynamic environment, as standard object tracking algorithms use shape and model assumptions. The grid representation has become very popular, since it does not rely on such assumptions and allows representing arbitrary shapes; and with GTAM also the dynamic world.

Detection of Principal Moving Directions

Forks in the road and road junctions represent points of decision of the robot and impact navigation. Algorithms that extract information about the road, such as the road course and the road boundaries, often depend on knowing the topology of the road. Different to the topology given directly by the obstacles, the road topology may differ, such as when sparse boundary elements, like traffic cones, mark the road.

A novel approach to detect the principal moving directions has been presented. The problem is considered as path planning and path clustering. A local path planner has been developed that is able to efficiently plan a set of paths without goal poses to plan towards. Two different families of planning algorithms have been combined to efficiently sample a set of equal length paths that approximate the reachable set. The search is performed on a graph with a set of motion primitives that are adapted according to an expected drivable velocity on the given road. It is created and evaluated online. The A* algorithm is applied to yield a focused search towards the goal criterion and to assure that the optimal path, given the graph discretizations, is found. Using the same partially-explored graph, the node expansion strategy is adopted from the Rapidly Exploring Random Tree algorithm to achieve a uniform exploration of the search space.

The principal moving directions are then detected by clustering the paths. A novel clustering method has been presented that groups local trajectories according to the obstacles in the environment. Different to path homotopy, the trajectories do not need to have equal end states. A path equivalence definition has been provided. It is efficiently approximated by sampling endpoints of inter-trajectories, forming polygons, and by evaluating the polygons against the environment. The computational complexity is linear in the number of trajectories and thus scales well, compared to higher-order complexities of standard clustering algorithms, such as k-means, hierarchical clustering, or DBSCAN.

Since motion planning is used for the detection of the principal moving directions, constraints due to the shape and the possible motions under a given velocity are exploited. Therefore, the approach yields good results even with sparse road boundaries, as with a given road velocity, the probability of wrong estimations is greatly reduced. The number of principal moving directions is implicitly estimated and does not need to be given. The basis for planning paths is a map of the static world, which is achieved with GTAM even in dynamic environments. One of the key enablers for real-time path planning on workspace cost maps are the configuration space costs.

Configuration Space Costs

Checking robot configurations for collision and evaluating their costs takes the most computational resources of the majority of current motion planning algorithms. With configuration space obstacles, whole robot configurations are checked for collision using single look-ups. They thus, depending on the complexity of the planning problem, provide huge performance increases, compared to direct evaluations. They are pre-computed by incorporating the robot geometry into the obstacles.

Many planning problems use costs rather than only binary obstacle information, in order to find optimal paths or trajectories. Costs are often given due to the workspace.

Examples of workspace costs are distances to objects or traffic rules, such as driving in the right-most lane. Configuration space costs have been introduced and defined. They are a generalization of the configuration space obstacles and allow the evaluation of the cost of a complete configuration, which includes the check for collision, with a single look-up. Two methods for their efficient calculation have been presented. FAMOD, which is approximate in the output range but is able to handle arbitrary robot footprints and vHGW-360, which works with rectangular footprints but yields resolution-exact results.

Real robots have a shape and an extent, and they are not points in the workspace. Hence, similar to collision checking, if the costs are given in the form of a workspace cost map, the shape needs to be incorporated and it has been shown how. Costs are a generalization of collision, as the highest cost value is simply used to denote collision. During the design of an algorithm, there is often a trade-off between pre-computation and computation on-the-fly. During the performance tests, it was observed that it typically only requires the check of a four-digit number of configurations with the proposed methods on a graphics card to break-even with the time it takes for pre-computation and the same amount of look-ups. Since large planning problems easily require the check of millions of configurations, the discrete calculation of the configuration space costs will often provide huge speed-ups.

Road Course Estimation

The road course and the road boundaries represent the shape of the road ahead of the vehicle. They define where it is allowed to move and what the curvatures of the roadway are. The road boundaries are therefore important semantic information about the road layout. While in a-priori map-based navigation, they can be used for localization, in sensor-based navigation, they represent one form of guidance, as demonstrated in the experiments in the previous chapter.

A novel approach has been presented that is based on motion planning. The detected principal moving directions through the environment are used as road boundary separators, i.e., they divide the obstacles in the environment into a part that contains the left boundary and one that holds the right boundary. In these obstacle sets, the boundary elements are detected and a semantic continuous representation is obtained. Due to the principal moving directions, which are found by considering collision and velocity constraints, the approach also works with multiple present road courses, with arbitrary road shapes, and with sparse road boundary elements. Moreover, since it works on a grid representation, it is independent of specific features or particular sensors. The detected road courses and their boundaries are validated and tracked to achieve a high robustness.

Rather than directly fitting a road model into the sensor data or into an occupancy grid, the presented method works indirectly over motion planning. Sparse boundaries are particularly challenging, since the number of occupied grid cells corresponding to the road boundaries is low. Additionally, there can be a significant number of occupied cells corresponding to arbitrary static structures, leading to a failure in direct model fitting. Furthermore, estimating the number of road courses is a challenging problem by itself and influences the road model. The number of road courses is estimated according to the number of principal moving directions and refined by validation and tracking. Apart from model fitting, simply relying on the predicted vehicle trajectory has also been demonstrated

to lead to wrong separations or limited planning horizons. It only works well for particular road boundaries and shapes and assumes that there is always exactly one road course. The presented approach uses all of the above components: GTAM for a map of the environment, the principal moving directions as boundary separators, and the configuration space costs to efficiently calculate collisions and costs with workspace cost maps.

All of the presented methods have been implemented under consideration of real-time performance. Most parts have been developed as parallel GPU software. The algorithms have been evaluated with a full-sized autonomous vehicle platform. The input data has been recorded in real street environments, urban as well as highway scenarios, and autonomous navigation experiments have been conducted on a closed track.

This thesis concludes with a motivation for future research. Regarding environment modeling and perception for autonomous vehicle local navigation, there are still many challenges that have to be faced. The increasing computational performance alone will not automatically solve all the problems, although parallel hardware, such as graphics cards, allows more complex algorithms to be applied in real-time applications. Driven by the increase of interest in autonomous vehicles during the last years, this area of research is expected to continue to yield new and superior concepts and algorithms with an even broader range of possible impact.

A Appendix

This chapter presents additional material to the main chapters of this thesis. Section A.1 presents the research vehicles and the sensors used for the evaluations and Section A.2 presents the hardware for the computations. Section A.3 gives details about the coordinate systems and Section A.4 details about the smoother that was used to create the input for vehicle control. Finally, Section A.5 presents sample images for the evaluation data sets.

A.1 Prototype Vehicle and Sensor Setup

The test vehicles, with which the algorithms were evaluated, are a modified BMW 5 series car (F10) on the one hand, and a modified BMW i3 car (I01), on the other hand. The sensors that were used consist of a 4-layer laser scanners and imaging radar sensors. The test vehicle and the sensor positions are given in Figure A.1. Amongst other additional equipment, the vehicle is also equipped with a DGPS (Differential Global Positioning System) in order to precisely localize it in the world.

