DISS. ETH NO. 19260

AUTONOMOUS VEHICLE NAVIGATION IN DYNAMIC URBAN ENVIRONMENTS FOR INCREASED TRAFFIC SAFETY

A dissertation submitted to

ETH ZURICH

for the degree of

Doctor of Sciences

presented by

KRISTIJAN MAČEK

M.Sc., University of Zagreb

03 December 1975

citizen of Slovenia and Croatia

accepted on the recommendation of

Professor Roland Siegwart, Principal Supervisor
Professor Urbano Nunes, Co-Examiner
Professor Ivan Petrović, Co-Examiner
Doctor Thierry Fraichard, Co-Examiner

2010

Abstract

In recent years, there has been an extensive amount of research done on various aspects of general vehicle traffic. It has been recognized both by industry manufacturers and research units that introducing intelligent driver assistance systems on-board vehicles and enhancing the road infrastructure with additional networked sensory information has an important impact on future of transportation. On one side, these systems can greatly improve both traffic flow, reduce energy consumption and emissions and on the other size they can enhance driver comfort and safety of all the traffic participants.

This thesis addresses in particular the safety aspect of urban traffic by exploring the potentials of autonomous vehicle navigation using on-board proprio- and exteroceptive sensors and actuators that are used in a closed-loop fashion from dynamic scene analsis to decision making based on hierarchical planning techniques. An autonomous vehicle navigation framework implicitly enables three principal driver mitigation levels: in the passive mode it can provide an informative scene interpretation and a warning signal in critical situations, in the semi-autonomous mode it can act as an overriding copilot in order to prevent accidents potentially provoked by driver misjudgement or fatigue and in the fully autonomous mode it enables a situation adaptable driving mode in which respecting the traffic participants safety and regulations can be explicitly embedded in the system.

The development of autonomous navigation framework in this work follows the categorization of the vehicle environment both from perceptual and motion planning problem into static and dynamic ones. Static environments may include a-priori unknown static obstacles or slowly moving ones where the underlying assumption is the that the environment representation and motion planning can be based on instantaneous sensory information or the frozen world assumption. Since the full information about obstacle configurations is not known a-priori, a success-

ful autonomous navigation scheme requires both a global on-line path planning capability for objective deliberation and a reactive obstacle avoidance layer for safety. The contribution for these types of environments is in a novel hierarchical navigation structure that combines different available global path planning and path following schemes with obstacle avoidance methods for an occupancy grid based environment representation. Experimental results are provided for a vehicle platform under real-world conditions. Alternative formulations to obstacle avoidance via a probabilistic Bayesian inference and to motion planning problem using optimization techniques were also studied.

The frozen world assumption is violated in highly dynamic environments, therefore the timing constraints arising from the dynamic object perception, motion planning and execution have to be explicitly accounted for. Two distinct motion planning approaches were proposed in the presence of dynamic objects, in transformed state space and in trajectory space of vehicle and obstacle motion. The key component is the partial motion planning module, where potentially feasible vehicle trajectories are explored within a given decision time limit. The trajectory exploration phase is directed both by attaining the goal configuration as well as exploring the free space in order to avoid collision with static and dynamic obstacles. Moreover, the safety aspect of motion planning in dynamic environments is a critical issue, due to the inherent uncertainty about future obstacle motion, sensory limitations and real time planning requirements. Two safety levels are distinguished, namely the passive safety where the ego-vehicle is guaranteed to come to a stop before any future potential collision could occur beyond the prediction horizon and the passive friendly safety, where other dynamic objects are allowed to come to a stop if they follow a particular breaking policy.

In order to close the loop from perception to motion planning, the dynamic scene analsis was performed, where the road networked structure was described via lane detection. The potential dynamic object candidates were extracted by building an incremental locally consistent environmental map based on on-line

sensory data and detection of temporal differences in map occupancy evidence.

The ultimate goal pursued in this thesis was to development a navigation architecture that would allow for safe autonomous vehicle navigation in dynamic urban scenarios. In order to successfully navigate in such environments, a planning hierarchy must be established from the strategic mission plan, global waypoints route planning followed by a tactical behavioral level which triggers different vehicle behavior modes according to the given traffic situation and rules, further down to the operational level which must generate feasible trajectories for the vehicle to be executed. In this respect, a hierarchical navigation scheme is proposed and verified in simulation given various dynamic urban scenarios.

Zusammenfassung

In den letzten Jahren wurden verschiedene Aspekte von allgemeinem Fahrzeugverkehr zunehmend gründlich recherchiert. Industrielle Hersteller sowie auch Forschungszentren sind zur Schlussfolgerung gekommen, dass die Einführung intelligenter Fahrzeugassisstenzsystemen auf Bord Fahrzeugen und die Erweiterung der Strasseninfrastruktur durch zusätzliche vernetze sensorielle Informationsstellen einen erheblichen Einfluss auf das zukünftige Transportwesen haben werden. Auf einer Seite können solche Systeme den Verkehrsdurchfluss deutlich verbessern, den Energieverbrauch und Emission reduzieren, auf anderer Seite aber auch den Fahrkomfort und die Sicherheit aller Verkehrsteilnehmer steigern.

Diese Doktorarbeit befasst sich in besonderem mit dem Sicherheitsaspekt des städtischen Strassenverkehrs durch die Erforschung des Potenzials autonomes Fahrzeugfahrens. Es wurden propriozeptive und exterozeptive Sensoren und Aktuatoren auf Bord des Fahrzeuges angewendet. Die Rückführung wurde von Erfassung und Analyse dynamischer Umgebungen bis zur hierarchischen Bewegungsplannung geschlossen. Ein autonomes Fahrzeugnavigationssytem ermöglicht impliziterweise drei hauptsächliche Fahrerassistenzstufen: im passiven Modus ermöglicht das System eine informative Umgebungsanalyse dass einen Warnsignal in kritischen Situationen generieren kann, im semi-autonomen Modus agiert das System als ein übergeordneter Kopilot dass die Verkehrsunfällen vermeiden lässt, die durch die falsche Situationseinschätzung oder Müdigkeit des Fahrers potenziell entstehen können und im autonomen Modus adaptiert das System die Fahrweise automatisch situationsgemäss so das sich die Sicherheit der Verkehrsteilnehmer und die Verkehrsregeln explizit miteinbeziehen lassen.

In dieser Arbeit basiert die Entwicklung eines autonomomen Navigationssystems mittels Kategorisierung der Fahrzeugumgebung in statische und dynamische Typen. Diese Kategorisierung ist bei Umgebungserfassung- sowie bei Bewegungsplannungproblemen angewendet worden. Die statische Fahrzeugumgebung kann de-

mentsprechend statische oder langsam bewegliche Hindernisse enthalten die im Vorraus nicht bekannt sind. Die grundlegende Annahme besteht somit darin, dass die Umgebungsbeschreibung und Bewegungsplannung nur auf unmittelbarer sensorieller Information basieren, der sogenannten gefrorenen Welt Annahme. Da die umfassende Information über die Umgebungshindernisse zunächst nicht vorhanden ist, muss ein erfolgreiches autonomes Navigationssystem eine globale on-line Pfadplannungfunktionalität für Zielstellung enthalten sowie eine reaktive Kollisionsvermeidungschicht die für Sicherheit verantwortlich ist. Der Beitrag zur Navigation in statischen Fahrzeugsumgebungen ist eine neuartige hierarchische Navigationsstruktur die verschiedene vorhandene globale Pfadfandungs- und Pfadverfolgungschemas mit Kollisionsvermeidungschicht kombiniert wobei die Umgebungsbeschreibung durch eine Griddarstellungsstruktur konstruiert wird. Ergebnisse experimenteller Testversuchen an einer Fahrzeugtestplatform unter reellen Bedingungen sind präsentiert worden. Eine alternative Problemformulierung zur Kollisionsvermeidung durch bayesische Inferenz und die Anwendung Optimizationstechniken zur Bewegungsplannung wurden auch analysiert.

Die Annahme zur sogennanten gefrorenen Welt ist in intensiv dynamischen Umgebungen nicht mehr gültig. Die zeitlichen Beschränkungen die durch die Umgebungserfassung- Bewegungsplannung- und Bewegungsausführungsproblemen entstehen, müssen explizit in Berechnung genommen werden. Zwei verschiedene Ansätze zur Bewegungsplannung in Umgebungen mit dynamischen Hindernissen wurden vorgeschlagen, nämlich die Bewegungsplannung im transformierten Zustandsraum und die Bewegungsplannung im Trajektorienraum der Fahrzeug- und Hindernissenbewegung. Eine der wichtigsten Komponenten ist der Partieller-Bewegungsplannungmodul wodurch potenziell ausführbare und zulässige Trajektorien wärend der Dauer der vorgegebenen zeitlichen Entscheidungsgrenzwert gesucht werden. Die Trajektorien-Suchphase wird durch Zielorientierung wie auch durch die Suche im freien Raum gelenkt, damit die statischen und dynamischen Hindernisse vermieden werden können. Überdies ist die Sicherheit der Be-

wegungsplannung in dynamischen Umgebungen ein kritischer Aspekt wegen der inhärenten Ungewissheit über zukünftige Hindernissenbewegungen, Sensoreneinschränkungen und Anspruch auf Echtzeit Bewegungsplannung. Zwei Sicherheitsstufen wurden hier differenziert. Die erste ist die passive Sicherheit wodurch garantiert wird, dass das Eigenfahrzeug zum Stillstand gebremst werden kann eher eine zukünftige Kollision jenseits des Vorhersage-Horizonts auftreten könnte. Die zweite ist die passiv-freundliche Sicherheit wodurch garantiert wird, dass neben dem Eigenfahrzeug auch die dynamischen Objekten zum Stillstand gebremst werden können, falls sie gewisse Bremsmanöver ausführen.

Damit die Rückführung von Umgebungserfassung bis zur Bewegungsplannung geschlossen werden kann wurde die Analyse und Beschreibung der dynamischen Fahrzeugumgebung durchgeführt. Die Strassenstruktur wurde durch Strassenkanten-Erkennung beschrieben. Die on-line vorhandenen sensorischen Daten wurden in einer inkrementaler, lokalerweise konsistenter Umgebungsmape fusioniert. Die potenziellen dynamischen Objekten-Kandidaten wurden danach anhand zeitlicher Differenz des Belegungsgrads der Umgebungsmape extrahiert.

Das Endziel dieser Doktorarbeit war die Entwicklung einer Navigationsarchitektur dass eine sichere autonome Fahrzeugnavigation in dynamischen städtischen Umgebungen ermöglichen würde. Damit dieses Ziel erreicht werden kann, muss eine Bewegungsplannung-Hierarchie aufgesetzt werden. Diese Bewegungsplannung-Hierarchie fängt mit einer strategischen Zielsetzung und Plannung der globalen Wegpunkten-Fahrstrecke an. Es wird von einer taktischen Verhaltensebene gefolgt, wo gemäss der gegebenen Verkehrssituation verschiedene Fahrzeugverhaltensmodi aufrufbar sind. Eine weitere operative Navigationsebene ist für die Generierung und Ausführung einer zulässigen Fahrzeugstrajektorie verantwortlich. Im Sinne der erwähnten Navigationsebenen wurde ein hierarchisches Navigationsschema vorgeschlagen und durch Simulation verschiedener dynamischen städtischen Szenarien verifiziert.

Acknowledgements

In the first place I would like to express my thanks to *Prof. Roland Siegwart*, my principal advisor, for the opportunity of pursuing a PhD at the Autonomous Systems Lab. It has been a great learning experience both professionally and personally. Roland's immense enthusiasm and commitment to new ideas in the robotic research has been a great source of inspiration. His open discussion and comprehension has given my work at Autonomous Systems Lab the necessary guidance, support and independence to accomplish my PhD.

To my thesis committee members *Prof. Ivan Petrović*, *Prof. Urbano Nunes* and *Dr. Thierry Fraichard* for accepting to be my co-examiners and examining carefully the presented work.

I have shared many days of experimental work, developments, discussions and presentations with colleagues *Sascha Kolski* and *Luciano Spinello*. To *Dizan Vasquez* who supported me with developing the simulation tools necessary to test navigation algorithms as well as many interesting discussions on robotics. To *Thierry Fraichard* for very enriching discussions on motion planning and constructive revision of my research in this field. To *Roland Philippsen* and *Dave Fergusson* for their contributions in global planning algorithms that have been applied in this work. To *Ralf Kästner* who has been a great colleague providing support on experimental vehicle development. My personal research work with the Smart testing vehicle would not have been possible without software and hardware contributions of *Pierre Lamon*, *Rudolph Triebel*, *Francois Pomerleau*, *Frédéric Pont*, *Marcelo Becker*, *Felix Mayer*, *Cyrill Stachniss*, *Janosch Nikolic*, *Stefan Bertschi, Markus Bühler*, *Dario Fenner*, *Daniel Burnier*, *Tarek Baaboura* and many students and external partners who participated in vehicle platform development. To *Konrad Thoma* and *Richard Glatzel* on their contribution to dynamic vehicle modelling and *Brian Williams* on his contribution to the lane detection module presented in this work.

To *Margot Fox-Ziekau*, *Luciana Borsatti* and *Marie-José Pellaud* for their valuable help in all administrative matters.

To all the current and former members of the Autonomous Systems Lab who contributed to the productive but also culturally enriching environment that the Lab is. I would like to mention *Stefan Gächter, Viet Nguyen*, *Davide Scaramuzza*, *Samir Bouabdallah*, *Xavier Perrin*, *André Noth*, *Fabien Tâche*, *Ambroise Krebs*, *Jérome Mâye*, *Christian Bermes*, *Martin Rufli*, *David Remy*, *Masoud Asadpour*, *Majid Nili*, *Ahad Harati*, *Shrihari Vasudevan*, *Christian Bacs*, *Jens Bathelt*, *Wolfgang Fischer*, *Gilles Caprari*, *Stephan Weiss*, *Markus Höpfinger*, *Stefan Leutenegger*, *Thomas Tüer*, *Francesco Mondada*, *Björn Jensen*, *Jan Weingarten*, *Jiwon Shin*, *Cédric Pradalier*, *Francis Colas*, *Agostino Martinelli*, *Adriana Tapus*, *Carmen Kobe* who made my stay with the Autonomous Systems Lab ever so more enjoyable.

To my former university mentors *Prof. Ivan Petrović*, *Prof. Nedjeljko Perić* and colleagues *Jadranko Matuško*, *Miroslav Barić* and *Mato Baotić* at the Faculty of Electrical Engineering and Computing Zagreb for their kind support during my visiting student exchange to Switzerland preceeding my PhD studies.

To *Paulo Augusto Dal Fabbro*, *Alexandre Dal Fabbro*, *Mark Butcher*, *Pietro Bucella*, *Marcelo Leite Ribeiro*, *Danijel Močibob*, *Luis Borges*, *Sandy El Helou*, *Joana Comas*, *Montserrat Badia*, *Sabrina Carvalho*, *Simone Voldrich*, *Vlasta Merc*, *Marko Budimir*, *Dragan Čorić*, *Ana Paragiš*, *Ana Vidiš*, *Mario Barešić*, *Mario Starc*, *Marina Starc*, *Polona Alt* and other dear friends that filled my PhD time in Switzerland with lots of nice moments.