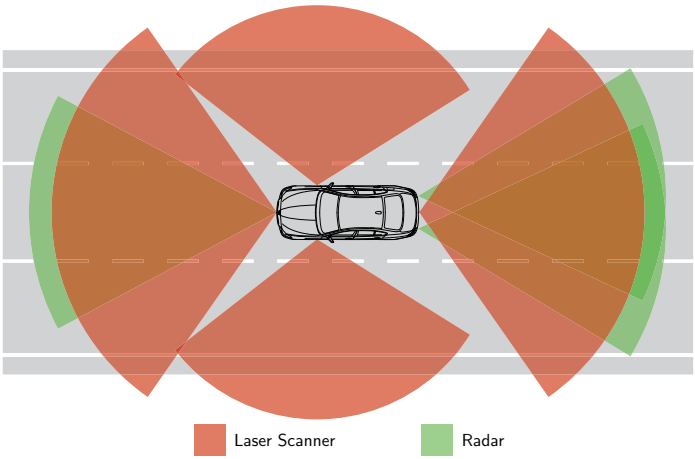
All sensors are fully integrated into the vehicle and not, as in other automated driving prototype cars, mounted on the body of the car. Many research automated driving vehicles use a 360 degree laser scanner mounted on the top. This has several advantages for the environment perception algorithms. There is only one sensor with full 360 degree field of view, the placement of the sensor allows observations over the top of other traffic participants and over a large distance, and additionally, the size and the weight of the sensor do not matter, since the space on top of the vehicle is not limited. Typically in such a setup, accurate, large, and expensive laser scanners are used. Although robust environment perception is much harder with sensors that are fully integrated in the vehicle, since they are typically low over the ground and have a limited field of view, from an aerodynamic and efficiency point of view, non-integrated sensors are inferior and facilities such as car washers cannot be used.

The used 4-layer laser scanner has a horizontal field of view (FOV) of 110 degrees and a vertical FOV of 3.2 degrees. It runs at 12.5 Hz with an angular resolution between 0.125 and 0.5 degrees decreasing from the sensor axis to the outside. It aslo has another operating mode at 25 Hz and a constant 0.25 degree resolution. The maximum detection range is at around 200 m. In practice, however, the maximum detection range, especially for continuous road boundaries parallel to the sensor axis, is quite below.

The radar sensor has a horizontal field of view of 18 degrees with a 1 degree angular resolution and a maximum range of 200 m. It runs at 15 Hz. The sensor is similar to a series production radar used in driver assistance systems, such as the adaptive cruise control (ACC).

(a) A modified BMW 5 Series car.

(b) A modified BMW i3 car.



(c) Used sensors in the work of this thesis.

Figure A.1: The test vehicles and the sensors used in this thesis. Images courtesy of BMW Group. Most results were obtained with the two sensors to the front.

A.2 Hardware and Software Computing Platform

In the following, the hardware used for the computations is given. All CPU programs were developed in C++ and compiled with Microsoft Visual Studio 2010 under 64-bit Microsoft Windows 7. The GPU was programmed with Nvidia CUDA 5.0. All timing and performance measurements throughout this thesis were done on the same machine. Standard PC hardware was used. It is given in the following table:

Table A.1: Computing hardware.

CPU	Intel Core i7-3770 @ 3500 MHz, 4 cores, 8 threads, 8 MB cache
GPU	Nvidia GeForce GTX 660 Ti, 1344 CUDA cores, 2048 MB memory, PCI Express 3.0
Primary memory	8192 MB

A.3 Local Grid Mapping

In this section, the different coordinate systems and the layout of the grid structure are described.

World Coordinate System (WCS) The world coordinate system represents a fixed position in the world and does not move over time. Note that world in this case does not necessarily equal the physical world, as described below. Initially, the WCS equals the vehicle coordinate system.

Vehicle Coordinate System (VCS) The vehicle coordinate system is fixed at a certain position on the vehicle, in particular at the center of the rear axis. The x axis points into the orientation of the vehicle and the y axis to the left.

Sensor Coordinate System (SCS) The sensor coordinate system SCS^k denotes the origin of a particular sensor k. The x axis points into the direction of the sensor axis and the y axis to the left. The origin of the SCS is given in the vehicle coordinate system, since it stays constant over time.

Map Grid Coordinate System (MGCS) The map grid coordinate system denotes the filtered grid map. Since grids are essentially images, the conventional image coordinate system is used, i.e., the origin is at the top left corner of the grid, the x axis points to the right (over the columns of the matrix) and the y axis point downwards (over the rows). Positive angles, in this work, always go from the x axis to the y axis and are therefore clockwise in the grid coordinate systems. Note that in different implementations, one can choose to set the origin at the bottom left corner and setting the axis equal to the WCS. This way, the transformation between the WCS and the MGCS is easier, since the y does not have to be inverted and the angles match. Since the algorithms are implemented on a graphics card, the conventional image coordinate system was nevertheless preferred. In any case, it is noted that the grid coordinate

system and the way how the grid actually resides in memory needs to be consistent. In particular, the first array element is the origin of the grid coordinate system.

Scan Grid Coordinate System (SGCS) The scan grid coordinate system represents a single scan of one particular sensor. The coordinate system is similar to the MGCS. Different coordinate systems are used for scans and the map, since the map can cover a larger region than a single scan. Moreover, it is beneficial that the layout of the SGCS is optimized to efficiently cover the field of view of the particular sensor, e.g., for a sensor that observes an area to the right of the robot, it is advantageous that the VCS is to the left of the center of the scan grid.

In order to filter over time while the robot is moving, the map grid from the previous time step and the current scan grid must be transformed so that the same locations of the world are combined. Here, an approach similar to [71] and [168] is used. If the vehicle is set to a fixed position with fixed orientation in the grid, every update needs to rotate the previous map according to the difference of the vehicle orientations at t and $t+1$ due to the movement. This requires resampling, which leads to discretization and aliasing errors [168]. Therefore, the position of the vehicle, or more precisely the origin of the vehicle coordinate system, is not fixed but rotates on a circle with radius r_m, centered at the grid center, with the current vehicle orientation, i.e., the x axis of the VCS, always pointing towards this center. This way, two grids can be aligned by a translation only. Unless only sensors to the front are used and the vehicle is primarily driving forwards, such as in this work, the radius r_m will often be 0, especially if the robot is equipped with sensors that perceive 360 degrees of the environment.

The new origin of the local grid of size $n_{\text{cells}} \times n_{\text{cells}}$ in the world coordinate system at time t

$$o_t^{\text{MGCS}} = \text{round}\Big(o_t^{\text{VCS}} + R_\theta \begin{pmatrix} r_m & 0 \end{pmatrix}^{\text{T}} + \begin{pmatrix} -n_{\text{cells}}/2 & n_{\text{cells}}/2 \end{pmatrix}^{\text{T}}\Big) \tag{A.1}$$

is calculated based on the current location o_t^{VCS} and the current orientation θ of the vehicle coordinate system, which in turn is computed using odometry information. Note that although discrete cells are used, the position of the vehicle is stored using continuous values. Note also, that here it is assumed that the units used throughout all coordinate systems are cells to ease notation.

Similarly, the coordinate system of the scan grid is set such that the current orientation of the sensor axis yielding that particular scan, i.e., the x axis of the SCS, points towards the scan grid center rotating on a circle with some other radius r_s. While for the map grid, r_m will often be 0 as mentioned above, for one particular sensor scan, it is beneficial that the field of view of the sensor has maximum overlap with the scan grid. The available memory is then used more efficiently. Figure A.2 shows a diagram of the relations between the coordinate systems. Note that the map grid is only shown for one particular time step t, while two scan grids from two different time steps t and $t+1$ are shown.