Finally, to my brothers *Mario* and *Robert* as well as their partners *Saša* and *Alenka*, my mum *Marija* and my dad *Marijan*, for their love, care and support, always.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and problem statement

1.1.1 Motivation - current traffic safety situation

According to a National Highway Traffic Safety Administration (NHTSA) completed statistic report for years 2007 and 2008 there have been $41,259$ and $37,261$ total fatalities in traffic crashes in these years, respectively, in the USA alone [NHTSA, 2009]. Passenger vehicle casualties amounted to $29,079$ (70.7%) and $25,251$ (67.7%), whereas the pedestrian casualties amounted to $4,699$ (11.4%) and $4,378$ (11.8%). The rest of the casualties belong to the groups large trucks, motorcycles, pedalcyclists and other. It has also been estimated that 2.49, respectively 2.35 million people were injured (non-fatal) in these years alone. Though there is a overall decline in number of casualties over the last years, notably about 8.4% in 2009 with respect to 2008 according to the unfinished statistics of year 2009 [NHTSA, 2010], the human implications of these accidents are still significant. Due to similar technological and traffic safety regulations levels in Europe, comparable unfortunate figures persist, e.g. $41,304$ passenger vehicle casualties and 1.7 million people injured in the year 2005 [SEiSS, 2005], [eImpact, 2008].

Putting in perspective also the economic implications of traffic accidents, which

include medical and insurance costs, workspace productivity loss and property damage among others, a study of the year 2000 shows the total cost of $230.6 billion in the USA alone [NHTSA, 2002] equaling 2.3% of the USA Gross Domestic Product (GDP). This is approximately the amount a typical Western country spends yearly on education and research activities, for instance. Very similar costs incur in Europe also, e.g. €160 billion in 2005, roughly amounting to 2% of Europe's GDP [SEiSS, 2005].

In presence of this figures, there is a clear need to improve the overall traffic safety situation. The general category of new technologies under which the road transportation improvements are made in recent years are called the **"Intelligent Transportation Systems" - ITS**. These include systems for improving the overall safety and efficiency of road transportation as well as user comfort and convenience. Conservative estimates made for the OECD countries ("Organization for Economic Co-Operation and Development" - representing a vast majority of world's industrial countries) as to the benefits of ITS in the next 20 years are following [OECD, 2003]:

- ITS could save as many as $47,000$ lives per year;
- ITS could potentially reduce the total number of road crash injuries and fatalities by approximately 40%;
- The savings related to a 40% reduction in injuries and fatalities would be approximately USD 194 billion annually.

1.1.2 Problem statement

Exploration of the methods and potential of fully autonomous vehicle navigation is the primary focus of this thesis. This task of finding essentially a vehicle command at each time instant that enables feasible (safe and potentially optimal) navigation of the ego-vehicle in a static or dynamic environment towards a given goal can provide indispensable insight into the future extension of ITS systems that would

increase the autonomous vehicle maneuverability and consequentially the overall traffic safety level as well. This could be achieved by scaling down in principle a huge set of possible motion planning solutions to the ones acceptable for traffic hazard mitigation.

At the other extreme, a future traffic situation can be envisaged where the vehicles operate permanently in the autonomous mode, therefore putting the driver "out of the equation". There are numerous implications to such a scenario but there is one that is particularly important - without the driver a huge behavioral uncertainty is removed from the analysis, enabling very precise vehicle motion prediction, planning and control. Since all the controlled vehicles would follow predefined navigation patterns, the only real uncertainty would remain in interpreting the motion of uncontrolled traffic participants, such as pedestrians, cyclists, animals, etc. and correct assessment of the structural characteristics of the road itself. This would enable a very precise analysis of possible critical scenarios and defining the upper bound of how safe the vehicle traffic can be in presence of autonomous vehicle agents. Moreover, due to the orderly driving patterns, the overall traffic flow could be significantly increased. For instance, the current safety distance/time recommendations that are applicable for a human driver could be drastically reduced.

In order to develop a viable autonomous vehicle system, a possible structure to some of the key aspects is proposed here:

1. Vehicle dynamics modeling.
2. Autonomous navigation scheme for static and unstructured environments.
3. Autonomous navigation scheme for dynamic environments with explicit obstacle motion modeling.
4. Dynamic scene analysis.
5. Hierarchical navigation scheme for urban autonomous vehicle navigation.

In this thesis, each of the identified elements is analyzed individually and validated according to the proposed algorithmic solutions.

1.2 State of the art

In order to identify the current state of the art in the Intelligent Transportation Systems domain both approaches stemming from automotive industry research and development as well as that of the robotic community are referenced here.

1.2.1 Intelligent Transportation Systems (ITS)

The Intelligent Transportation Systems (ITS) can be classified according to different criteria, possibly the most generic being [OECD, 2003]:

- **Vehicle-Based Safety Systems**: these safety systems assume presence of on-board sensors that process the traffic conditions in real time. Based on the situation assessment additional on-board units can issue warnings or take partial to full control of the vehicle, depending on the mitigation level between the driver and the ITS. The advantage of these systems is that they can warn the driver of potential dangers or override to a specified degree the driver's control of the vehicle in attempt to avoid collisions. Notably, these benefits are only available to vehicles equipped with such on-board equipment. Nowadays, the open issues/problems concerning these systems are mainly based on the need to correctly asses many various, complex and partially unpredictable traffic situations. Besides guaranteeing reliability it is also important to make drivers aware to which extend the on-board safety systems are able to reduce the traffic dangers.

- **Infrastructure-Based Safety Systems**: these safety systems are primarily comprised of roadside sensors that collect information and roadside equipment that issues warnings and advises on different actions to take. The

advantages of these systems are measurement of different phenomena and detection of events that on-board sensors typically cannot detect, such as weather conditions, obstacles and traffic around curves or in the distance (including the area traffic infrastructure). Variable data can be provided on roadside signs and information can be provided to all potentially affected vehicles in the vicinity. A problem associated with infrastructure-based systems is that the data must be standardized to improve driver understanding of the provided information. Moreover, whereas the additional investments due to such systems on new road networks may be considerate, the adaptation of the already existing road infrastructure may pose an even bigger problem.

- **Co-operative Safety Systems**: these safety systems utilize both infrastructure based and vehicle based systems with communication links between them. The advantage of these systems is that information is received from the infrastructure (e.g. speed limits, traffic and road conditions) and provided dynamically at the appropriate time to individual vehicles. Information can also be transmitted in the opposite direction, i.e. from vehicle to infrastructure, for example to automatically notify emergency services when a vehicle is in a collision. Such services can only be provided to vehicles that are equipped with on-board units. Digital maps and technologies to pinpoint exact locations are also considered to be co-operative technologies, since safety-related information can be combined with the maps stored in the on-board equipment, and a wider service area can be set as compared with information provided by the infrastructure. Issues particularly related to co-operative systems include the need to maintain a balance between system safety, reliability and cost and the standardization of the human-machine interface.

Since each of the aforementioned groups of safety systems targets specific safety functions, it is to be expected that the ITS which will be operational in the general

transportation in the following years will most probably represent an integral solution of these complementary subsystems.

1.2.2 Advanced Driver Assistance Systems (ADAS)

In this work the focus will be related to the **Vehicle-Based Safety Systems**, also referred to as **"Advanced Driver Assistance Systems" - ADAS** in the narrow sense. An important classification of the ADAS can be done according to the urgency with respect to a potential crash situation. A timeline is presented in Fig. 1.1 which distinguishes between five nominal modes of operation, namely:

1. Normal Driving: the system provides support and information to the driver.
2. Warning and Assistance (Emergency): the vehicle predicts a dangerous situation and optionally assists the driver.
3. Pre-crash: the crash is unavoidable, the vehicle is being prepared for the crash.
4. In-crash: the crash happens.
5. Post-crash: the crash has been happened and emergency services are approaching.

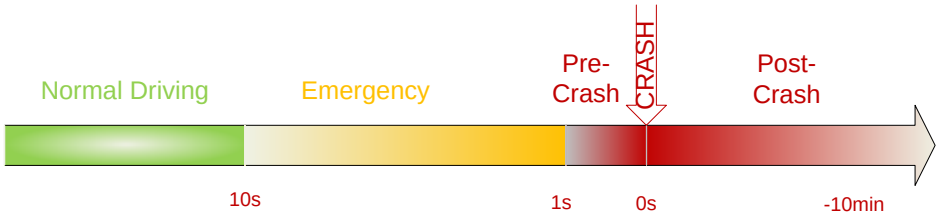


Figure 1.1: ADAS operation modes with respect to the crash event.

The timeline values are orientational only, since the emergency of a driving situation depends heavily on the vehicle speed. Each of these stages with respect to the crash event require different levels of safety vs. comfort engagement of

the ADAS systems, starting as comfort functions at the beginning of the timeline with a growing emphasis on safety as the crash event approaches. Interestingly, the safety-relevant systems are often introduced as "comfort" systems, since it is known from experience that buyers would rather pay for comfort than safety [SEiSS, 2005].

The Tab. 1.1 and Tab. 1.2 show a set of low-level vehicle control systems that support the Advanced Driver Assistance Systems [Wiki-ADAS-en, 2010], [Wiki-ADAS-de, 2010]. The complexity of these low-level driving and safety support systems can be very high and their presence indispensable for the overall vehicle functionality and safety. The drawing line between low-level vehicle control systems and complex structural adaptations with respect to the "Advanced" Driver Assistance Systems is therefore fluid. The term "Advanced" can be more properly understood as high level warning, assistance and control, where the interaction level with the driver takes place at nominal driver perception and control levels, such as detecting critical driving related features of the environment (e.g. vehicles and other participants), assessing the emergency level, notifying/warning the driver, controlling the vehicle to a predefined setpoint and reacting to immediate dangers in form of emergency maneuvers.

The Tab. 1.3, Tab. 1.4 and Tab. 1.5 show a non-exhaustive list of Advanced Driver Assistance Systems (ADAS) ordered approximately according to increasing time emergency (ref. to Fig. 1.1) [SEiSS, 2005], [Continental-ADAS, 2010], [Wiki-ADAS-en, 2010], [Wiki-ADAS-de, 2010]. Manufacturers may use different commercial names when referring to similar type systems, therefore the most common nomenclature is used here.

The introduction of ADAS to the vehicle transportation through research, development and finally serial production in newer generation vehicles can be viewed as a "bottom-up" technological development. Each of the ADAS targets a specific driving and vehicle operation functionality with increasing "higher-level" driving conditions analysis and control. Moreover, the ADAS have been introduced to the

market gradually over several last decades, according to the rate of technological progress but even more importantly due to the incremental way the market is able to absorb these new products from the socio-economic and regulatory point of view. In fact, many up-to-date regulations and recommendations concerning the traffic safety have been formed a-posteriori based on available pre-production ADAS solutions coming from the automotive industry.

When analyzing the level of autonomicity of current ADAS it can be stated that on the level of steering mitigation, the Lane Keep Assist (LKA) and the Lane Change Support (LCS) (Tab. 1.4) are the cutting-edge solutions in operation. They both aim at keeping the ego-vehicle in the current lane due to possible departure or rear-approaching vehicles when changing lanes, respectively. As far as longitudinal velocity control is concerned the Follow-To-Stop Assist (Tab. 1.5), a combination of Adaptive Cruise Control (ACC) and Active Brake Assist (ABA), solves the problem of vehicle following in front and performing emergency braking if a critical (crash) situation in front of the ego-vehicle is detected. In summary, the autonomicity of the current ADAS solutions are limited to a set of well defined control patterns within the current lane.

Clearly, these ADAS advancements will have a huge impact on overall traffic safety in the present and in near future. However, it is also evident that the full range of potentially useful maneuvers has not been exploited yet in the current ADAS implementations. Some interesting useful maneuvers are presented in Tab. 1.6. These and similar maneuvers could be considered as a subset of feasible motion planning solutions within the scope of fully autonomous vehicle navigation which will be discussed in Sec. 1.2.3.

1.2.3 Autonomous vehicle navigation

The problem of navigation of autonomous agents, which in essence comprise of environment perception analysis and task planning and execution part, has been an important focus of research within the robotic community for several decades.

Navigation algorithms have been developed and tested on various robotic platforms. First reports on autonomous vehicle navigation go back two decades ago.

VaMoRs was a 5-ton test vehicle are given being able to stopping in front of obstacles of at least $0.5\,\mathrm{m}^2$ cross section on unmarked two-lane roads at velocities up to 40 km/h [Dickmanns, 1991], [Dickmanns and Christians, 1991]. Within the NavLab project a color-based road classifier and follower was developed [Thorpe *et al.*, 1988], [Thorpe *et al.*, 1991a], [Thorpe *et al.*, 1991b]. A Mercedes-Benz T-model OSCAR perforemed lane-keeping task based on vision-based lane detection and a steering wheel neural controller what learned the closed-loop nonlinear dynamics of the vehicle on-line. The vehicle drove up to $180\,\mathrm{km/h}$ on a public highway [Neusser *et al.*, 1993]. Within the EUREKA framework project PROMETHEUS (Program for European Traffic with Highest Efficiency and Unprecedented Safety). a van for automatic driving on a private road network was reported with a subsequent second generation sedan vehicle commissioned for driver-vehicle interactions studies with the option to automatically evaluate actual traffic situations around the vehicle in real-time [Nagel *et al.*, 1995]. The successor demonstrator to the VaMoRs truck was VaMP, a Mercedes passenger car which autonomously performed missions on German freeways. VaMP perceived the environment with its sense of vision and radarbased sensors and controlled its actuators for locomotion and attention focussing. A hierarchical system architecture for autonomous visual road vehicle guidance was developed, which consists of four levels: a vehicle level, a 4D-level, a rule-based level, and a knowledge-based level [Maurer *et al.*, 1996], [Maurer and Dickmanns, 1997], [Gregor *et al.*, 2002]. The VaMP demonstrator was indeed one of the most referenced platforms belonging to the last decade's autonomous vehicle research achievements. It drove more than 2,000 km from Munich to Copenhagen and back in traffic at up to $180\,\mathrm{km/h}$, planning and executing maneuvers to pass other cars with safety driver approval. However, on a few critical situations, such as unmodeled construction areas a safety driver took over completely. Moreover, despite its huge distance covered, its operation was

still limited to the highly structured highway environment.

The main target of the ARGO project was the development of an active safety system with the ability to act also as an automatic pilot for a standard road vehicle with a real-time embedded vision system as the main perceptional framework [Bertozzi and Broggi, 1999], [Bertozzi *et al.*, 2000], [Broggi *et al.*, 2001]. Another PROMETHEUS project based demonstrator VITA II which enabled autonomous driving on highways was assigned several vision modules to a set of processors in order to study algorithmic and system-architecture challenges for complex urban traffic. The perceptional tasks included road structure detection, traffic participants detection and tracking, traffic sign recognition. The authors introduced their version of an intelligent Stop-and-Go system and discussed appropriate algorithms and approaches for vision-module control [Franke *et al.*, 1998], [Paetzold and Franke, 2000]. The follow-up UTA project (Urban Traffic Assistant) aimed at accomplishing a fully functional Stop-and-Go system for an innercity environment. The DaimlerChrysler demonstrator UTA II was designed with special attention for information, warning and intervention systems for this purpose. Whereas, the radarbased advanced cruise control was already commercialized by DaimlerChrysler (DC) in 1999 in their premium class vehicles and a vision-based lane departure warning system for heavy trucks was introduced by DC in 2000, the UTA II demonstrator was geared-up for the full driver assistance task list available in the year 2000 - a stereo-based object detection and tracking, pedestrian recognition based on neural network, lane detection and tracking on the highway and in the city, traffic signs and traffic light recognition, recognition of road markings, crosswalk recognition, vehicle classification, vehicle control (lateral/longitudinal), driver interface (2D/3D visualization) [Franke *et al.*, 2001].

Recent developments in the field of autonomous vehicle navigation are closely related to the DARPA ("Defense Advanced Research Projects Agency") [Web-DARPA, 2010a] autonomous vehicle competitions. In the years 2004 and 2005 two DARPA Grand Challenges took place in the off-road Mojave desert area, Nevada

[Web-DARPA, 2010b], [Web-DARPA, 2010c]. The autonomous vehicles were required to take course in different terrains such as dirt roads, trails, lakebeds and rocky terrain. In 2004 none of the 15 finalists completed a 7.4 miles long course within a 10 h limit which proved to the difficulty of developing an autonomous robot that is robust, perceptive, and intelligent enough to travel long distances in unstructured terrain. In fact, the CMU vehicle Sandstorm came furthest to a 7.36 miles length before getting stuck on an embankment. In the second DARPA Grand Challenge in 2005, 195 applicants started preparing for the race, whereas only 23 finalist vehicles were allowed to the final race track after the elimination phase of the National Qualification Event (NQE). Interestingly, 22 of the 23 finalists traveled farther than the 7.36 miles traversed by the most successful entry from 2004. In the end, 5 vehicles successfully completed the course with The Stanford Racing Team's vehicle platform Stanley completing the 132 miles long unpaved course in only 6 h,53 min [Iagnemma and Buehler, 2006], [Kurjanowicz, 2006], [Buehler, 2006].

The DARPA Urban Challenge was held on in November of year 2007 at a former airforce base in California. This event differed from the previous two in the sense that it required teams to build an autonomous vehicle capable of driving in traffic, performing complex maneuvers such as merging, passing, parking and negotiating intersections [Buehler *et al.*, 2008], [Web-DARPA, 2010d]. In addition to changing the environment to the urban like scenario that simulated a city-like traffic, this event was also unique as it was the first time autonomous vehicles have interacted with both manned and unmanned vehicle traffic. The manned vehicles were driven by the DARPA instructors, whereas the unmanned ones were the autonomous vehicles competing in the same traffic area in order to reach their goal objectives, handed over to the teams just prior to the respective vehicle's start time. The objectives were simulated military supply missions, i.e. reaching distinctive waypoints in a pre-determined order. While in the run mode, the vehicles merged into moving traffic, navigated traffic circles, negotiated busy

intersections and avoided various obstacles. Race rules required that the 96 km (60 mile) course be completed in less than 6 hours. The vehicles were carefully monitored to obey all traffic regulations while avoiding other competitors and humandriven traffic vehicles. If a vehicle failed to follow a traffic rule or performed a dangerous maneuver, it was disqualified immediately from the race.

Initially, there were 68 teams that applied to the DARPA Urban Challenge, of which 35 teams were invited to the National Qualification Event (NQE) which was a rigorous eight-day vehicle testing period. Thereafter, 11 finalists made it to the final event whereas 6 vehicles actually finished the race successfully. The CMU Tartan Racing Team's vehicle Boss accomplished the track in 4 h,10 min at the average speed of 22.53 $^{\mathrm{km}}/_{\mathrm{h}}$ [Urmson *et al.*, 2008]. Second placed Stanford Racing's vehicle Junior performed very closely at 4 h,29 min at the average speed of 22.05 $^{\mathrm{km}}/_{\mathrm{h}}$ [Montemerlo *et al.*, 2008a]. Similarly, the third placed Virginia Tech Victor Tango's vehicle Odin finished at 4 h,36 min at the average speed of 20.92 $^{\mathrm{km}}/_{\mathrm{h}}$ [Bacha *et al.*, 2008]. These three vehicle teams were the clear winners, however worth noting are also the other three finishing teams, notably MIT with Talos [Fletcher *et al.*, 2008], University of Pennsylvania Ben Franklin Racing Team with Little Ben [Bohren *et al.*, 2008] and Cornell's vehicle Skynet [Miller *et al.*, 2008].

Low-level vehicle control system	**Function**	**Technical solution**
Electric Power Steering (EPS)	Reduce the torque effort by providing steering assist to the driver	An electric motor coupled directly to either the steering gear or steering column
Powertrain Control (PCM)	Engine and transmission control	An electronic component consisting of engine and transmission control unit
Traction Control System (TCS) or Anti-Slip Regulation (ASR)	Prevent loss of traction to the driven road wheels	Combined throttle and individual brake pressure control
Electronic Stability Control (ESC)	Improves the safety of a vehicle's stability by detecting and minimizing skids	Individual brake pressure control, optionally also engine control using additional inertial measurement unit input data
Roll Stability Control (RSC)	Improves vertical vehicle's stability, impending rollover	Individual brake pressure control, optionally also engine control using additional inertial measurement unit input data
Antilock Braking System (ABS)	Prevents the wheels from locking up and ceasing to rotate while braking	Individual brake pressure applied to each wheel
Emergency Brake Assist (EBA)	Increases braking pressure up to maximum in an emergency situation (response significantly faster than full driver's foot movement)	Detection of a "panic gas release" and assessment of rate by which the brake is applied by the driver

Table 1.1: A low-level vehicle control systems overview supporting the Advanced Driver Assistance Systems (ADAS) (I).

Low-level vehicle control system	**Function**	**Technical solution**
Variable Gear Ratio Steering (VGRS)	Torque assist to aid the driver's evasive steering measures	Variable steering gear ratios
Adaptive Variable Suspension (AVS)	Collision-avoidance Steering Support	Variable suspension damper firmness
Supplemental Restraint System (SRS) or Air Bag System	Inflatable restraints provide a significant reduction of crash impact on a passanger's body	Frontal, side, side-torso, side-curtain, knee, rear, rear-curtain, seat-belt airbags
Adaptive airbag systems	Multi-stage airbags where the pressure within the airbag is adjusted according to the severity of crash	A control chemical reaction produces a burst of propellant to inflate the bag, optionally deflate and re-inflate in several stages in successive collisions
Pedestrian Protection System	Generic term for structural modifications in vehicle hood, windshield and bumpers to reduce the impact on pedestrians	Deformable and/or softer material hoods and windshields, deeper and lower bumpers, possible pedestrian airbags in the hood
Frontal Protection System (FPS)	Frontal structure to protect pedestrians and cyclists at impact	Pedestrian and cyclist "friendly" mounting structure, primarily replacement for the bull bar in SUVs

Table 1.2: A low-level vehicle control systems overview supporting the Advanced Driver Assistance Systems (ADAS) (II).

ADAS system type	**Function**	**Technical solution**
Head-Up-Display (HUD)	Displays relevant driver information directly on the windshield projection plane. Facilitation of driver's view of the road and display information	Solid-state LED display in front of the windshield
Alcohol Detection System	Prevents a drunk driver from starting the engine	Chemical and pulse rate sensors
Intelligent Headlamp Control (IHC) or Advanced Front Lighting (AFL)	Switching between low- and high intensity light beams based on current traffic	Camera traffic detection systems with advanced headlight technologies
Adaptive Curve Illumination (ACL)	Pointing the vehicle light beams in the tangential direction of curved roads	Camera lane detection systems with advanced headlight technologies
Intelligent Parking Assist System (IPAS) or Advanced Parking Guidance System (APGS)	Assists the parking maneuver	Camera and ultrasonic sensors used to determine parking free space, Electric Power Steering used for steering-in
Night Vision Support	Detects pedestrians, animals and other heat-emmiting objects before they are visible to the human eye in the headlights	Thermal imaging camera
Driver Drowsiness Detection System	Monitoring of driver alertness to the driving task	Eye-closure vision detection system or Brain-wave monitors
Traffic Sign Recognition or Speed Limit Monitoring (SLM)	Detection of traffic signs and alert/notification on a driver display about the speed limits	Camera system with or without an underlying navigation system

Table 1.3: An Advanced Driver Assistance Systems (ADAS) overview (I).

ADAS system type	**Function**	**Technical solution**
Blind Spot Detection (BSD)	Warns the driver if there are other vehicles in the blind spot of the side-view mirror	Camera or lidar system (short range up to 20m)
Lane Change Assist (LCA)	Detects, tracks and informs the driver about oncoming vehicles from the rear. Active if the side indicator light is on	Rear oriented radar (middle range up to 70m)
Lane Departure Warning (LDW)	Detects the lane structure and possible ego-vehicle departure from the nominal lane trajectory giving an acoustic warning. Active if the side indicator light is off	Mono or stereo camera system
Lane Keep Assist (LKA)	Using the information based on Lane Departure Warning provides force-feedback on the steering wheel both as a warning and corrective assistance towards the middle of the lane	Lane Departure Warning and Electric Power Steering
Blind Spot Intervention (BSI) or Lane Change Support (LCS)	Using the information based on Lane Change Assist provides force-feedback on the steering wheel both as a warning and corrective assistance back to the original lane of departure	Lane Change Assist and Electric Power Steering
Adaptive Cruise Control (ACC)	Adaptive speed control of the ego-vehicle in order to follow the flow of traffic ahead	Lidar or radar based vehicle detection possibly coupled with a camera, low-level automatic engine and brake control (driver still required to brake in extreme situations)
Stop-and-Go Assist or City Safety System	A version of the Adaptive Cruise Control for slow traffic (especially traffic jams)	Similar to Adaptive Cruise Control but with automatic stopping

Table 1.4: An Advanced Driver Assistance Systems (ADAS) overview (II).

ADAS system type	**Function**	**Technical solution**
Forward Collision Alert (FCA)	Warning of an imminent collision with the vehicle in front (Pre-crash)	Radar, lidar or camera front vehicle detection with distance and rate-of-approach assessment
Active Brake Assist (ABA) or Collision Mitigation Braking System (CMBS) or Collision Warning with Brake Assist (CWBA)	Detection, warning and mitigation of critical front traffic situations. The Emergency Brake Assist mode is activated first. If the driver does not react within preset conditions, maximum automatic braking is applied in order to avoid or significantly reduce the severity of the crash	Forward Collision Alert detection system with automatic brake control
Follow-To-Stop Assist	An traffic situation adaptive longitudinal velocity system that is capable of full brake control till vehicle stop within a large speed range (typically 0-210 km)	A combination of Adaptive Cruise Control and Active Brake Assist
Pre-Sense Rear	Detection of an imminent collision from rear	Radar sensor
Pre-Crash Assist	Generic term for preparation of the vehicle for imminent collision	Adjustment of passenger seats and optionally active head restraints, tightening of seat-belts, brakes pre-charging, assessment of airbag triggering conditions
Advanced Automatic Collision Notification or eCall	Automatic notification of the eCall emergency center about the location and time of accident, vehicle board computer data	Mobile network device able to connect to different local eCall emergency centers (roaming)

Table 1.5: An Advanced Driver Assistance Systems (ADAS) overview (III).

Potential future additional ADAS functionalities	**Description**
Evasive steering maneuver to the adjacent lanes	In certain critical situations the braking distance to the vehicle in front may be too short to avoid a crash, therefore an alternative evasive steering maneuver may be executed to the left adjacent lane or the safety stop lane to the right if there is no oncoming traffic or other obstacles in the way.
Avoiding rear collision by an acceleration maneuver and/or evasive steering maneuver	Early detection of an approaching vehicle from behind via e.g. Pre-Sense Rear and if estimating the rate-of-approach as critically high, the ego-vehicle may be accelerated in the current lane or perform an evasive maneuver to the adjacent lanes, if there is free space.
Prioritizing collision avoidance with respect to the more vulnerable traffic participants	Some traffic participants such as pedestrians or cyclists are especially vulnerable to any collision. Assuming that due to situational complexity a crash of the ego-vehicle is inevitable, in such a case a maneuver may be chosen to collide with other vehicles (e.g. stopped vehicles in front or relatively slowly approaching from the rear or side), rather than crashing into a pedestrian or a cyclist. The argument for such a behavioral pattern is that there is a high chance that other vehicles will also be equipped with passive/active ADAS which render their passangers much less vulnerable than the fully exposed pedestrians. In total, this might result in a chain collision with a higher material damage, but less severe human injuries, potentially life-saving.
Overtake maneuver	By combining a Follow-To-Stop Assist with an evasive maneuver between adjacent lanes, an overtake maneuver can be executed in nominal traffic situation, compliant also with the Speed Limit Monitoring Assist. This would increase the driver comfort level, especially in highly structured traffic environments such as highways, thereby also increasing the overall traffic flow.

Table 1.6: A list of potential future additional ADAS functionalities.

1.3 Contributions

The overall aim of this thesis was to develop algorithmic methods for the autonomous vehicle navigation in static and dynamic environments, with the vision of increasing the traffic safety, be such automatic systems introduced. The particular contributions of this work can be accounted for as follows.

A 3DOF dynamic Ackermann vehicle model for motion prediction was derived that takes into account the relevant physical forces acting on the vehicle. The geometry and inertia of the vehicle was included in the model as well as the interaction parameters with respect to the ground/road surface based on the contact dynamics. The parameters of the vehicle that are not measurable directly were estimated via an optimization routine. Experimental validation of the model showed that the dynamic physical effects are well within the range of expected values.

The autonomous navigation task in static, slowly changing environments that may be unknown prior to the navigation execution was approached by proposing a hierarchical navigation structure that includes the all the necessary levels of deliberation to complete the task. In order to account for structured as well as unstructured environments, the perceptional information about the environment configuration was described in a form of a generic grid map. At the global navigation level previously developed planning techniques were used which provide global connectivity information at a sample rate that may be asynchronous to sudden or unforeseen changes in the environment. Therefore a reactive collision avoidance level was developed running at a high control cycle in order to avoid collisions and also to account for non-holonomic constraints not taken into account in the global planning cycle. The negotiating intermediate level between the global navigation and reactive avoidance level in form of path following was proposed for the cases were the global connectivity information is in form of an interpolated navigation cost or a geometric global path. The hierarchical navigation scheme was tested successfully on an experimental vehicle in unstructured environments such as parking lots in presence of moving pedestrians.

In the case where the low-level vehicle control parameters may not be known or the expert knowledge of the human driver can be introduced on-the-fly to the automatic vehicle, a probabilistic formulation based on Bayesian reasoning was analyzed for the task of reactive obstacle avoidance case and was tested on the experimental vehicle for the case without parametric system adaptation.

Generic dynamic systems optimization techniques may be used for the motion planning problem where the terminal and path constraints must be properly described for the vehicle motion and environmental obstacles. The case of autonomous vehicle parking is an appealing case in static environments that was studied in this work where the overall optimization objective is minimum parking time while taking into account the vehicle motion constraints. The effect of two different optimization approaches on the quality of motion planning solution was compared.