The grid discretization can be thought of as an infinite rasterization of the world that is fixed through the initial placement of the world coordinate system WCS and stays constant over time. The local grid maps at time instance t, i.e., MGCS_t and SGCS_t, are an exact integer window of this grid world. Since the origins of the grid coordinate systems are

stored in the WCS, the grids are easily aligned correctly without the need of resampling, as the cells always perfectly align.

The world coordinate system does not represent absolute coordinates, as in simultaneous localization and mapping, but a *drift space* using odometry. The odometry information needed to obtain VCS_t is calculated in a separate central component, so that it is equal for all modules, such as mapping or path planning. The drift space has the advantage of being smooth and continuous over time. Moreover, absolute coordinates are irrelevant for this work, since the grid is only used online and is not stored. If an offline, large map is required, then the local grids can be aligned with SLAM methods [157]. Note that compared to other mobile robots, odometry in full-sized vehicles is locally very accurate under moderate dynamics. Slipping of wheels almost never occurs due to stability systems and accelerations are typically modest.

A.4 Path Smoothing

This section describes the smoother, which was used to generate the trajectory for the autonomous navigation experiment in Section 6.2.4.

The smoothing consists of two steps. Let τ_p^n denote the nodes of the primary path, which the vehicle follows. First, the path nodes τ_p^n are optimized with a gradient descent similar to [49], but with different optimization criteria. Subject to the optimization is

$$\sum_j w_j f_j \rightarrow \min \tag{A.2}$$

consisting of the sum of the individual weighted optimization terms

$$\begin{aligned} f_1 &= \sum_i^m ||\tau_{p,i}^n - \tau_{s,i}^n||, \qquad & f_2 &= \sum_i^m ||\Delta\tau_{s,i+1}^n - \Delta\tau_{s,i}^n||, \\ f_3 &= \sum_i^m ||\tau_{s,i}^n - \gamma_i||, & f_4 &= \sum_i^m |\Delta\psi_{i+1} - \Delta\psi_i|, \\ f_5 &= \sum_i^m ||\tau_{s,i}^n - \tau_{s_{t-1},i}^n||, \end{aligned} \tag{A.3}$$

where $\tau_{p,i}^n$ and $\tau_{s,i}^n$ denote node i of the original path and the current smoothed path respectively, and τ_s^n is initialized with τ_p^n. Moreover, ψ_i denotes the angle between $\tau_{s,i}^n$ and $\tau_{s,i+1}^n$, γ_i denotes the closest point from $\tau_{s,i}^n$ on the generalized Voronoi diagram of the semantic road boundary, and $\tau_{s_{t-1}}^n$ denotes the smoothed path from $t-1$.

After path node optimization, a penalized regression spline [135] is fit into τ_s^n. It allows sampling the final path as dense as required by the underlying controller [169] as well as assuring the continuity requirements.

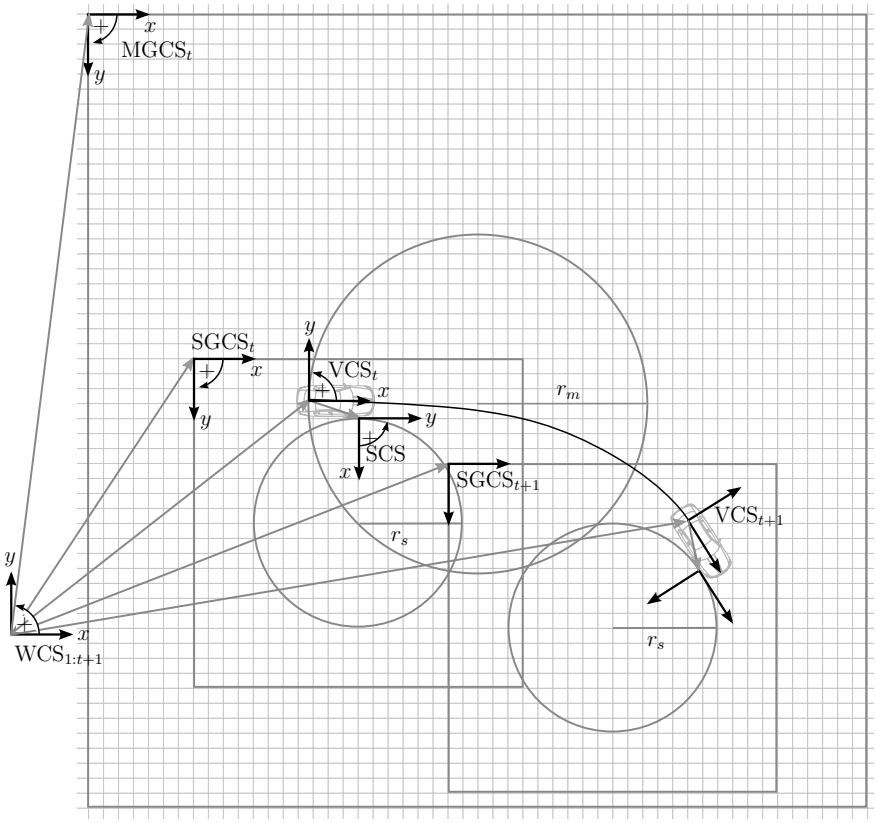


Figure A.2: Relation between the world coordinate system (WCS), the vehicle coordinate system (VCS), the sensor coordinate system (SCS), the map grid coordinate system (MGCS), and the scan grid coordinate system (SGCS).

A.5 Data Sets

Two different ground truth data sets were created by manually labeling. One data set was used for the evaluation of the grid-based tracking and mapping and one for the evaluation of the road course estimation.

A.5.1 Grid-based Tracking and Mapping

The data set for the evaluation of GTAM in Section 6.1.2 and Section 6.1.3 consists of around 5 minutes of real street scenarios. It includes short sequences of highways as well as urban areas, different traffic participants, such as vehicles, pedestrians, and bicycles, and different road boundaries, such as parked vehicles, guardrails, and tunnel walls. Figure A.3 shows sample camera images. The dynamic objects were labeled manually, so that in every frame all true dynamic objects and their positions are available. The cycle time was at 80 ms. With the labeled objects, masks were then generated that define whether each occupied cell of every scan grid corresponds to a dynamic or a static object. According to these classified scan grids, the evaluation was performed.

A.5.2 Road Course Estimation

The data set for the evaluation of the RCE also consists of real street scenarios, as well as of simulated road construction sites marked with traffic cones. The duration totals around 7.5 minutes. Similar to the data set from the previous section, this one also consists of short sequences of various streets. It contains highways and urban areas and a variety of different road boundaries, such as guardrails, walls, bushes, traffic cones and other traffic markers, and parked vehicles. Figure A.4 shows sample camera images. Only static scenes are used in this data set to focus on the road course estimation alone. Similar to the dynamic objects from the previous section, in this data set, the semantic continuous road boundaries were labeled manually with splines. Again, the cycle was at 80 ms and at every frame all true valid road boundaries are available.

Figure A.3: Camera images from the data set used for the evaluation of the grid-based tracking and mapping.

Figure A.4: Camera images from the data set used for the evaluation of the road course estimation.

Own Publications

[1] M. Aeberhard, S. Rauch, M. Bahram, G. Tanzmeister, J. Thomas, Y. Pilat, F. Homm, W. Huber, and N. Kaempchen, "Experience, results and lessons learned from automated driving on germany's highways," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 42–57, Spring 2015.

[2] M. Friedl, A. Hupka, and G. Tanzmeister, "Vollautomatisiertes Valet Parking: Funktions- und Planungsarchitektur," in *10. Workshop Fahrerassistenzsysteme*. Walting: Uni-DAS e. V., 2015.

[3] M. Knecht, G. Tanzmeister, C. Traxler, and M. Wimmer, "Interactive BRDF estimation for mixed-reality applications," *Journal of WSCG*, vol. 20, no. 1, pp. 47–56, Jun. 2012.

[4] A. Lawitzky, D. Althoff, C. Passenberg, G. Tanzmeister, D. Wollherr, and M. Buss, "Interactive scene prediction for automotive applications," in *Proc. of the IEEE Intelligent Vehicles Symposium*, Jun. 2013, pp. 1028–1033.

[5] G. Tanzmeister, M. Friedl, A. Lawitzky, D. Wollherr, and M. Buss, "Road course estimation in unknown, structured environments," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2013, pp. 630–635.

[6] G. Tanzmeister, M. Friedl, D. Wollherr, and M. Buss, "Path planning on grid maps with unknown goal poses," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2013, pp. 430–435.

[7] G. Tanzmeister, M. Friedl, D. Wollherr, and M. Buss, "Efficient evaluation of collisions and costs on grid maps for autonomous vehicle motion planning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2249–2260, Oct. 2014.

[8] G. Tanzmeister, J. Thomas, D. Wollherr, and M. Buss, "Grid-based mapping and tracking in dynamic environments using a uniform evidential environment representation," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2014, pp. 6090–6095.

[9] G. Tanzmeister, D. Wollherr, and M. Buss, "Environment-based trajectory clustering to extract principal directions for autonomous vehicles," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 667–673.

[10] G. Tanzmeister, D. Wollherr, and M. Buss, "Grid-based multi-road-course estimation using motion planning," *IEEE Transactions on Vehicular Technology*, pp. 1–12, Apr. 2015, IEEE Early Access Articles, DOI: 10.1109/TVT.2015.2420752.

Bibliography

[11] M. Aeberhard, S. Schlichtharle, N. Kaempchen, and T. Bertram, "Track-to-track fusion with asynchronous sensors using information matrix fusion for surround environment perception," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1717–1726, Dec. 2012.

[12] D. Althoff, J. Kuffner, D. Wollherr, and M. Buss, "Safety assessment of robot trajectories for navigation in uncertain and dynamic environments," *Autonomous Robots*, vol. 32, no. 3, pp. 285–302, Apr. 2012.

[13] M. Althoff, "Reachability analysis and its application to the safety assessment of autonomous cars," Ph.D. dissertation, Institute of Automatic Control Engineering (LSR), Technische Universität München, 2010.

[14] M. Ardelt, C. Coester, and N. Kaempchen, "Highly automated driving on freeways in real traffic using a probabilistic framework," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1576–1585, Dec. 2012.

[15] M. Ardelt, "Hybrid control strategies for advanced safety- and driver assistance systems," Ph.D. dissertation, Institute of Automatic Control Engineering (LSR), Technische Universität München, 2012.

[16] S. Atev, G. Miller, and N. Papanikolopoulos, "Clustering of vehicle trajectories," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 647–657, Sep. 2010.

[17] H. Badino, U. Franke, and D. Pfeiffer, "The stixel world - a compact medium level representation of the 3d-world," in *Proc. of the German Association for Pattern Recognition (DAGM)*, 2009.

[18] M. Bahram, A. Wolf, M. Aeberhard, and D. Wollherr, "A prediction-based reactive driving strategy for highly automated driving function on freeways," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2014, pp. 400–406.

[19] Q. Baig and O. Aycard, "Improving moving objects tracking using road model for laser data," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2012, pp. 790–795.

[20] S. Bayerl and H.-J. Wuensche, "Detection and tracking of rural crossroads combining vision and lidar measurements," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2014, pp. 1274–1279.

[21] K. Bekris, B. Chen, A. Ladd, E. Plaku, and L. Kavraki, "Multiple query probabilistic roadmap planning using single query planning primitives," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003, pp. 656–661.

[22] S. Bhattacharya, V. Kumar, and M. Likhachev, "Search-based path planning with homotopy class constraints," in *Proc. of the AAAI Conference on Artificial Intelligence*, 2010, pp. 1230–1237.

[23] BMW Group PressClub, "BMW at the consumer electronics show (CES) 2015 in Las Vegas," https://www.press.bmwgroup.com/global/pressDetail.html?title=bmw-at-the-consumer-electronics-show-ces-2015-in-las-vegas&outputChannelId=6&id=T0199262EN&left_menu_item=node__6728#avMedia, 2015.

[24] M. Bouzouraa and U. Hofmann, "Fusion of occupancy grid mapping and model based object tracking for driver assistance systems using laser and radar sensors," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2010, pp. 294–300.

[25] S. Brechtel, T. Gindele, and R. Dillmann, "Recursive importance sampling for efficient grid-based occupancy filtering in dynamic environments," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2010, pp. 3932–3938.

[26] A. Broggi, P. Cerri, M. Felisa, M. C. Laghi, L. Mazzei, and P. P. Porta, "The VisLab intercontinental autonomous challenge: an extensive test for a platoon of intelligent vehicles," *International Journal of Vehicle Autonomous Systems*, vol. 10, no. 3, pp. 147–164, 2012.

[27] D. Buzan, S. Sclaroff, and G. Kollios, "Extraction and clustering of motion trajectories in video," in *Proc. of the International Conference on Pattern Recognition*, vol. 2, 2004, pp. 521–524.

[28] P. Chen, H. Zhao, C. Tao, and H. Sang, "Block-run-based connected component labelling algorithm for GPGPU using shared memory," *Electronics Letters*, vol. 47, no. 24, pp. 1309–1311, Nov. 2011.

[29] S. Choi, J.-Y. Lee, and W. Yu, "Fast any-angle path planning on grid maps with non-collision pruning," in *Proc. of the IEEE International Conference on Robotics and Biomimetics*, 2010, pp. 1051–1056.

[30] S. Choi and W. Yu, "Any-angle path planning on non-uniform costmaps," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2011, pp. 5615–5621.

[31] B. Clarke, S. Worrall, G. Brooker, and E. Nebot, "Sensor modelling for radar-based occupancy mapping," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 3047–3054.

[32] C. Coué, C. Pradalier, C. Laugier, T. Fraichard, and P. Bessiere, "Bayesian occupancy filtering for multitarget tracking: an automotive application," *International Journal of Robotics Research*, vol. 25, no. 1, pp. 19–30, Jan. 2006.

[33] J. Crisman and C. Thorpe, "SCARF: a color vision system that tracks roads and intersections," *IEEE Journal of Robotics and Automation*, vol. 9, no. 1, pp. 49–58, Feb. 1993.

[34] B. Curto, V. Moreno, and F. Blanco, "A general method for C-space evaluation and its application to articulated robots," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 24–31, Feb. 2002.