Vehicle motion planning in dynamic environments requires special attention to the coordination between dynamic object motion prediction, motion planning cycle and motion execution. The timing constraints relating to these cycles were explicitly accounted for. Two distinct motion planning procedures were proposed based on the description of the ego-vehicle motion, namely the transformed and trajectory space based and analyzed in simulated generic dynamic obstacle environment and urban dynamic scenarios. In comparison, the main advantageous feature of the trajectory space approach is that it enables partial motion planning for any type of dynamic ego-platform among dynamic obstacles that are described with arbitrary motion models. The trajectory space partial motion planning was integrated into a hierarchical navigation scheme for autonomous vehicle navigation in urban scenarios where the global mission is defined via a set of navigation waypoints defined according to a topological description of an urban road network. Simulated autonomous navigation results were provided in the presence of dynamic objects such as vehicles and pedestrians for the lane, intersection and parking zone situations. The safety level with respect to the braking capabilities

of the ego-vehicle and that of the dynamic objects was formulated explicitly.

In order to close the loop from motion planning to perception, a dynamic scene analysis was made that includes the localization of the ego-vehicle in 3D that can be based only inertial sensory information if the GPS signal is unavailable. Furthermore, a vision based lane detection module was developed that defines the basic road structure for the ego-vehicle to navigate to as well as a segmentation of the surrounding vehicle environment into static and dynamic object candidates based on temporal occupancy map labelling.

1.4 Structure of this work

According to the problem statement in Sec. 1.1.2 this thesis is divided into following units: Chap. 2 describes the experimental vehicle platform used in navigation schemes validation. Chap. 3 deals with autonomous vehicle navigation in static and unstructured environments, whereas the Chap. 4 analysis the relevant aspects of navigation in presence of dynamic obstacles. Chap. 5 analyzes the perceptional part of a dynamic scene. Chap. 6 is a fusion of a proposed motion planning scheme in dynamic environments of Chap. 4 and perceptional analysis of Chap. 5 applied to the dynamic urban scenario.

Chapter 2

Vehicle platform

2.1 Introduction

Any autonomous navigation strategy includes a form of motion planning, thereby the simulation of the ego-vehicle platform motion, its trajectory prediction, is needed. The vehicle dynamic motion model should include all the relevant vehicle structural aspects as well as its physical interaction with the environment. Preferably, the model should be compact enough so that it can be simulated computationally affordable for different initial conditions, motion commands or workspace configurations. The aim in this chapter is to develop a dynamic model for an Ackermann-like vehicle with relevant parameter identification that emulates well the motion of the experimental platform vehicle.

Sec. 2.2 introduces briefly the vehicle platform used in experimental validation, followed by the vehicle dynamic modeling in Sec. 2.3.

2.2 Vehicle platform overview

Fig. 2.1 shows the experimental vehicle platform SmartTer that was used in the experiments throughout this thesis [Lamon *et al.*, 2006a]. Different sensory setups can interchangeably be used on the SmartTer vehicle.

Figure 2.1: SmartTer experimental vehicle.

The setup relevant to this work is based on the following infrastructure:

- ALASCA XT 4-plane laser with a horizontal range of up to 100 m;
- An outdoor SICK laser mounted sideways;
- A stereo-rig camera unit mounted on rooftop with frontal view;
- A centrally mounted camera inside the vehicle;
- A GPS unit;
- An Inertial Measurement Unit;
- Low level brake actuator and gas pedal control unit controllable via CAN interface;
- A remote safety stop unit;
- A set of rack mounted computer units.

2.3 Vehicle dynamics modeling

In order to develop stabilizing control laws in dynamic vehicle regimes, a vehicle model that takes dynamics into account is developed in this section. The model is suitable for lane-keeping control and obstacle avoidance maneuvers when lateral dynamic effects based on wheel slip come into place, i.e at increased vehicle speeds, steering angles and decreased road friction.

The vehicle dynamic model is developed for an Ackermann-like vehicle based on a static tire-road friction model and laws of technical mechanics. The model takes as input the steering angle of the wheels in front and the rotational velocities of the drive wheels in the back of the vehicle. It delivers a 3-DOF output in terms of CoG vehicle velocity, body slip angle and the yaw rate of the vehicle in the x-y plane, as well as estimates on the forces acting on the system. It is suitable for modeling dynamic vehicle regimes in e.g. overtaking maneuvers/obstacle avoidance and lane-keeping, enabling active steering control by stabilizing the dynamics of the vehicle. The physical model description is based on previous works combined with a suitable friction model that is tractable in practice. Experimental verification of the obtained model is given for the Smart testing vehicle platform, where a separate analysis is done for directly measured as opposed to estimated/optimized parameters of the model [Macek *et al.*, 2007].

A vehicle can be analyzed as consisting of five individual subsystems: one vehicle body and four wheels. All the five bodies can in general move freely with respect to each other in six degrees of freedom. Without further simplification the dynamics of a vehicle model would therefore consist of thirty differential equations. However, for control purposes this is neither necessary nor efficient. The existing models cope with this problem by simplifying the model architecture as much as possible while still satisfying the requirements of the model. The simplest solution found in literature is the one-track bicycle model well described by Mitschke in [Mitschke, 1990]. This model combines the front and rear wheels respectively and treats the vehicle as a bicycle. It describes the vehicle motion in three degrees

of freedom (x-y position and yaw rate). More accurate models can be found in [Junjie *et al.*, 2004] and [Selby *et al.*, 2001] which are four wheel models describing also the pitch and roll movements of the vehicle (five DOF).

For the lane-keeping control, our model is required to generate an accurate prediction of the yaw rate and the longitudinal/lateral acceleration of the vehicle, therefore the calculation of the pitch and roll rate is not necessary and will not be considered here. In consequence, a four wheel three degrees of freedom model is derived, which describes the motion of the car in the x- and y- direction on a horizontal plane and the rotation about the z-axis normal to it.

In Sec. 2.3.1, the architecture of the model is presented, with each segment of the model analyzed in detail. The vehicle model proposed follows closely the derivation of [Kiencke and Nielsen, 2000b] which is general for simulation of behavior of passenger cars. In Sec. 2.3.2, the procedure to measure and estimate the parameters of the model is given. The relevant unknown parameters of the model are identified by an optimization routine. The model/parameter validation is based and validated on real vehicle data, where a good fit and realistic estimation of the dynamic vehicle behavior is observed. The interest of the section can be found mainly in combining an existing systematic vehicle dynamic model derivation with a choice of a friction model that suits well the vehicle behavior tested on real data and is still simple enough for control purposes. Furthermore, a systematic methodology for model parameter measurement and estimation is presented.

2.3.1 Vehicle model

2.3.1.1 Model Architecture

As mentioned earlier, the dynamic model developed here consists of five connected subsystems: one vehicle body and four wheels, which are rigidly coupled to the vehicle body. The model is symmetrical about the vehicle body's longitudinal

axis. The rear wheels, which are driven by a torque, cannot be steered and are therefore aligned with the vehicle body's longitudinal axis. The front wheels can be steered according to the Ackermann steering model and the vehicle motion is constrained to horizontal movements only.

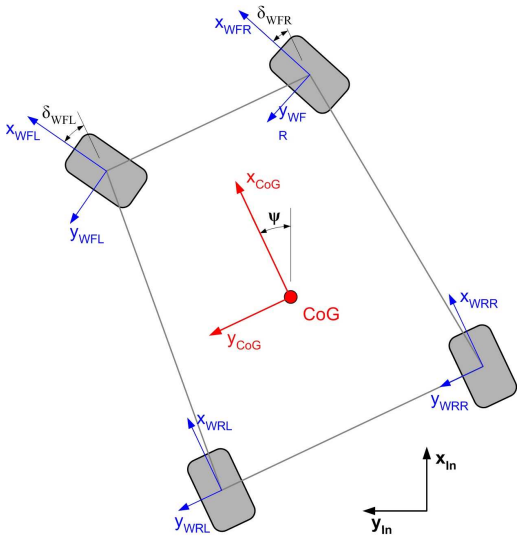


Figure 2.2: Coordinate Systems: *In* - Inertial, *CoG* - Center of Gravity, *W* - Wheel.

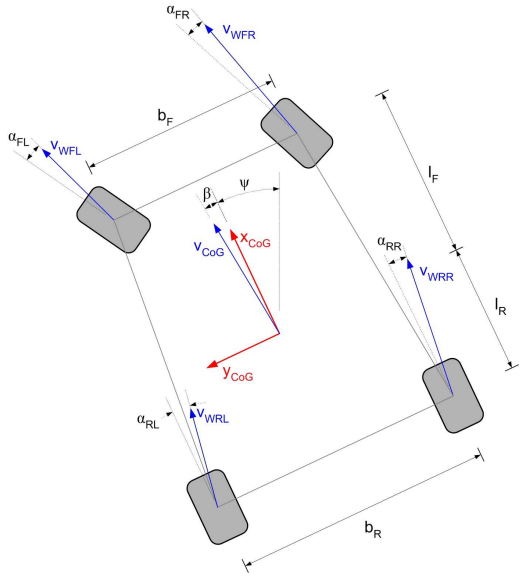


Figure 2.3: Geometry and defined velocities of the vehicle.

As shown in Fig. 2.2 different coordinate systems for all subsystems are defined with respect to an inertial (In) coordinate system, which is defined as a fixed system in which the x- and y- axis describe a horizontal plane and the z-axis points upwards. The center of gravity (CoG) coordinate system is associated with the vehicle body and has its origin in the CoG of the total system (i.e. vehicle body plus the five wheels). Each wheel has its own (W) coordinate system. The wheels always touch the level horizontal plane described by the x_{In}- and y_{In}-axis in one single point (Wheel Ground Contact Points). Relative to the CoG-system the wheels are allowed to rotate freely about their rotational axis y_W. The steering angles $\delta_{W_{FL}}$ and $\delta_{W_{FR}}$ of the front wheels and the driving torque T_{Drive} that acts on both rear wheels are the physical inputs to the vehicle.

Due to the fact that the wheels are rigidly connected to the vehicle body no

suspension is implemented in our model. However, this assumption is acceptable because the suspension forces are only internal forces to the vehicle system and do not have an effect on the horizontal motions (x-, y- position and yaw motion). However, different normal forces on the outside and inside wheels in a turn are still taken into account as proposed in Sec. 2.3.1.4.2.

2.3.1.2 Vehicle kinematics

2.3.1.2.1 Vehicle Body Kinematics The velocity of the CoG in the CoG-system is defined by:

$$\overrightarrow{{}_{c}v_{CoG}} = \begin{bmatrix} v_{CoG} \cdot \cos(\beta) \\ v_{CoG} \cdot \sin(\beta) \end{bmatrix} \tag{2.1}$$

Here v_{CoG} is the absolute value of the velocity and β is the angle between the x-axis of the CoG-system and the velocity vector as shown in Fig. 2.3. β is known as the body side slip angle. Transformation of ${}_{c}v_{CoG}$ to the In-system yields:

$$\begin{aligned} \overrightarrow{{}_{i}v_{CoG}} &= \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \cdot \overrightarrow{{}_{c}v_{CoG}} \\ &= \begin{bmatrix} v_{CoG} \cdot \cos(\beta+\psi) \\ v_{CoG} \cdot \sin(\beta+\psi) \end{bmatrix} \end{aligned} \tag{2.2}$$

where the yaw angle ψ is the angle between the In-system and the CoG-system. The acceleration ${}_{i}a_{CoG}$ of the CoG is derived in the In-system [Kiencke and Nielsen, 2000b](p.261):

$$\begin{aligned} \overrightarrow{{}_{i}a_{CoG}} = v_{CoG}(\dot{\beta}+\dot{\psi}) \begin{bmatrix} -\sin(\beta+\psi) \\ \cos(\beta+\psi) \end{bmatrix} + \\ +\dot{v}_{CoG} \begin{bmatrix} \cos(\beta+\psi) \\ \sin(\beta+\psi) \end{bmatrix} \end{aligned} \tag{2.3}$$

Transforming the ${}_{i}a_{CoG}$ into the CoG-system is done by rotating the vector about ψ around the z_{In}-axis:

$$\overrightarrow{{}_{c}a_{CoG}} = v_{CoG}(\dot{\beta}+\dot{\psi})\begin{bmatrix} -\sin\beta \\ \cos\beta \end{bmatrix} + \dot{v}_{CoG}\begin{bmatrix} \cos\beta \\ \sin\beta \end{bmatrix} \tag{2.4}$$

with additional notation ${}_{c}a_{CoG,x} \equiv a_x$ and ${}_{c}a_{CoG,y} \equiv a_y$.

2.3.1.2.2 Wheel Kinematics The wheel ground contact point velocity is defined as the velocity of the ground contact point of each wheel not taking into account the rotational speed of the wheel [Kiencke and Nielsen, 2000b] (p.226), representing a stationary point with respect to the vehicle body [Williams, 1996](pp.68):

$$\overrightarrow{{}_{c}v_{w_n}} = \overrightarrow{{}_{c}v_{CoG}} + \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} x_n \\ y_n \\ 0 \end{bmatrix} \tag{2.5}$$

The wheel velocities then depend on v_{CoG}, the yaw rate $\dot{\psi}$, and the geometry of the vehicle, where x_n and y_n are the coordinates of the four wheel ground contact points, $n = 1, \ldots, 4$, given in the CoG-system. The geometry of the car is shown in Fig. 2.3. The lengths and widths l_F, l_R, b_R, b_R describe the exact position of each wheel ground contact point with respect to CoG. The four wheel velocities are given by:

$$\overrightarrow{{}_{c}v_{w_{FL,R}}} = \begin{bmatrix} v_{CoG}\cos\beta \mp \dot{\psi}\frac{b_F}{2} \\ v_{CoG}\sin\beta + \dot{\psi}l_F \end{bmatrix} \tag{2.6}$$

$$\overrightarrow{{}_{c}v_{w_{RL,R}}} = \begin{bmatrix} v_{CoG}\cos\beta \mp \dot{\psi}\frac{b_R}{2} \\ v_{CoG}\sin\beta - \dot{\psi}l_R \end{bmatrix} \tag{2.7}$$

The tire side slip angle α of each wheel is defined similar to the vehicle body

side slip angle β. α is the angle between the wheel axis x_{w_n} and the wheel velocity v_{w_n} as shown in Fig. 2.4. The four α's are calculated by:

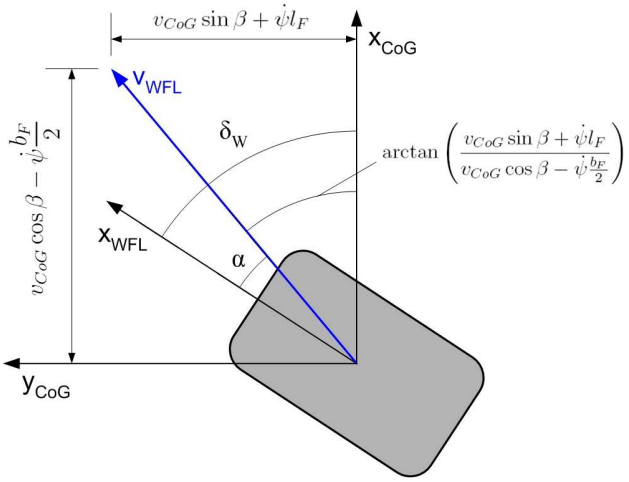


Figure 2.4: Side slip angle front left tire.