[35] R. Danescu, F. Oniga, and S. Nedevschi, "Modeling and tracking the driving environment with a particle-based occupancy grid," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1331–1342, Dec. 2011.

[36] R. Danescu, C. Pantilie, F. Oniga, and S. Nedevschi, "Particle grid tracking system stereovision based obstacle perception in driving environments," *IEEE Intelligent Transportation Systems Magazine*, vol. 4, no. 1, pp. 6–20, 2012.

[37] M. Darms, M. Komar, and S. Lueke, "Map based road boundary estimation," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2010, pp. 609–614.

[38] E. R. Davies, *Machine Vision: Theory, Algorithms, Practicalities*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann, 2005.

[39] Defense Advanced Research Projects Agency (DARPA), "Grand challenge 04," http://archive.darpa.mil/grandchallenge04/, 2004.

[40] Defense Advanced Research Projects Agency (DARPA), "Grand challenge 05," http://archive.darpa.mil/grandchallenge05/, 2005.

[41] Defense Advanced Research Projects Agency (DARPA), "Grand challenge 07 (urban challenge)," http://archive.darpa.mil/grandchallenge/, 2007.

[42] A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping," *The Annals of Mathematical Statistics*, vol. 38, no. 2, pp. 325–339, Apr. 1967.

[43] A. P. Dempster, "A generalization of Bayesian inference," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 30, no. 2, pp. 205–247, 1968.

[44] E. D. Dickmanns, *Dynamic Vision for Perception and Control of Motion*. London, UK: Springer-Verlag, 2007.

[45] E. D. Dickmanns and A. Zapp, "Autonomous high speed road vehicle guidance by computer vision," in *Proc. of the International Federation of Automatic Control World Congress*, 1987, pp. 221–226.

[46] T. A. Dingus, S. G. Klauer, V. L. Neale, A. Petersen, S. E. Lee, J. Sudweeks, M. A. Perez, J. Hankey, D. Ramsey, S. Gupta, C. Bucher, Z. R. Doerzaph, J. Jermeland, and R. Knipling, "The 100-car naturalistic driving study, phase 2 – results of the 100-car field experiment," Virginia Tech Transportation Institute and National Highway Traffic Safety Administration, Tech. Rep. DOT HS 810 593, Apr. 2006, http://www.nhtsa.gov/DOT/NHTSA/NRD/Multimedia/PDFs/Crash%20Avoidance/Driver%20Distraction/100CarMain.pdf.

[47] P. Dokládal and E. Dokládalová, "Computationally efficient, one-pass algorithm for morphological filters," *Journal of Visual Communication and Image Representation*, vol. 22, no. 5, pp. 411–420, Jul. 2011.

[48] D. Dolgov and S. Thrun, "Detection of principle directions in unknown environments for autonomous navigation," in *Proc. of Robotics: Science and Systems*, 2008.

[49] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, Apr. 2010.

[50] L. Domanski, P. Vallotton, and D. Wang, "Parallel van Herk/Gil–Werman image morphology on GPUs using CUDA," NVIDIA GPU Computing Poster Showcase, http://www.nvidia.com/content/GTC/posters/14_Domanski_Parallel_vanHerk.pdf, 2009.

[51] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Hoboken, NJ, USA: John Wiley & Sons, 2001.

[52] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, Jun. 1989.

[53] Ernst & Young, "Deploying autonomous vehicles: commercial considerations and urban mobility scenarios," http://www.ey.com/Publication/vwLUAssets/EY-Deploying-autonomous-vehicles-30May14/$FILE/EY-Deploying-autonomous-vehicles-30May14.pdf, Jun. 2014.

[54] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. of the International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 226–231.

[55] Eureka, "Prometheus project," http://www.eurekanetwork.org/project/-/id/45, 1987–1994.

[56] D. Ferguson and A. Stentz, "Anytime RRTs," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 5369–5375.

[57] D. Ferguson and A. Stentz, "Field D*: An interpolation-based path planner and replanner," in *Proc. of the International Symposium on Robotics Research*, 2005, pp. 1926–1931.

[58] T. Gasser, C. Arzt, M. Ayoubi, A. Bartels, J. Eier, F. Flemisch, D. Häcker, T. Hesse, W. Huber, C. Lotz, M. Maurer, S. Ruth-Schumacher, J. Schwarz, and W. Vogt, "Rechtsfolgen zunehmender fahrzeugautomatisierung," Bundesanstalt für Straßenwesen (BASt), Tech. Rep., Jan. 2012, available online: http://www.bast.de/DE/Publikationen/Foko/Downloads/2012-11.pdf?__blob=publicationFile.

[59] P. K. Ghosh and K. Kumar, "Support function representation of convex bodies, its application in geometric computing, and some related representations," *Computer Vision and Image Understanding*, vol. 72, no. 3, pp. 379–403, Dec. 1998.

[60] J. Gil and M. Werman, "Computing 2-D min, median, and max filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 5, pp. 504–507, May 1993.

[61] T. Gindele, S. Brechtel, J. Schröder, and R. Dillmann, "Bayesian occupancy grid filter for dynamic environments using prior map knowledge," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2009, pp. 669–676.

[62] Google Inc., "The google self-driving car project," https://plus.google.com/+GoogleSelfDrivingCars/, 2014.

[63] D. Grigoriev and A. Slissenko, "Polytime algorithm for the shortest path in a homotopy class amidst semi-algebraic obstacles in the plane," in *Proc. of the International Symposium on Symbolic and Algebraic Computation*, 1998, pp. 17–24.

[64] L. Guibas, L. Ramshaw, and J. Stolfi, "A kinetic framework for computational geometry," in *Proc. of the IEEE Annual Symposium on Foundations of Computer Science*, 1983, pp. 100–111.

[65] B. Gutjahr and M. Werling, "Automatic collision avoidance during parking and maneuvering - an optimal control approach," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2014, pp. 636–641.

[66] J. Han, D. Kim, M. Lee, and M. Sunwoo, "Enhanced road boundary and obstacle detection using a downward-looking lidar sensor," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 3, pp. 971–985, Mar. 2012.

[67] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. SMC-4, no. 2, pp. 100–107, Jul. 1968.

[68] Y. He, H. Wang, and B. Zhang, "Color-based road detection in urban traffic scenes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 309–318, Dec. 2004.

[69] D. Hearn and M. P. Baker, *Computer graphics with OpenGL*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2004.

[70] J. Hershberger and J. Snoeyink, "Computing minimum length paths of a given homotopy class," *Computational Geometry: Theory and Applications*, vol. 4, pp. 63–97, Jun. 1994.

[71] 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 *Proc. of the IEEE Intelligent Vehicles Symposium*, 2010, pp. 1006–1013.

[72] F. Homm, "Fahrzeugeigenlokalisation im Kontext hochautomatisierter Fahrfunktionen," Ph.D. dissertation, Lehrstuhl für Echtzeitsysteme und Robotik, Technische Universität München, 2012.

[73] A. Hupka, "Entwicklung einer situationsbeeinflussten Planungsarchitektur für das hochautomatisierte Parken in Parkhäusern," Master's thesis, Institut für Verkehrstelematik, Technische Universität Dresden, 2015.

[74] L. Jaillet, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 635–646, Aug. 2010.