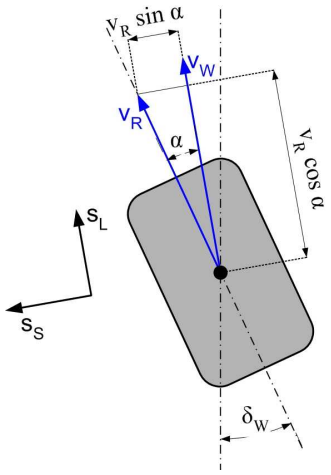


Figure 2.5: Wheel slip.

$$\alpha_{FL,R} = \delta_{W_{FL,R}} - \arctan\frac{v_{CoG}\sin\beta+\dot{\psi}l_F}{v_{CoG}\cos\beta\mp\dot{\psi}\frac{b_F}{2}} \tag{2.8}$$

$$\alpha_{RL,R} = -\arctan\frac{v_{CoG}\sin\beta-\dot{\psi}l_R}{v_{CoG}\cos\beta\mp\dot{\psi}\frac{b_R}{2}} \tag{2.9}$$

The wheel equivalent velocity of each wheel, v_{R_n}, is given by the rotational speed of the wheel ω_n and the effective wheel radius, r_{eff} [Kiencke and Nielsen, 2000b](pp.249-250):

$$v_{R_n} = \omega_n \cdot r_{eff} \tag{2.10}$$

where its direction is the positive x-direction of the wheel coordinate system, describing motion of the wheel if it rolled perfectly on the road.

2.3.1.2.3 Wheel Slip Calculation The three expressions for v_w, α and v_R are now combined in the slip equations. The slip is a normalized description of the tire movement relative to the road and is depicted well by Burckhardt in [Burckhardt, 1993]. This relative movement is solely responsible for the tire friction forces and

is therefore important for the friction forces calculation. The longitudinal slip s_L is defined in the direction of v_w and describes the relative movement in forward direction. The side slip s_S is defined perpendicular to v_w and describes the relative movement in side direction as shown in Fig. 2.5. The two slip equations are given in Tab. 2.11 [Kiencke and Nielsen, 2000b](p.237). Burckhardt differentiates between the driving vehicle and the braking vehicle, thus the slip stays always between -1 and 1.

	Braking $v_R \cos\alpha \leq v_W$	Driving $v_R \cos\alpha > v_W$
Longitudinal slip	$s_L = \frac{v_R \cos\alpha - v_W}{v_W}$	$s_L = \frac{v_R \cos\alpha - v_W}{v_R \cos\alpha}$
Side slip	$s_S = \frac{v_R \sin\alpha}{v_W}$	$s_S = \tan\alpha$

(2.11)

2.3.1.3 Vehicle dynamics

The dynamics of the system can be derived using the principles of linear and angular momentum in two-dimensions for the vehicle body and the four wheels [Williams, 1996](pp.68).

2.3.1.3.1 Vehicle Body Dynamics According to the principle of linear momentum, the acceleration of the car is determined by the sum of all forces ${}_cF_i$ acting upon it:

$$\sum_{i=1}^{k} \overrightarrow{{}_cF_i} = m_{CoG} \cdot \overrightarrow{{}_ca_{CoG}} \tag{2.12}$$

The rotational motion of the wheels is neglected here, thus the only rotation allowed by the model architecture is the yaw motion around the z-axis, with T_i being the torques induced by the forces acting on the car (angular momentum):

$$\sum_{i=1}^{l} T_i = J_z \cdot \ddot{\psi} \tag{2.13}$$

2.3.1.3.2 Wheel Dynamics The torque produced about the y-axis of each individual wheel is the engine torque T_{Drive}, which is for the front wheels equal to zero and for the rear wheels counter-balanced by the longitudinal friction forces F_{WL_n} on each wheel:

$$T_{drive_n} = J_{W_n}\dot{\omega}_n + r_{eff} \cdot F_{WL_n} \tag{2.14}$$

2.3.1.4 Forces description

The most important forces for the dynamics of the vehicle are the friction forces which are generated at the tire/road contact area [Mitschke, 1995](p.17). All the relevant forces for the model acting horizontally upon the system are shown in Fig. 2.6.

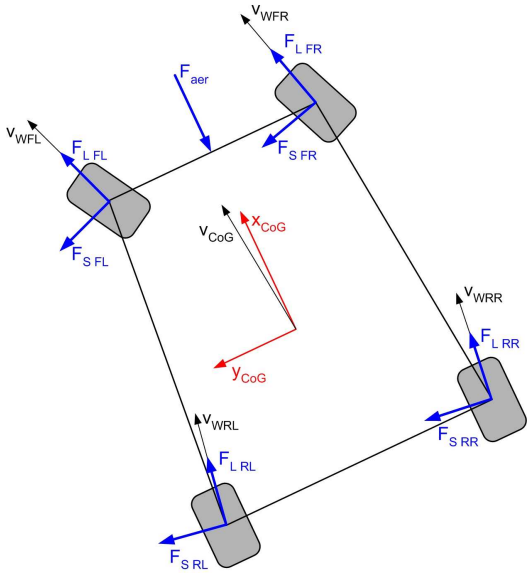


Figure 2.6: Forces: F_L (longitudinal) and F_S (lateral) friction forces, F_{aer} the aerodynamic force.

2.3.1.4.1 Aerodynamic Force The aerodynamic resistance due to the vehicle's motion through the air is the major braking force in the model and is aligned with

the x_{CoG} direction of the car:

$$\overrightarrow{{}_cF_{aer}} = -sgn(v_{CoG}) \cdot c_{aer} A_L \frac{\rho_{Air}}{2} \cdot v_{CoG}^2 \cdot \overrightarrow{{}_ce_x} \tag{2.15}$$

with ρ_{Air} being air density, c_{aer} aerodynamic constant and A_L the effective aerodynamic surface of the vehicle.

2.3.1.4.2 Normal Forces The normal forces are calculated at each wheel ground contact point in the positive z_w direction. Due to the asymmetry of the vehicle body along the vehicle body's lateral axis the normal forces differ already with no motion of the car and become larger at the rear wheels while accelerating and larger at the outside wheels while in a turn. The approach of [Kiencke and Nielsen, 2000b] (pp.306-308) takes into account the shifting of the wheel load while accelerating or driving in a turn.

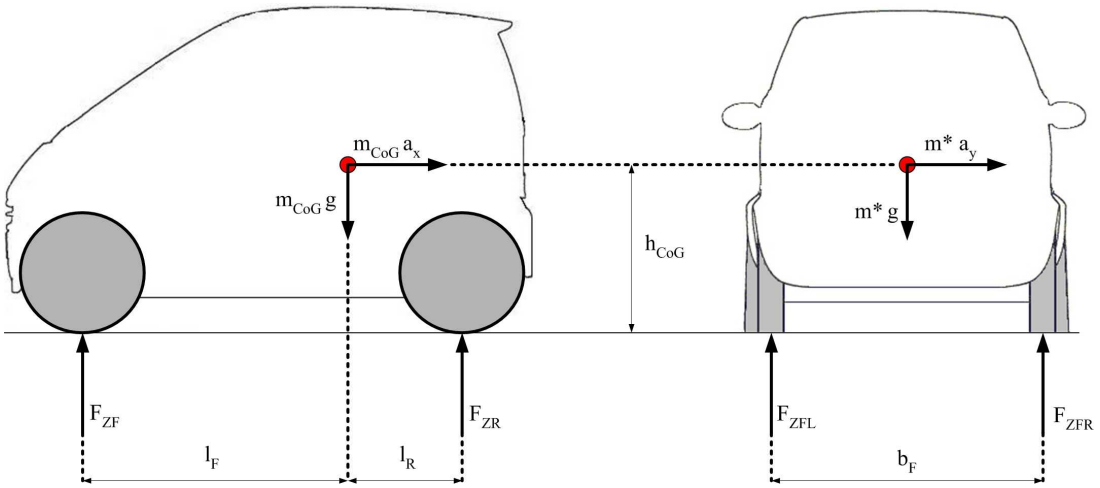


Figure 2.7: Normal Forces Calculation.

Assuming no suspension and roll or pitch motion, the dependencies of the normal forces on the longitudinal and lateral accelerations, a_x and a_y, can be calculated separately. By balancing different loads on axles due to longitudinal acceleration and furthermore axle load shift due to driving in a turn (see Fig. 2.7),

the torque balance at the front left contact point yields:

$$F_{ZFR} = \frac{1}{2} m_{CoG} \cdot \left(\frac{l_R}{l} g - \frac{h_{CoG}}{l} a_x \right) + \\ + m_{CoG} \cdot \left(\frac{l_R}{l} g - \frac{h_{CoG}}{l} a_x \right) \cdot \frac{h_{CoG} \cdot a_y}{b_F \cdot g} \tag{2.16}$$

where similar expressions can be derived for other normal forces as well.

2.3.1.4.3 Friction Forces The behavior of friction at the tire/road contact area is a highly non-linear phenomenon which is complex to describe. Several approaches to the friction characteristics have been developed. Among these are dynamic friction models of which the LuGre model is among the most promising [Canudas de Wit *et al.*, 1995], [Claeys *et al.*, 2001], [Villella, 2004]. In contrast, static friction models empirically approximate the tire characteristics and are well studied. Their output are two forces for each wheel which act upon the wheel ground contact point in the direction of v_W, F_L - the longitudinal friction force and perpendicular to it, F_S - the lateral friction force.

2.3.1.4.4 Burckhardt Friction Model The static friction model that is used in our system was proposed by Burckhardt in [Burckhardt, 1993]. The model aims at obtaining a realistic relationship between the slip of the tires and the friction coefficients in longitudinal and lateral direction μ_L and μ_S. Two auxiliary parameters s_{Res} and μ_{Res} are introduced:

$$s_{Res} = \sqrt{s_L^2 + s_S^2} \tag{2.17}$$

$$\mu_{Res} = c_1 \cdot \left(1 - e^{-c_2 \cdot s_{Res}}\right) - c_3 \cdot s_{Res} \tag{2.18}$$

with calculation of the slips s_L and s_S as explained in Sec. 2.3.1.2.3.

By choosing values for the coefficients c_1, c_2 and c_3 according to different

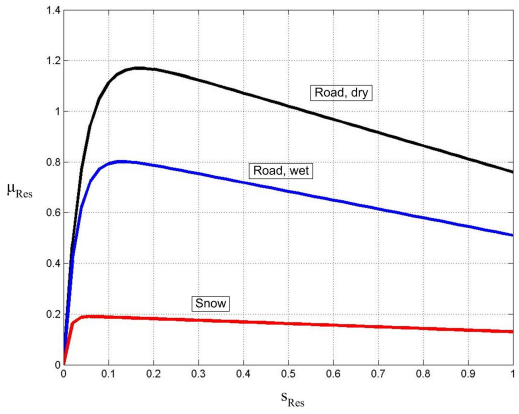


Figure 2.8: Friction coefficients for different ground surface conditions.

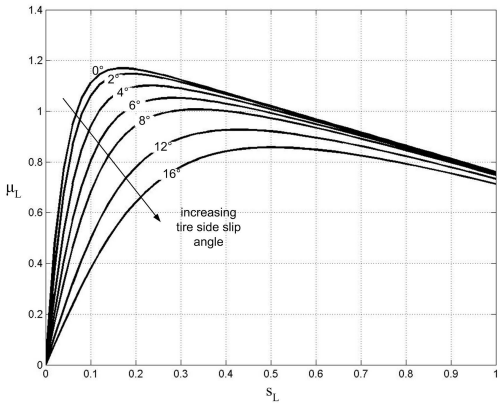


Figure 2.9: μ_L dependency on slip s_L and different α side slip angles.

conditions for dry asphalt, wet asphalt and snow, the Fig. 2.8 shows the resulting friction coefficient as given by Eq. 2.18. As one can see in Fig. 2.9, Eq. 2.18 also provides the dependency of the friction coefficients on the tire side slip angles α.

Now the longitudinal and lateral friction coefficient μ_L and μ_S are calculated by Eq. 2.19 and the longitudinal and lateral friction forces F_L and F_S are given in Eq. 2.20 and Eq. 2.21.

$$\mu_L = \mu_{Res}\frac{s_L}{s_{Res}} \quad , \quad \mu_S = \mu_{Res}\frac{s_S}{s_{Res}} \tag{2.19}$$

with the longitudinal and lateral friction force being:

$$F_L = \mu_L \cdot F_Z = \mu_{Res} \cdot \frac{s_L}{s_{Res}} \cdot F_Z \tag{2.20}$$

$$F_S = \mu_S \cdot F_Z = \mu_{Res} \cdot \frac{s_S}{s_{Res}} \cdot F_Z \tag{2.21}$$

2.3.1.5 Steering

This section deals with the relationship between the angle on the steering column δ_S and the two wheel steering angles $\delta_{W_{FL}}$ and $\delta_{W_{FR}}$ on the front wheels.

2.3.1.5.1 Steering Column Model For the derivation of the relationship between δ_S, the steering column angle and δ_W, the steering angle of the hypothetical wheel in the front center, the model used is a static proportional relationship, where i_T is the transmission coefficient of [Reimpel and Betzler, 2000](pp.224).

$$\delta_W = \frac{\delta_S}{i_T} \tag{2.22}$$

2.3.1.5.2 Ackermann Steering Since all y-axes of the wheels should intersect in the instantaneous center of motion (ICM), the left and right front wheel angles, $\delta_{W_{FL}}$ and $\delta_{W_{FR}}$ are a function of δ_W:

$$\delta_{W_{FL,R}} = \arctan\left(\frac{l}{\frac{l}{\tan\delta_W} \mp \frac{b_F}{2}}\right) \tag{2.23}$$

2.3.1.6 Summary of the Model

The complete vehicle model combines individual segments elaborated so far. The steering and torque as inputs are propagated through the model, generating forces that produce following outputs: the CoG-velocity v_{CoG}, the vehicle body side slip angle β and the yaw rate $\dot{\psi}$ as shown in the equations of motion:

$$\begin{aligned} \dot{v}_{CoG} &= \frac{\cos\beta}{m_{CoG}}\left[\Sigma F_X - c_{aer}A_L\frac{\rho}{2}\cdot v_{CoG}^2\right] \\ &\quad + \frac{\sin\beta}{m_{CoG}}\left[\Sigma F_Y\right] \end{aligned} \tag{2.24}$$

$$\begin{aligned} \dot{\beta} &= \frac{\cos\beta}{m_{CoG}\cdot v_{CoG}}\left[\Sigma F_Y\right] \\ &\quad - \frac{\sin\beta}{m_{CoG}\cdot v_{CoG}}\left[\Sigma F_X - c_{aer}A_L\frac{\rho}{2}\cdot v_{CoG}^2\right] - \dot{\psi} \end{aligned} \tag{2.25}$$

$$\begin{aligned} J_Z\ddot{\psi} &= [F_{YFL} + F_{YFR}]\cdot l_F - [F_{YRL} + F_{YRR}]\cdot l_R \\ &\quad + [F_{XFR} - F_{XFL}]\cdot b_F + [F_{XRR} - F_{XRL}]\cdot b_R \end{aligned} \tag{2.26}$$

where the sum of forces along each coordinate are:

$$\Sigma F_X = F_{XFL} + F_{XFR} + F_{XRL} + F_{XRR} \quad (2.27)$$

$$\Sigma F_Y = F_{YFL} + F_{YFR} + F_{YRL} + F_{YRR} \quad (2.28)$$

2.3.2 Parameter estimation and experimental model validation

2.3.2.1 Acquisition of Measurement Data

From the CAN-Bus (Controller Area Network) that is included serially in the Smart vehicle, the following data can be obtained:

- δ_S - steering wheel angle
- w_{FL}, w_{FR} - rotational wheel speed front left/right
- w_{RL}, w_{RR}- rotational wheel speed rear left/right

The IMU300CC-100 is an external inertial measurement unit supplementary integrated in the test vehicle with following measurements taken:

- $\dot{\psi}$ - yaw rate (range $100°/s$, bias $< \pm 2.0°/s$, scale factor accuracy $< 1\%$)
- a_x, a_y - acceleration in the x- and y- direction (range $\pm 2g$, bias $< \pm 30mg$, scale factor accuracy $< 1\%$).

Both sensor systems take measurements at a sample rate of T_s=11ms.

2.3.2.2 Parameter Identification

2.3.2.2.1 Assured Parameters In this category of parameters, all natural constants and tabulated values are considered, as well as the car geometry which is given by the drivers manual.

The coefficients of the friction model given in Sec. 2.3.1.4.4 are taken for a dry asphalt road, i.e. $c_1 = 1.2801$, $c_2 = 23.99$ and $c_3 = 0.52$.

2.3.2.2.2 Measured Parameters

2.3.2.2.2.1 Effective Wheel Radius r_{eff} is a function of the two static wheel radii r_0 and r_{stat}, where r_0 is the radius of the unloaded wheel and r_{stat} is the compressed wheel radius as the vehicle stands on its four wheels [Kiencke and Nielsen, 2000b](pp.250):

$$r_{eff} = r_0 \cdot \frac{\sin(\arccos(\frac{r_{stat}}{r_0}))}{\arccos(\frac{r_{stat}}{r_0})} \tag{2.29}$$

which for Smart vehicle yields $r_{eff} = 0.273m$.

2.3.2.2.2.2 Total Mass of the Smart and x-Position of the CoG The vehicle was weighted on an industrial scale with an accuracy of 10kg. To find out the x-position of the CoG described by the two lengths l_R and l_F, the law of lever was used:

$$l_F = l \cdot \frac{m_{rear}}{m_{rear} + m_{front}} \ , \ l_R = l \cdot \frac{m_{front}}{m_{rear} + m_{front}} \tag{2.30}$$

where the results also depend on the number of passengers. For the case of no load: $m = 760kg$, $m_{front} = 330kg$, $m_{rear} = 430kg$, $l_F = 1.025m$, $l_R = 0.787$.

2.3.2.2.3 Estimated Parameters

2.3.2.2.3.1 Transmission Coefficient - i_{Trans} The transmission coefficient is important for the car behavior because it determines the magnitude of the wheel steering angles. The initial estimate was i_{Trans}= 15, which is in the range given for other vehicles by [Reimpel and Betzler, 2000](pp.224) and was later optimized.

2.3.2.2.3.2 Inertial Momenta of the Vehicle Body and the Wheels To estimate the value of the inertial moment the formula [Reimpel and Betzler, 2000] (p.406) was used:

$$J_z = 0.1269 \cdot L_T \cdot l \tag{2.31}$$

where L_T is the total length of the vehicle and l the length between the front and rear axis. This formula gives a result for J_z of about $500kgm^2$.

For the estimation of the inertial momenta of the front wheels, the assumption is that the wheel is a solid disk of radius r=20cm and a mass of m=5kg:

$$J_{W_F} = \frac{1}{2} m \cdot r^2 \tag{2.32}$$

which yields the inertial momentum of a single front wheel as J_{W_F}=$0.1kgm^2$.

2.3.2.2.4 Optimization of the Parameters

To optimize the estimated parameters the difference between a set of experimentally measured and modeled outputs is compared. An error function is defined as the sum of all normalized, weighted and squared differences between the five measured and modeled variables:

$$err = \frac{1}{n} \begin{bmatrix} \left(1 \cdot \frac{1}{\llcorner \dot{\psi} \urcorner}\right)^2 \\ \left(0.5 \cdot \frac{1}{\llcorner a_x \urcorner}\right)^2 \\ \left(0.5 \cdot \frac{1}{\llcorner a_y \urcorner}\right)^2 \\ \left(0.5 \cdot \frac{1}{\llcorner \omega_{FL} \urcorner}\right)^2 \\ \left(0.5 \cdot \frac{1}{\llcorner \omega_{FR} \urcorner}\right)^2 \end{bmatrix} \cdot \sum_{i=1}^{n} \begin{bmatrix} (\Delta_i(\dot{\psi}))^2 \\ (\Delta_i(a_x))^2 \\ (\Delta_i(a_y))^2 \\ (\Delta_i(\omega_{FL}))^2 \\ (\Delta_i(\omega_{FR}))^2 \end{bmatrix} \tag{2.33}$$

where the difference for each individual measurement is:

$$\Delta_i(\circ) = (\circ)_{measured}\, i - (\circ)_{modeled}\, i \tag{2.34}$$

The error of the yaw rate $\dot{\psi}$ is weighted twice as high as the errors of the four other variables because of its importance and the quality of the measurements. The data is normalized with the difference between the maximum and minimum value of each measured parameter over the whole test drive ($\llcorner \dot{\psi} \urcorner$, $\llcorner a_x \urcorner$, $\llcorner a_y \urcorner$, $\llcorner \omega_{FL} \urcorner$, $\llcorner \omega_{FR} \urcorner$). The Matlab^{TM} function *fminsearch* is applied in order to search for the minimum of the error function. The optimized parameter values were

derived as:

$$i_{Trans} = 28.5576 \; , J_z = 1490.3 kgm^2 \; , J_{W_F} = 0.1071 kgm^2$$

which is in good correspondence with the Smart vehicle used, since smaller vehicles types have typically a higher transmission coefficient.

2.3.2.3 Experimental Model Validation

After identifying the unknown parameters of the model and optimizing it with measured data, the model is validated by comparing it to a new test drive of the Smart vehicle. It is expected that the vehicle model follows the measured data without any further optimization. The inputs to the model are shown in Fig. 2.10, 2.11 and 2.12, as front steering wheel angle δ_W (with the estimated steering angles on left and right front wheel δ_{WL}, δ_{WR}) and the rear rotational wheel velocities ω_{RL}, ω_{RR}, respectively. Peak differences in rear wheel velocities indicate that the driving regime is dynamic. Note that the wheel dynamics is induced with the drive torques on rear axis as explained in Sec. 2.3.1.3.2, however, the torque quantities cannot be measured directly. They can only be estimated in a forward manner by an additional model of vehicle engine and transmission or in a backward manner according to Eq. 2.14, therefore the rear wheel velocities are considered as direct inputs here and represent the driving quantities.

In Fig. 2.15, 2.16 and 2.17 the yaw rate $\dot{\psi}$ and accelerations a_x and a_y are given as a comparison between the measured and modeled dynamic variables, since they can be directly measured by the IMU unit. For the given precision of the IMU, the model errors are: $|\Delta_{max}|_{\dot{\psi}} = 6.7°/s$, $\sigma_{\dot{\psi}} = 2.3°/s$, $|\Delta_{max}|_{a_x} = 0.71 m/s^2$, $\sigma_{a_x} = 0.33 m/s^2$ and $|\Delta_{max}|_{a_y} = 1.42 m/s^2$, $\sigma_{a_y} = 0.74 m/s^2$. As can be seen, the measured and modeled data correspond well, with maximal model deviations $|\Delta_{max}|$ due to peaks of highly dynamic regime. The estimated forces on the front left wheel of Fig. 2.13 show that the lateral force $F_{S_{FL}}$ is considerable and changes

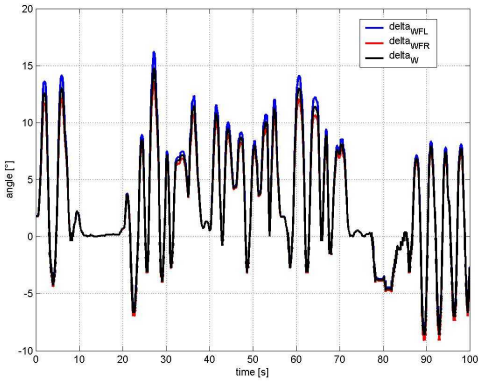


Figure 2.10: Measured input steering angle δ_W and estimated left and right front wheel angles δ_{FL} and δ_{FR}.

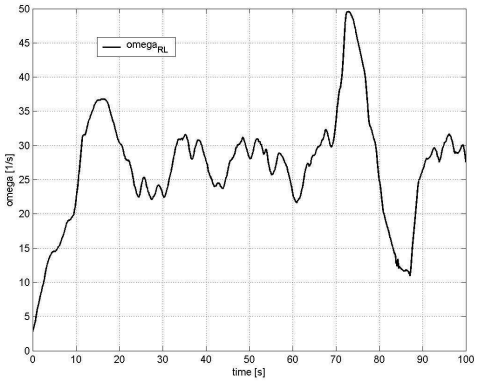


Figure 2.11: Measured input wheel velocity rear left ω_{RL}.

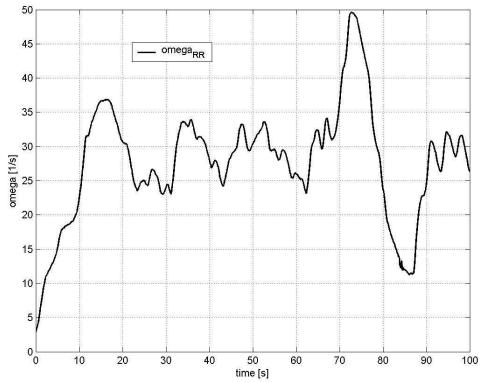


Figure 2.12: Measured input wheel velocity rear right ω_{RR}.

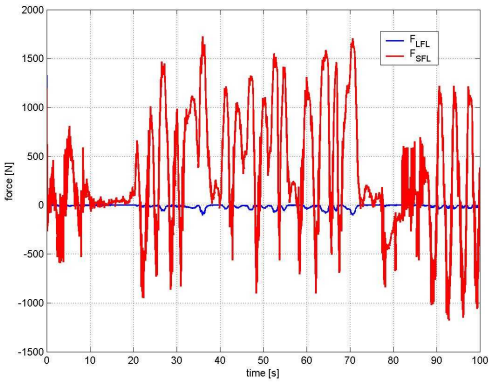


Figure 2.13: Estimated wheel forces front left, longitudinal $F_{L_{FL}}$ and lateral $F_{S_{FL}}$.

direction according to the steering angle. The longitudinal force $F_{L_{FL}}$ at this wheel is small negative or close to zero due to the small wheel inertia and net rolling resistance, opposing the longitudinal motion. The estimated longitudinal force of the rear right wheel $F_{L_{RR}}$ of Fig. 2.14 is in contrast to $F_{L_{FL}}$ much bigger since it represents one of the two driving forces exerted on the vehicle by the engine and transmission, changing from the acceleration phases (positive) to the braking phases (negative). The overall magnitudes of the forces also correspond well to those found in other literature [Kiencke and Nielsen, 2005], [Mitschke, 1990] taking

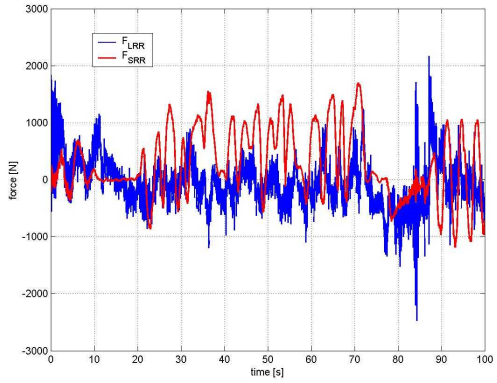


Figure 2.14: Estimated wheel forces rear right, longitudinal $F_{L_{RR}}$ and lateral $F_{S_{RR}}$.

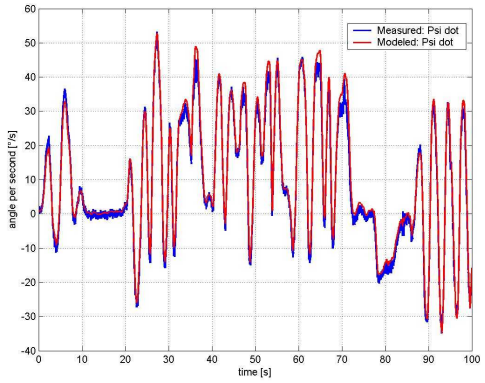


Figure 2.15: Estimated and measured yaw rate $\dot{\psi}$.

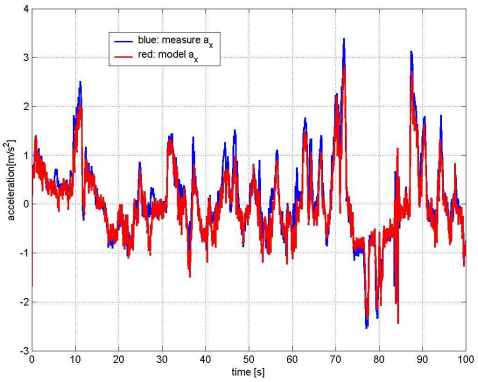


Figure 2.16: Estimated and measured acceleration in the x-direction a_x.

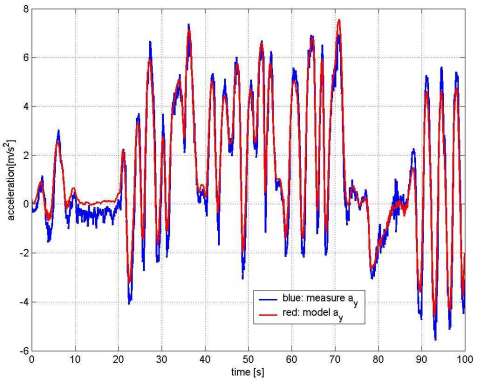


Figure 2.17: Estimated and measured acceleration in the y-direction a_y.

into account the scale of the car class. The developed dynamic model will be used for feed-forward trajectory simulation within a control scheme which relies also on the on-line feedback action to account for model inaccuracies.

2.4 Conclusion

Sec. 2.2 provided a brief overview of the experimental vehicle platform Smart-Ter followed by the Sec. 2.3 were a 3DOF dynamic vehicle model was developed describing the motion of vehicle in horizontal plane by considering lateral and

longitudinal body dynamics and friction forces exerted by the ground surface acting on the vehicle. A detailed analysis included identification of all the relevant parameters of the model, either directly measured or estimated. An experimental verification based on comparison of the model output and measured data of a test vehicle proved a good correlation and validity of the model. The model can be used for example for lane-keeping and obstacle avoidance control of the vehicle. In particular, the dynamics of the vehicle must be taken into account at higher speeds, i.e. in excess of $30km/h$ and on surfaces with small friction adhesion, e.g. wet or icy ground surface. If additional information on the type of ground surface is provided via a detection system (e.g. vision, acoustic) an automatic switching between different friction models could also be performed.

Chapter 3

Motion planning in static environments

3.1 Introduction

This chapter deals with navigation techniques suitable for autonomous vehicle navigation in static, slowly changing and unstructured environments. Typical navigation environments in this category would be parking lots in urban areas or off-road terrains, however only the case of urban areas is further analyzed here.

The Sec. 3.2 presents a hierarchical navigation scheme for static environments that is both complete in the sense of global objective and reactive in order to account for previously unseen, partially known or changing obstacle configurations in the environment.

A probabilistic obstacle avoidance module is developed in Sec. 3.3 that has the potential of straightforward inclusion of expert knowledge of a human driver.