[75] P. Jeong and S. Nedevschi, "Efficient and robust classification method using combined feature vector for lane detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 4, pp. 528–537, Apr. 2005.

[76] T. Jochem, D. Pomerleau, B. Kumar, and J. Armstrong, "PANS: a portable navigation platform," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 1995, pp. 107–112.

[77] A. Jøsang, J. Diaz, and M. Rifqi, "Cumulative and averaging fusion of beliefs," *Information Fusion*, vol. 11, no. 2, pp. 192–200, Apr. 2010.

[78] A. Jøsang and S. Pope, "Dempster's rule as seen by little colored balls," *Computational Intelligence*, vol. 28, no. 4, pp. 453–474, Nov. 2012.

[79] N. Kaempchen, M. Aeberhard, M. Ardelt, and S. Rauch, "Technologies for highly automated driving on highways," *Automobiltechnische Zeitschrift*, vol. 114, pp. 34–38, Jun. 2012.

[80] N. Kaempchen, "Feature-level fusion of laser scanner and video data for advanced driver assistance systems," Ph.D. dissertation, Fakultät für Ingenieurwissenschaften und Informatik, Universität Ulm, 2007.

[81] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schröder, M. Thuy, M. Goebl, F. v. Hundelshausen, O. Pink, C. Frese, and C. Stiller, "Team AnnieWAY's autonomous system for the 2007 DARPA urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 615–639, Sep. 2008.

[82] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *Proc. of the IEEE Conference on Decision and Control*, 2010, pp. 7681–7687.

[83] L. Kavraki, "Computation of configuration-space obstacles using the fast fourier transform," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 408–413, Jun. 1995.

[84] Z. Kim, "Robust lane detection and tracking in challenging scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 16–26, Mar. 2008.

[85] J. King and M. Likhachev, "Efficient cost computation in cost map planning for non-circular robots," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 3924–3930.

[86] K. Klasing, D. Wollherr, and M. Buss, "Cell-based probabilistic roadmaps (CPRM) for efficient path planning in large environments," in *Proc. of the International Conference on Advanced Robotics*, 2007, pp. 1075–1080.

[87] K. Kluge and C. Thorpe, "The YARF system for vision-based road following," *Mathematical and Computer Modelling*, vol. 22, no. 4-7, pp. 213–233, Aug. 1995.

[88] R. A. Knepper, S. S. Srinivasa, and M. T. Mason, "Toward a deeper understanding of motion alternatives via an equivalence relation on local paths," *International Journal of Robotics Research*, vol. 31, no. 2, pp. 167–186, Feb. 2012.

[89] K. R. S. Kodagoda, S. S. Ge, W. S. Wijesoma, and A. P. Balasuriya, "IMMPDAF approach for road-boundary tracking," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 2, pp. 478–486, Mar. 2007.

[90] M. Konrad, D. Nuss, and K. Dietmayer, "Localization in digital maps for road course estimation using grid maps," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2012, pp. 87–92.

[91] M. Konrad, M. Szczot, and K. Dietmayer, "Road course estimation in occupancy grids," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2010, pp. 412–417.

[92] T. Kühnl, F. Kummert, and J. Fritsch, "Monocular road segmentation using slow feature analysis," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2011, pp. 800–806.

[93] J.-C. Latombe, *Robot Motion Planning*. Boston, MA, USA: Kluwer Academic, 1991.

[94] J.-P. Laumond, *Robot Motion Planning and Control*. Berlin, Germany: Springer-Verlag, 1998.

[95] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept., Iowa State University, Ames, IA, USA, Tech. Rep. TR 98-11, 1998.

[96] S. M. LaValle, *Planning Algorithms*. Cambridge, UK: Cambridge University Press, 2006.

[97] A. Lawitzky, D. Wollherr, and M. Buss, "Maneuver-based risk assessment for high-speed automotive scenarios," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1186–1191.

[98] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: A partition-and-group framework," in *Proc. of the ACM SIGMOD International Conference on Management of Data*, 2007, pp. 593–604.

[99] D. Lemire, "Faster retrieval with a two-pass dynamic-time-warping lower bound," *Pattern Recognition*, vol. 42, no. 9, pp. 2169–2180, Sep. 2009.

[100] Q. Li, N. Zheng, and H. Cheng, "Springrobot: a prototype autonomous vehicle and its algorithms for lane detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 300–308, Dec. 2004.

[101] G. Lidoris, D. Wollherr, and M. Buss, "Bayesian state estimation and behavior selection for autonomous robotic exploration in dynamic environments," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 1299–1306.

[102] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, Aug. 2009.

[103] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic A*: An anytime, replanning algorithm," in *Proc. of the International Conference on Automated Planning and Scheduling*, 2005, pp. 262–271.

[104] M. Likhachev, G. J. Gordon, and S. Thrun, "ARA*: Anytime A* search with provable bounds on sub-optimality," in *Advances in Neural Information Processing Systems*. MIT Press, 2003.

[105] T. Luettel, M. Himmelsbach, and H.-J. Wuensche, "Autonomous ground vehicles—concepts and a path to the future," *Proc. of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1831–1839, May 2012.

[106] M. Manz, M. Himmelsbach, T. Luettel, and H.-J. Wuensche, "Detection and tracking of road networks in rural terrain by fusing vision and lidar," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 4562–4568.

[107] M. Manz, F. von Hundelshausen, and H.-J. Wuensche, "A hybrid estimation approach for autonomous dirt road following using multiple clothoid segments," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2010, pp. 2410–2415.

[108] M. Manz, "Modellbasierte visuelle Wahrnehmung zur autonomen Fahrzeugführung," Ph.D. dissertation, Institut für Technik Autonomer Systeme, Universität der Bundeswehr München, 2013.

[109] P. Maragos and R. Schafer, "Morphological filters–part I: Their set-theoretic analysis and relations to linear shift-invariant filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 8, pp. 1153–1169, Aug. 1987.

[110] McKinsey & Company, "Connected car, automotive value chain unbound," http://www.mckinsey.com/client_service/automotive_and_assembly/latest_thinking/publication_request_form, Sep. 2014.

[111] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proc. of the AAAI Conference on Artificial Intelligence*, 2002, pp. 593–598.

[112] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: The Stanford entry in the urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, Sep. 2008.

[113] J. Moras, V. Cherfaoui, and P. Bonnifait, "Credibilist occupancy grids for vehicle perception in dynamic environments," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2011, pp. 84–89.

[114] J. Munkres, *Topology*. Upper Saddle River, NJ, USA: Prentice Hall, 2000.

[115] C. K. Murphy, "Combining belief functions when evidence conflicts," *Decision Support Systems*, vol. 29, pp. 1–9, Jul. 2000.

[116] A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," in *Proc. of the AAAI Conference on Artificial Intelligence*, 2007, pp. 1177–1183.

[117] National Highway Traffic Safety Administration (NHTSA), "Preliminary statement of policy concerning automated vehicles," http://www.nhtsa.gov/staticfiles/rulemaking/pdf/Automated_Vehicles_Policy.pdf, 2013.

[118] T.-N. Nguyen, M.-M. Meinecke, M. Tornow, and B. Michaelis, "Optimized grid-based environment perception in advanced driver assistance systems," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2009, pp. 425–430.