The Sec. 3.4 analyzes the problem of time optimal navigation applied to the case of vehicle parking maneuver using two different constraint optimization tools.

3.2 Hierarchical navigation

The aim of this section is to develop a navigation strategy for an environment without identifiable or clearly present structure in the presence of slowly moving obstacles, i.e. without explicit motion modeling (a typical example are pedestrians). A suitable description of such a surrounding is an occupancy based representation of occupied/free/unexplored areas. To compute a global path from a start to a goal position, the obstacle areas are cast into the configuration space, thereby increased by the contour of the ego-vehicle itself. By this procedure, the ego-vehicle is reduced to a point in a grid for which any of the grid based global planning algorithms, such as A* [Hart *et al.*, 1968], D* [Stentz, 1994], FD* [Ferguson and Stentz, 2005] or E* [Philippsen, 2006] can be applied. The A* and E* planners accumulate previously unseen/changing structure of the environment in a buffer and launch a global re-planning phase after significant new information have been accumulated, whereas the D* and its interpolated version FD* planner deal with the environmental changes in an incremental way, thus using a true global replanning phase only at bootstrap.

The geometric path generated in the global planning phase has three major disadvantages:

- it does not comply to the kinodynamic constraints of the vehicle platform;
- the configuration space obstacle enhancement for all possible vehicle orientations is only approximative;
- its refresh (replanning) rate is too slow for sudden changes in the environment.

In order to resolve these problems a reactive navigation level has to be introduced. Its servo cycle rate is high enough to be able to react/avoid sudden changes in the environment, such as moving obstacles or previously unseen structure in the global environment representation, which requires an independent environment

representation, typically in form of a local grid map centered on the ego-vehicle's current position. It uses a vehicle motion model based on vehicle's dynamics modeling that includes the relevant kinodynamic constraints to project potential trajectory candidates for the next motion command to be issued. Trajectory (thereby motion commands) are checked against potential collision with obstacles in the workspace using the exact vehicle geometry (not the configuration space approximation). The motion commands that would end in collision are inhibited. Furthermore, a path following level is necessary to steer the vehicle towards the goal objective, serving as a connecting level between purely deliberative (global) and purely reactive (local) level of navigation.

A hierarchical navigation structure that includes the global planning, path following and reactive obstacle avoidance level is to be proposed and synchronized in different refresh cycles with respect to planning, control and environmental update.

3.2.1 Navigation architecture

Fig. 3.1 shows the proposed overall scheme for the navigation system of autonomous vehicles. The Goal Manager (GM) handles the global scenario of the vehicle, which is described by a set of goals/waypoints to pass through. The global planner (GP) receives the occupancy information, i.e. occupied areas of the environment, with the map anchor (current vehicle pose) from the Mapper module (MP) and dynamically replans the global path on-line. The versions of GP which are employed in our system and are able to handle dynamic environment changes are Field-$D^\star$ [Ferguson and Stentz, 2005] and $E^\star$ [Philippsen and Siegwart, 2005]. The current goal/waypoint is valid until the vehicle reaches a predefined vicinity around the goal, whereupon the next goal is taken and the GP restarted.

The dynamically changing global geometric path is computed at each control cycle and passed on to the Path Smoother/Optimizer (PSO) module, which in its basic version interpolates a smooth continuous path through the geometric path

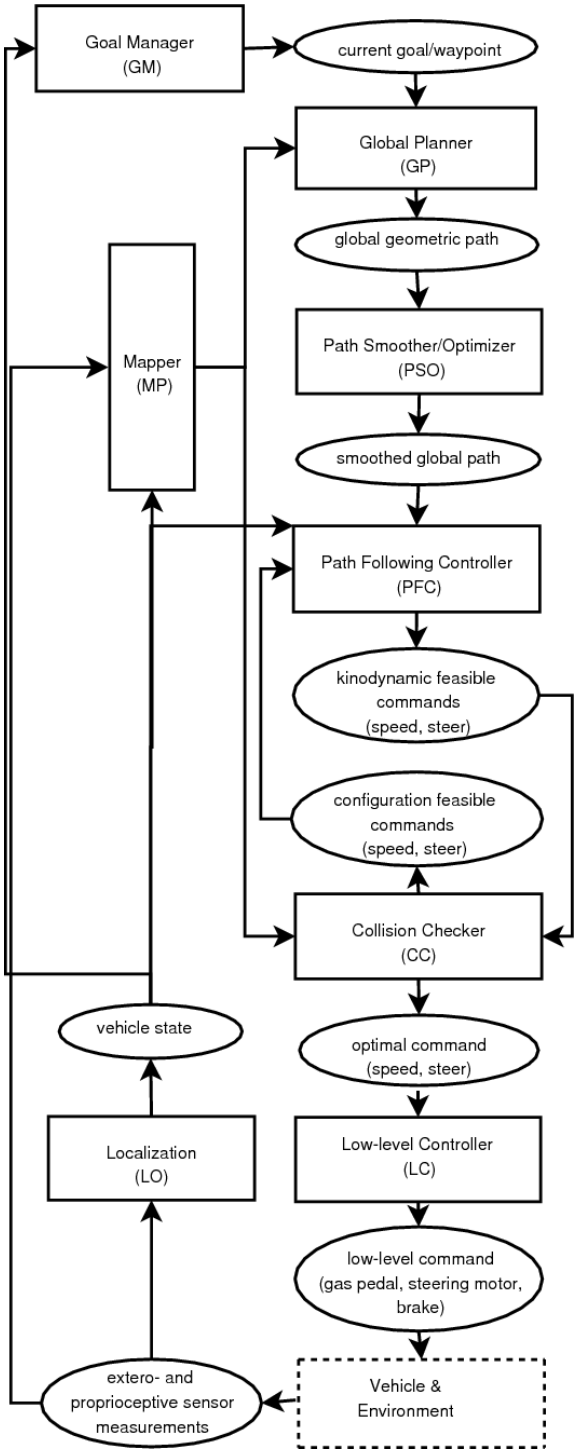


Figure 3.1: Hierarchical navigation architecture.

vertices, using cubic splines ([Bartels *et al.*, 1987]). This module can further optimize thus obtained smooth path by minimizing the curvature change (jerk) along

the path, which can improve both driving comfort and quality of path following control.

The Path Following Controller (PFC) is employed to follow the smoothed global path. The path to follow can also be a holonomic one. The PFC uses the kinematic model with dynamic limitations of the vehicle, such as longitudinal acceleration and steering rate limits, to generate a set of kinodynamically feasible commands for each state of the vehicle, described by the pose $\{x, y, \theta\}$ and the kinematic state $\{v, \phi\}$, v and ϕ being the longitudinal velocity and the steering angle, respectively. Each kinodynamically feasible command pair within the set $\{v_i, \psi_j; i = 1, \ldots, N_v, j = 1, \ldots, N_\kappa\}$ at represents a possible circular trajectory, if the commands are kept constant from the current control cycle onwards. Therefore, each possible trajectory is checked for collision against obstacles in the Collision Checker (CC) module, i.e. configuration space feasible trajectories. The CC module receives obstacle occupancy data directly from the MP module on-line. This renders the CC module independent of GP and ensures safe navigation even in situation when the GP is in a replanning phase and a collision-free global path is not available. From the set of kinodynamically and configuration space feasible commands/trajectories, the optimal command $v^\star, \phi^\star$ taken is the one that ensures path following (steering) and comfort driving by limiting the lateral acceleration (longitudinal velocity profile).

The problem of path following in this architecture is analyzed with particular emphasis on integrating the global path planning, path following and a collision avoidance scheme in a unified framework. They assume existence of a navigation function that generates an on-line path towards a global objective and optionally also generates traversability cost estimates for changing environments with limited vehicle's sensor range (the GP module of the navigation architecture). Whereas the traditional path following algorithms aim at minimizing an error function with respect to a given path and kinematic and/or dynamic model of the robot, the problem of collision avoidance is often neglected or simply cast to the replanning

phase of the global planner which issues the given path. Such approaches that do not check explicitly for collision for the given state of the ego-robot and the environment can easily lead to hazardous situations, in particular if latencies are present in the global path planning phase. In order to address obstacle avoidance directly, a navigation framework is presented here that combines a path following control scheme to attain a global objective with a collision checking scheme that incrementally builds collision-free trajectories, thus ensuring ego-robot safety at all times, with respect to the partially known static environment obstacles and kinodynamic limitations of the ego-robot itself. Two novel path following schemes are presented. Firstly, the "Traversability-Anchored Dynamic Path Following" (TADPF) [Macek *et al.*, 2008a] that was a follow-up development of a reference point path following technique in [Macek *et al.*, 2005]. Secondly, a combined KinoDynamic Sliding Mode Path Following (KDSMPF), based on a previously developed SMPF technique [Solea and Nunes, 2006]. Both the TADPF and KDSMPF path following technique enabling collision-free navigation along the global path with directly taking into account the dynamic limits of the vehicle as well as specifics of the information provided by different global planner module types.

The low-level controller (LC) converts the kinematic level commands $v^\star, \phi^\star$ to the actuator signals on the gas pedal, brake (if in deceleration phase) and steering motor reference. The longitudinal velocity controller is based on a Fuzzy logic controller, which handles also the acceleration/deceleration/cruising phases. The steering column controller is a PID implementation of power steering reference.

3.2.2 Feasible trajectories

The first step towards autonomous vehicle navigation is defining the set of feasible trajectories to pursue. In order to achieve reactive level of navigation, kinodynamically and configuration space feasible trajectories must be defined.

3.2.2.1 Kinodynamically feasible trajectories

Kinodynamically feasible trajectories depend directly on the vehicle/robot motion model, which is for this purpose assumed to be the Ackermann kinematic model:

$$\dot{x} = \cos\theta\, v_l\,, \qquad \dot{y} = \sin\theta v_l\,, \qquad \dot{\theta} = \frac{v_l}{L}\tan\phi\,, \tag{3.1}$$

with $\{x, y, \theta\}$ being the robot pose and $\{v_l, \phi\}$ the longitudinal velocity and steering angle as control inputs and L the axes distance of the front and rear wheels.

According to the Ackermann kinematics, the vehicle follows a circular path for a given kinematic level control input $\{v_l, \phi\}$. Therefore a set of arc vehicle trajectories can be defined as:

$$\mathcal{A} = \{a_{i,j} = \{x_{i,j}, y_{i,j}\}\,; i = 1 \ldots N_v, j = 1 \ldots N_\kappa\}\ , \tag{3.2}$$

where N_{v_l} denotes the number of arc sets due to longitudinal velocity $v_{l,i}$ discretization and N_κ the number of arcs due to curvature κ_j discretization, which corresponds to a steering angle ϕ_j.

At each control cycle a trajectory $a_{i,j}$ is chosen, corresponding to a control input $(v_{l,i}, \phi_j)$ that is feasible with respect to the environment constraints, e.g. obstacles and goal direction, but also according to the limitations on the vehicle motion itself. The kinematic limitations on the vehicle motion are the maximum longitudinal velocity $v_{l,max}$, the minimum allowed[1] vehicle speed $v_{l,min}$, and the maximum steering angle ϕ_{max}. The dynamic limitations are the maximum longitudinal acceleration $\dot{v}_{l,max}$ and the maximum steering rate $\dot{\phi}_{max}$. The aim here is to define a minimum set of arcs necessary to take into account the dynamic limitations of the vehicle at each time instant. From Eq. 4.36 it follows that:

$$\ddot{\theta} = \frac{v_l}{L\cos^2\phi}\dot{\phi} + \frac{\dot{v}_l}{L}\tan\phi\,. \tag{3.3}$$

[1] unless the goal is reached or during an emergency brake

Typically, the low-level steering control loop (e.g. power steering) is faster than the longitudinal velocity control loop, thus for small time increments the longitudinal velocity can be considered constant with respect to the angular rate of the vehicle. Therefore, the second term can be neglected in Eq.3.3. Given that the vehicle is currently on a trajectory defined by $\{v_l, \phi\}$ and the steering rate is at its maximum $\dot{\phi} = \dot{\phi}_{max}$, the curvature change within the kinematic level control sample time-step T_s can be expressed as:

$$\Delta\kappa\left(\phi\right) = \frac{v_l}{L\cos^2\phi}\dot{\phi}_{max}T_s\,. \tag{3.4}$$

Taking the smallest curvature change within a control cycle T_s such that switching between neighboring arcs is feasible according to steering limitations leads to the a-priori number of arcs that can be chosen at any given control cycle.

Although the combined set of arcs according to limitations of Eq. 3.4 could be used for steering control, it is desired to impose additional constraints on longitudinal velocity due to the comfort of the drive, which is related to the maximum lateral acceleration along a given trajectory. In the perfect path following case, the vehicle's curvature would be equal to that of the path, so the maximum lateral acceleration for each $v_{l,i}$ is defined as:

$$a_{L,i,max} = \kappa v_{l,i}^2 \leq a_{L,max} \tag{3.5}$$

assuming constant movement along the path with curvature κ on the interval duration T_s.

Note that even though the dynamic and kinematic constraints of the vehicle motion are taken into account, surface contact dynamics is not modeled here, i.e. there is no longitudinal or side slip of the wheels, under assumption of a surface with high adhesion, such as asphalt. Nevertheless, the vehicle body acceleration limit is still taken into account in $\dot{v}_{l,max}$.

3.2.2.2 Configuration space feasible vehicle trajectories

Given a set of kinodynamically feasible trajectories it is further necessary to check this set against potential collision with obstacles. Each obstacle whether pedestrian, vehicle or other generic structures in the environment can be described in workspace of the vehicle as:

$$\mathcal{O}(t) = \{p(t) \mid p(t) = p_1(t) \dots p_n(t)\} \tag{3.6}$$

where $p_1(t) \dots p_n(t)$ denote the vertices of the polygon the object describes. The polygon $\mathcal{O}$ can be a grid cell object of a traversability map given by the Mapper module of Sec. 3.2.1 or a fully tracked objects such as vehicle and pedestrians of Chap. 5, where the important distinction is made between objects whose workspace configuration does not change in time i.e. $\mathcal{O}(t_c) = \ldots = \mathcal{O}(t_c + T_h)$ and moving objects whose trajectory is described as $\mathcal{Q}(\mathcal{O}t)$ in time t. Here t_c denotes current planning instant and T_h the total prediction horizon beyond which the movement is not considered from the current t_c. The objects in this Chapter are assumed static, or "quasi-static" with respect to the replanning cycle of the navigation scheme.

The notion of safety as considered in this work is defined by the braking capability of the vehicle given the current kinematic state v_l, ϕ and the set of kinodynamically feasible trajectories/arcs $a_{i,j}$. Indeed, for each such arc the brake time is defined as:

$$T_{b,i} = \frac{v_{l,i}}{\dot{v}_{l,max}} \tag{3.7}$$

for which the vehicle is able to come to a complete stop for the given kinematic state v_l, ϕ. Assuming that the motion of the vehicle will be defined in the next T_h time by the arc $a_{i,j}$, each arc's obstacle collision check must extend to the length $l(a_{i,j})$ in the workspace:

$$l(a_{i,j}) = v_{l,i}(T_h - T_b) + \frac{1}{2}\frac{v_{l,i}^2}{\dot{v}_{l,max}} \tag{3.8}$$

For $T_h = T_b$ there is no collision check beyond the instantaneous breaking limit and the collision check scheme becomes similar to that of the Dynamic Window approach of [Fox *et al.*, 1996].