[119] T.-N. Nguyen, B. Michaelis, A. Al-Hamadi, M. Tornow, and M.-M. Meinecke, "Stereo-camera-based urban environment perception using occupancy grid and object tracking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 154–165, Mar. 2012.

[120] NVIDIA Corporation, "CUDA C best practices guide," http://docs.nvidia.com/cuda/pdf/CUDA_C_Best_Practices_Guide.pdf, Jul. 2014.

[121] NVIDIA Corporation, "CUDA Fast Fourier Transform library (cuFFT)," https://developer.nvidia.com/cufft, Jul. 2014.

[122] NVIDIA Corporation, "CUDA parallel computing platform," http://www.nvidia.com/object/cuda_home_new.html, Jul. 2014.

[123] NVIDIA Corporation, "NVIDIA Performance Primitives (NPP)," https://developer.nvidia.com/npp, Jul. 2014.

[124] D. Nyga, M. Tenorth, and M. Beetz, "How-models of human reaching movements in the context of everyday manipulation activities," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 6221–6226.

[125] D. Pagac, E. Nebot, and H. Durrant-Whyte, "An evidential approach to map-building for autonomous vehicles," *IEEE Transactions on Robotics*, vol. 14, no. 4, pp. 623–629, Aug. 1998.

[126] K. Peterson, J. Ziglar, and P. Rybski, "Fast feature detection and stochastic parameter estimation of road shape using multiple lidar," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2008, pp. 612–619.

[127] S. Pietzsch, T.-D. Vu, J. Burlet, O. Aycard, T. Hackbarth, N. Appenrodt, J. Dickmann, and B. Radig, "Results of a precrash application based on laser scanner and short-range radars," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 4, pp. 584–593, Dec. 2009.

[128] M. Pivtoraiko, "Differentially constrained motion planning with state lattice motion primitives," Ph.D. dissertation, The Robotics Institute, Carnegie Mellon University, 2012.

[129] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, Mar. 2009.

[130] E. Plaku, L. E. Kavraki, and M. Vardi, "Discrete search leading continuous exploration for kinodynamic motion planning," in *Proc. of Robotics: Science and Systems*, 2007.

[131] R. Rojas et al., "AutoNOMOS Labs FU Berlin," http://autonomos.inf.fu-berlin.de/, 2014.

[132] S. Rauch, A. Savkin, T. Schaller, and P. Hecker, "Hochgenaue Fahrzeugeigenlokalisierung und kollektives Erlernen hochgenauer digitaler Karten," in *Proc. of the AAET Automatisierungssysteme, Assistenzsysteme und eingebettete Systeme für Transportmittel*, 2012.

[133] A. Richardson and E. Olson, "Iterative path optimization for practical robot planning," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 3881–3886.

[134] Roland Berger Strategy Consultants, "Think act autonomous driving," http://www.rolandberger.ch/media/pdf/Roland_Berger_TABAutonomousDrivingfinal20141211.pdf, Nov. 2014.

[135] D. Ruppert, M. P. Wand, and R. J. Carroll, *Semiparametric Regression*. Cambridge, UK: Cambridge University Press, 2003.

[136] G. Sánchez and J.-C. Latombe, "On delaying collision checking in PRM planning: Application to multi-robot coordination," *International Journal of Robotics Research*, vol. 21, no. 1, pp. 5–26, 2002.

[137] F. Saust, J. Wille, B. Lichte, and M. Maurer, "Autonomous vehicle guidance on Braunschweig's inner ring road within the stadtpilot project," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2011, pp. 169–174.

[138] M. Schmid, "Umgebungserfassung für Fahrerassistenzsysteme mit hierarchischen Belegungskarten," Ph.D. dissertation, Institut für Technik Autonomer Systeme, Universität der Bundeswehr München, 2012.

[139] E. Schmitzberger, J. L. Bouchet, M. Dufaut, D. Wolf, and R. Husson, "Capture of homotopy classes with probabilistic road map," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 2002, pp. 2317–2322.

[140] M. Schreier, V. Willert, and J. Adamy, "Grid mapping in dynamic road environments: Classification of dynamic cell hypothesis via tracking," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2014, pp. 3995–4002.

[141] F. Schwarzer, M. Saha, and J.-C. Latombe, "Exact collision checking of robot paths," in *Algorithmic Foundations of Robotics V*, ser. Springer Tracts in Advanced Robotics. Berlin, Germany: Springer-Verlag, 2004, vol. 7, pp. 25–42.

[142] G. Shafer, *A Mathematical Theory of Evidence*. Princeton, NJ, USA: Princeton University Press, 1976.

[143] P. Smets, "The combination of evidence in the transferable belief model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 5, pp. 447–458, May 1990.

[144] P. Smets, "Belief functions: The disjunctive rule of combination and the generalized bayesian theorem," in *Classic Works of the Dempster–Shafer Theory of Belief Functions*, ser. Studies in Fuzziness and Soft Computing. Berlin, Germany: Springer-Verlag, 2008, vol. 219, pp. 633–664.

[145] P. Smets and R. Kennes, "The transferable belief model," *Artificial Intelligence*, vol. 66, no. 2, pp. 191–234, Apr. 1994.

[146] Statistisches Bundesamt Deutschland, "Verkehrsunfälle," https://www.destatis.de/DE/Publikationen/Thematisch/TransportVerkehr/Verkehrsunfaelle/VerkehrsunfaelleJ2080700137004.pdf?_blob=publicationFile, Jul. 2014.

[147] A. Stentz, "The focussed D* algorithm for real-time replanning," in *Proc. of the International Joint Conference on Artificial Intelligence*, 1995, pp. 1652–1659.

[148] C. Sung, D. Feldman, and D. Rus, "Trajectory clustering for motion prediction," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1547–1552.

[149] T. Suzuki, M. Hashimoto, and K. Takahashi, "Laser-based road recognition for a smart electric wheelchair," in *Proc. of the IEEE International Conference on Robotics and Biomimetics*, 2011, pp. 993–997.

[150] R. Szeliski, *Computer Vision: Algorithms and Applications*, 1st ed. Berlin, Germany: Springer-Verlag, 2010.

[151] M. Tay, K. Mekhnacha, C. Chen, M. Yguel, and C. Laugier, "An efficient formulation of the bayesian occupation filter for target tracking in dynamic environments," *International Journal of Vehicle Autonomous Systems*, vol. 6, no. 1, pp. 155–171, Jan. 2008.

[152] The Boston Consulting Group, "Self-driving-vehicle features could represent a $42 billion market by 2025," http://www.bcg.com/media/PressReleaseDetails.aspx?id=tcm:12-180096, Jan. 2015.

[153] The Milwaukee Sentinel, "'Phantom auto' will tour city," http://news.google.com/newspapers?id=unBQAAAAIBAJ&sjid=QQ8EAAAAIBAJ&pg=7304,3766749, Dec. 1926.

[154] C. Thorpe, M. Herbert, T. Kanade, and S. Shafter, "Toward autonomous driving: the CMU Navlab part II — architecture and systems," *IEEE Expert*, vol. 6, no. 4, pp. 44–52, Aug. 1991.

[155] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA grand challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, Sep. 2006.

[156] S. Thrun, S. Thayer, W. Whittaker, C. Baker, W. Burgard, D. Ferguson, D. Hahnel, D. Montemerlo, A. Morris, Z. Omohundro, C. Reverte, and W. Whittaker,

"Autonomous exploration and mapping of abandoned mines," *IEEE Robotics and Automation Magazine*, vol. 11, no. 4, pp. 79–91, Dec. 2004.

[157] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. Cambridge, MA, USA: The MIT Press, 2005.

[158] M. J. Thurley and V. Danell, "LTU-CUDA," https://github.com/VictorD/LTU-CUDA, Jul. 2014.

[159] M. Thurley and V. Danell, "Fast morphological image processing open-source extensions for GPU processing with CUDA," *IEEE Journal of Selected Topics in Signal Processing*, vol. 6, no. 7, pp. 849–855, Nov. 2012.

[160] C. Urmson and R. Simmons, "Approaches for heuristically biasing RRT growth," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, 2003, pp. 1178–1183.

[161] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. R. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, Aug. 2008.

[162] M. Van Droogenbroeck and M. J. Buckley, "Morphological erosions and openings: Fast algorithms based on anchors," *Journal of Mathematical Imaging and Vision*, vol. 22, no. 2/3, pp. 121–142, May 2005.

[163] M. van Herk, "A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels," *Pattern Recognition Letters*, vol. 13, no. 7, pp. 517–521, Jul. 1992.

[164] F. von Hundelshausen, M. Himmelsbach, F. Hecker, A. Mueller, and H.-J. Wuensche, "Driving with tentacles: Integral structures for sensing and motion," *Journal of Field Robotics*, vol. 25, no. 9, pp. 640–673, Sep. 2008.

[165] T.-D. Vu, O. Aycard, and N. Appenrodt, "Online localization and mapping with moving object tracking in dynamic outdoor environments," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2007, pp. 190–195.

[166] C.-C. Wang and C. Thorpe, "Simultaneous localization and mapping with detection and tracking of moving objects," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2002, pp. 2918–2924.

[167] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *International Journal of Robotics Research*, vol. 26, no. 9, pp. 889–916, Sep. 2007.

[168] T. Weiss, B. Schiele, and K. Dietmayer, "Robust driving path detection in urban and highway scenarios using a laser scanner and online occupancy grids," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2007, pp. 184–189.

[169] M. Werling and L. Gröll, "Low-level controllers realizing high-level decisions in an autonomous vehicle," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2008, pp. 1113–1119.

[170] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *International Journal of Robotics Research*, vol. 31, no. 3, pp. 346–359, Mar. 2012.

[171] M. Werling, "Ein neues Konzept für die Trajektoriengenerierung und -stabilisierung in zeitkritischen Verkehrsszenarien," Ph.D. dissertation, Institut für Angewandte Informatik/Automatisierungstechnik (AIA), Karlsruher Institut für Technologie, 2010.

[172] W. S. Wijesoma, K. R. S. Kodagoda, and A. P. Balasuriya, "Road-boundary detection and tracking using ladar sensing," *IEEE Transactions on Robotics*, vol. 20, no. 3, pp. 456–464, Jun. 2004.

[173] D. Wollherr, "Design and control aspects of humanoid walking robots," Ph.D. dissertation, Institute of Automatic Control Engineering (LSR), Technische Universität München, 2005.

[174] H.-J. Wuensche, "Bewegungssteuerung durch Rechnersehen," Ph.D. dissertation, Institut für Systemdynamik und Flugmechanik, Universität der Bundeswehr München, 1987.

[175] R. R. Yager, "On the Dempster–Shafer framework and new combination rules," *Information Sciences*, vol. 41, no. 2, pp. 93–137, Mar. 1987.

[176] T. Yang and V. Aitken, "Evidential mapping for mobile robots with range sensors," *IEEE Transactions on Instrumentation and Measurement*, vol. 55, no. 4, pp. 1422–1429, Aug. 2006.

[177] H. Yoo, U. Yang, and K. Sohn, "Gradient-enhancing conversion for illumination-robust lane detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1083–1094, Sep. 2013.

[178] L. A. Zadeh, "Review of a mathematical theory of evidence," *AI magazine*, vol. 5, no. 3, pp. 81–83, 1984.

[179] H. Zhang, X. Xiao, and O. Hasegawa, "A load-balancing self-organizing incremental neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 6, pp. 1096–1105, Jun. 2014.

[180] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. Keller, E. Kaus, R. Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knöppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein, and E. Zeeb, "Making Bertha drive – an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, Summer 2014.

[181] J. Ziegler and C. Stiller, "Fast collision checking for intelligent vehicle motion planning," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2010, pp. 518–522.

[182] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for Bertha - a local, continuous method," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2014, pp. 450–457.

Online-Buchshop für Ingenieure

VDI nachrichten **BUCHSHOP**

Online-Shops

Fachliteratur und mehr - jetzt bequem online recherchieren & bestellen unter: www.vdi-nachrichten.com/Der-Shop-im-Ueberblick

Täglich aktualisiert: Neuerscheinungen VDI-Schriftenreihen

Im Buchshop von vdi-nachrichten.com finden Ingenieure und Techniker ein speziell auf sie zugeschnittenes, umfassendes Literaturangebot.

Mit der komfortablen Schnellsuche werden Sie in den VDI-Schriftenreihen und im Verzeichnis lieferbarer Bücher unter 1.000.000 Titeln garantiert fündig.

Im Buchshop stehen für Sie bereit:

VDI-Berichte und die Reihe **Kunststofftechnik:**

Berichte nationaler und internationaler technischer Fachtagungen der VDI-Fachgliederungen

Fortschritt-Berichte VDI:

Dissertationen, Habilitationen und Forschungsberichte aus sämtlichen ingenieurwissenschaftlichen Fachrichtungen

Newsletter „Neuerscheinungen":

Kostenfreie Infos zu aktuellen Titeln der VDI-Schriftenreihen bequem per E-Mail

Autoren-Service:

Umfassende Betreuung bei der Veröffentlichung Ihrer Arbeit in der Reihe Fortschritt-Berichte VDI

Buch- und Medien-Service:

Beschaffung aller am Markt verfügbaren Zeitschriften, Zeitungen, Fortsetzungsreihen, Handbücher, Technische Regelwerke, elektronische Medien und vieles mehr – einzeln oder im Abo und mit weltweitem Lieferservice

Die Reihen der Fortschritt-Berichte VDI:

1 Konstruktionstechnik/Maschinenelemente
2 Fertigungstechnik
3 Verfahrenstechnik
4 Bauingenieurwesen
5 Grund- und Werkstoffe/Kunststoffe
6 Energietechnik
7 Strömungstechnik
8 Mess-, Steuerungs- und Regelungstechnik
9 Elektronik/Mikro- und Nanotechnik
10 Informatik/Kommunikation
11 Schwingungstechnik
12 Verkehrstechnik/Fahrzeugtechnik
13 Fördertechnik/Logistik
14 Landtechnik/Lebensmitteltechnik
15 Umwelttechnik
16 Technik und Wirtschaft
17 Biotechnik/Medizintechnik
18 Mechanik/Bruchmechanik
19 Wärmetechnik/Kältetechnik
20 Rechnerunterstützte Verfahren (CAD, CAM, CAE CAQ, CIM ...)
21 Elektrotechnik
22 Mensch-Maschine-Systeme
23 Technische Gebäudeausrüstung

ISBN 978-3-18-524608-1