Therefore, in order to assess whether a given arc $a_{i,j}$ is obstacle free, each time discretized node $a_{i,j}(t_k)$, where $t_k = \{t_c, \dots t_k \dots, t_c + T_h; k \in N_k\}$, must be check against each individual object $\mathcal{O}(t_k)$ at time t_k in the future. Vehicle workspace configuration is defined by node $a_{i,j}(t_k)$ (pose) and its geometry, in this implementation a rectangular bounding box. If objects $\mathcal{O}$ are represented as generic polygons, the collision checking is more involved and tools such as PQP package [Larsen *et al.*, 2000] can be used. However, in this implementation the objects are either rectangles defined by their bounding boxes (pedestrians or vehicles) or point objects for generic occupancy of the traversability map (if the map resolution is high enough). This can speed up the collision check process significantly.

Such motion prediction and collision checking scheme insures safe navigation among static and dynamic obstacles in cluttered environments. However, it must be stated that the safety beyond T_h is guaranteed only in terms that if a collision with an object is to occur it will be the object hitting the ego-vehicle at full stop and not the vehicle itself actively causing collision (exempt the case where the ego-vehicle "blocks" an objects trajectory). In fact, choosing an appropriate T_h is directly linked with obstacle dynamic capabilities, perceptional constrains (accuracy of prediction of obstacle motion) and dynamic capabilities of the ego vehicle, where this choice on T_h is still an open issue on current research on motion planning in dynamic environments. Intuitively, it can be stated that choosing a higher T_h allows the ego-vehicle to travel at higher speeds for the same type of obstacle/environment dynamics and configuration, however, the costs of collision checking and future time exploration becomes computationally more expensive.

The exposed feasible trajectories scheme is based on the kinematic level steering control of the vehicle where the control input is the steering angle ϕ of the vehicle, thus circular arcs are a natural trajectory choice. If the vehicle is con-

trolled by its steering rate $\dot{\phi}$, which can be kept constant in a given control cycle T_s, the natural trajectory choice are clothoid curves [Kelly, 2002]. However, the collision checking scheme applies equally to such a case as well.

3.2.3 Traversability Anchored Dynamic Path Following (TADPF) controller

With the set of feasible vehicle trajectories according to Sec. 3.2.2, the deliberate navigation level must enable the vehicle to achieve a global objective, defined in our case by the Goal Manager (GM) as a set of waypoints/goals. In an unstructured environment where no topological information is available a Global Planner (GP) level is needed in order to compute a path towards the current goal. Both $FD^\star$ and $E^\star$ global geometric planners used in our case are guaranteeing global goal free-space connectivity in partially unknown/changing environments. The issue is how to integrate the global connectivity information with the currently available feasible vehicle trajectories. $FD^\star$ and $E^\star$ global planners operate on a discrete set of graph nodes, which in the workspace of the vehicle represent grid cells of the environment. If an obstacle is present within a certain grid cell, the cost of traversing that node can be set to infinite (not traversable) or to a cost that is associated to a size metric of the cell if there is no obstacle present. The costs can also interpolated between not traversable and free in the vicinity of obstacles for increased smoothness. This cell traversability cost can be denoted as c_t. In order to achieve global connectivity, the information about the remaining cost to global goal c_g is also available from graph node planners and are recalculated on-the-fly as new information about environment configuration is available. Due to the fact that for the global path following the traversability information is used, the PFC ("Path Following Controller") to be presented is termed as the TADPF - "Traversability-Anchored Dynamic Path Following" controller [Macek *et al.*, 2008a], [Philippsen *et al.*, 2007].

3.2.3.1 Configuration space feasible vehicle trajectories - traversability and obstacle cost criteria (Γ_t, Γ_o)

In order to chose an optimal vehicle trajectory-arc at each cycle T_s, an arc that is dynamically feasible is checked for potential collision with obstacles. The global navigation function provides the configuration space obstacle regions. If a prohibited node is encountered along an arc $a_{i,j}$ that is less then time $T_{b,i} = \frac{v_{l,i}}{\dot{v}_{l,max}}$ away from the starting vehicle position, the arc is banned. For a prediction horizon T_h of vehicle motion along an arc $a_{i,j}$ with length $l(a_{i,j}) = v_{l,i}T_h$, the traversability cost $\Gamma_t^{(i,j)}$ and cost to obstacles $\Gamma_o^{(i,j)}$ are given as:

$$\Gamma_t^{(i,j)} = \sum_{\iota=1}^{N_{t,\iota}} r\left(a_{i,j}^{(\iota)}\right) \quad , \quad \Gamma_o^{(i,j)} = l(a_{i,j}) - l(a_{i,j,o}) \tag{3.9}$$

where $a_{i,j}^{\iota}$ corresponds to a sampled point on the arc $a_{i,j}$, $r(\cdot)$ the increment in the navigation function value (traversability and goal directedness cost at that point) and $N_{t,\iota}$ being the number of discrete points up to the traversability prediction horizon $T_t \leq T_h$. The $l(a_{i,j,o})$ represents the length of the arc up to the first occurrence of an obstacle along it, with the condition being $l(a_{i,j,o}) \geq T_{b,i}$, i.e. the arc is still safe according to the dynamic breaking limitations of the vehicle. If an arc is completely void of obstacles then $\Gamma_o^{(i,j)} = 0$. Traversability cost Γ_t gives essentially the global direction of the vehicle to steer to and the obstacle cost slows down the vehicle in presence of close obstacles.

3.2.3.2 Path orientation cost criterion (Γ_r)

In Sec. 3.2.3.1 there was no geometric path to the goal position needed. However, by explicitly computing the path, additional information can be exploited. By following the negative gradient of the global navigation function from the current vehicle position $\{x, y\}$ to the goal $\{g_x, g_y\}$, a global reference path can be constructed that is a set of points $\left\{c_{r,d} = c_k; k = 1, \ldots, N_g, c_1 = \{x, y\}, c_{N_g} = \{g_x, g_y\}\right\}$. This

path is further smoothed by a spline technique to give a reference path $\mathcal{C}_r(s)$ which is described with the curvilinear parameter s and curve gradient $\|\mathcal{C}_r\,'(s)\| = \sqrt{p'^2(s) + q'^2(s)} \neq 0, \forall s \in [0, s_f]$, $p(s)$ and $q(s)$ denoting the $x-$ and $y-$ component of $C_r(s)$, respectively.

The kinematic level control objective in this case is to find a longitudinal velocity v_l and steering angle ϕ of the vehicle to follow the reference path $\mathcal{C}_r$ given by the global planner. In particular, a desired reference point is defined on $\mathcal{C}_r(s_d)$ as:

$$x_d = p(s_d)\,,\; y_d = q(s_d)\;,\; (0 \leq s_d \leq s_f)\,. \tag{3.10}$$

Determining the suitable position of the reference point is important for the path orientation cost of each arc. Here, it is proposed to set the curvilinear length of the reference point proportional to the current longitudinal velocity v_l of the robot with a time prediction horizon T_r:

$$s_d = T_r v_l \tag{3.11}$$

and the orientation cost of each arc as:

$$\Gamma_r^{(i,j)} = \frac{\sum_{\iota=1}^{N_{r,\iota}} \|\theta_d - \theta_{i,j}(\iota)\|}{l(a_{i,j,r})} \tag{3.12}$$

where θ_d is the reference point orientation and $\theta_{i,j}(\iota)$ orientation of the vehicle in the ι-th point along the arc $a_{i,j}$. The cost is scaled to the cumulative length $l(a_{i,j,r}) = v_{l,i}T_r$, with the T_r being the orientation cost prediction horizon. This scaling implicitly ensures that given the same orientation cost, the arcs with bigger cumulative length, i.e. the arcs with bigger longitudinal velocity will be chosen. Thus, the vehicle is assured to travel at maximum allowed current longitudinal velocity, taking into account overall constraints.

3.2.3.3 Optimal command choice with comfort criterion

The optimal steering commands $\phi^\star$ chosen at each control cycle minimizes the total weighted sum cost:

$$\phi^\star = \underset{\phi_{i^\star,j}}{\operatorname{argmin}} \left\{ \Gamma^{(i^\star,j)} = \gamma_t \Gamma_t^{(i^\star,j)} + \gamma_o \Gamma_o^{(i^\star,j)} + \gamma_r \Gamma_r^{(i^\star,j)} \right\}. \tag{3.13}$$

where each of the costs Γ_t, Γ_o and Γ_r are normalized before weighting. Although the combined cost Γ could be used also for the longitudinal velocity control, it is desired to impose additional constraints on its profile, due to the comfort of the drive, which is related to the maximum lateral acceleration along the path. In the perfect path following case, the vehicle's curvature would be equal to that of the path, so the maximum lateral acceleration for each $v_{l,i}$ is defined as:

$$a_{L,i,max} = \kappa_{max} v_{l,i}^2 \leq a_{L,max} \tag{3.14}$$

assuming constant movement along the path with curvature κ_{max} on the interval $s \in [0, s_\kappa]$, s_κ being the curvilinear path lookahead. Therefore, the set of feasible longitudinal velocities according to Sec. 3.2.2.1 is further constrained to a set of $\tilde{v}_{l,i}$ velocities based on Eq. 3.14. In order to minimize the travel time, the velocity chosen is:

$$v_l^\star = \max \{ \tilde{v}_{l,i} \}. \tag{3.15}$$

3.2.3.4 Simulation and experimental results

The TADPF path following scheme was tested in a simulation environment with polygonal obstacles. The vehicle scenario was to travel through a set of goal waypoints (global objectives) while avoiding any obstacles on the way. A lidar sensor is attached to the vehicle scanning in the frontal horizontal plane with limited range. As new environment information is available, the global navigation function is recomputed by wavefront expansion from the current goal position.

Based on the gradient information from the navigation function, a globally feasible path for the vehicle to follow is available on-line from any position in the free configuration space. Fig. 3.2 shows a situation where the vehicle approaches a goal position and the wayfront is re-expanded in order to account for newly available obstacle information given by the lidar. In Fig. 3.3 the situation is reversed, since the vehicle has all the information on the obstacle configuration available up to the goal position and the wavefront is retracting. On both Fig. 3.2 and 3.3 right, a zoom-in on the vehicle and currently feasible trajectories (arcs) is given. Free trajectories are marked green, the ones that hit obstacles but are still valid according to the dynamic breaking distance of the vehicle (shown on each arc with corresponding color points) are given in red and the trajectories along the re-propagating wayfront are given in magenta. In the later case, the global gradient is not available for the re-propagation period, however, the traversability information (obstacles) is updated at all times, ensuring safety. The reference point position on the path is given in green.

Figs. 3.6, 3.4, 3.5 show the steering comparison between the TADPF controllers where only traversability-obstacle cost is taken into account (Γ_t, Γ_o) versus reference path orientation cost controller (Γ_r) and the full controller with all three costs, when the vehicle is driven at a constant speed of $v_l = 20km/h$. As can be seen, both separate cost options give similar steering controls. The combined cost pattern depends on the weighting factors of each contribution, where a balanced weighting gives a smoothing effect on the net steering and trajectory.

In the next test drive, the vehicle was allowed to drive the speeds from $v_{l,min} = 10km/h$ up to $v_{l,max} = 30km/h$. When analyzing the longitudinal velocity of the traversability-obstacle cost in Fig. 3.7 it can be seen that the speed is maximized due to the max. traversability change criteria (Eq. 3.9) with the longest arc distance. Slowing down is performed at times only due to the obstacle cost component. By taking the lateral acceleration constraints into account from Eq. 3.14, the maximal v_l along the path can be limited as is shown for the case of the orien-

tation cost profile, where maximum speeds attained are lower (see Fig. 3.8). On average, the a_L stays well withing the bounds $a_{L,max} = 1m/s^2$ for fairly comfortable driving [c:I, 1997] as in Fig. 3.10. The peak values of up to $a_L = 2.5m/s^2$ are due to abrupt change of path orientation and curvature beyond a goal waypoint, when a completely new path is replanned to the next goal. This situation can be improved by parallel planning of the next goal path, while the current one is still being executed and jointed to the next one smoothly at the current goal position, which is the topic of future work. Fig. 3.9 shows the reference point position ρ which increases proportional to the longitudinal speed. Fig. 3.11 shows the current vehicle curvature and that of the closest point along the path for the vehicle and the reference point itself. In the perfect tracking case, the vehicle should have the same curvature as the reference point with a time delay related to the vehicle speed. Main differences can be observed due to initial misalignment of the vehicle orientation with respect to the path and the holonomicity of the path itself. Since the path is generated on-line and the vehicle motion is guaranteed to be safe due to the collision checks along currently chosen command/trajectory, small path following errors are acceptable. The velocity profile of Fig. 3.8 has a quality of a time-optimal profile, in the sense that the vehicle is always driving with the maximum allowed speed according to the safety and comfort measure on lateral acceleration. A further improvement is forseen in generating a smoother profile where the vehicle might take a longer time to traverse a given path but with less acceleration-deceleration phases.

An interesting aspect of traversability based navigation is in the fact that different traversability costs can be assigned to vicinity of different type of obstacles in the environment, if they are labelled appropriately. For instance, a cost mask around an obstacle that is explicitly detected as pedestrian can be given higher cost than that of a vehicle vicinity. This will have as effect that the global path will be additionally moved from the zones deemed as critical, which is the case of a pedestrian that is a completely unprotected object in the environment. In this

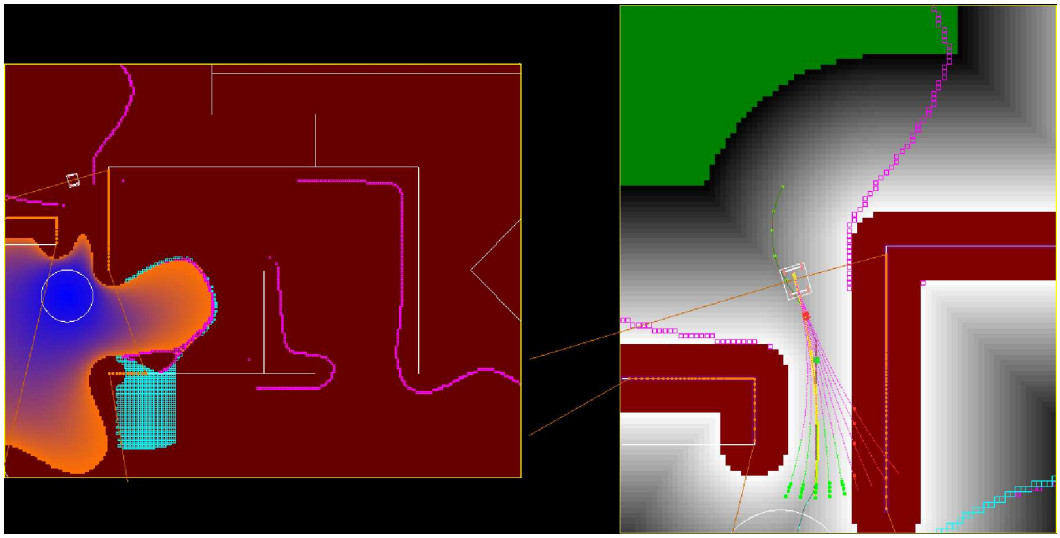


Figure 3.2: Hierarchical navigation with the TADPF controller I. Left: Global environment with the planning phase. Right: Local trajectory view.

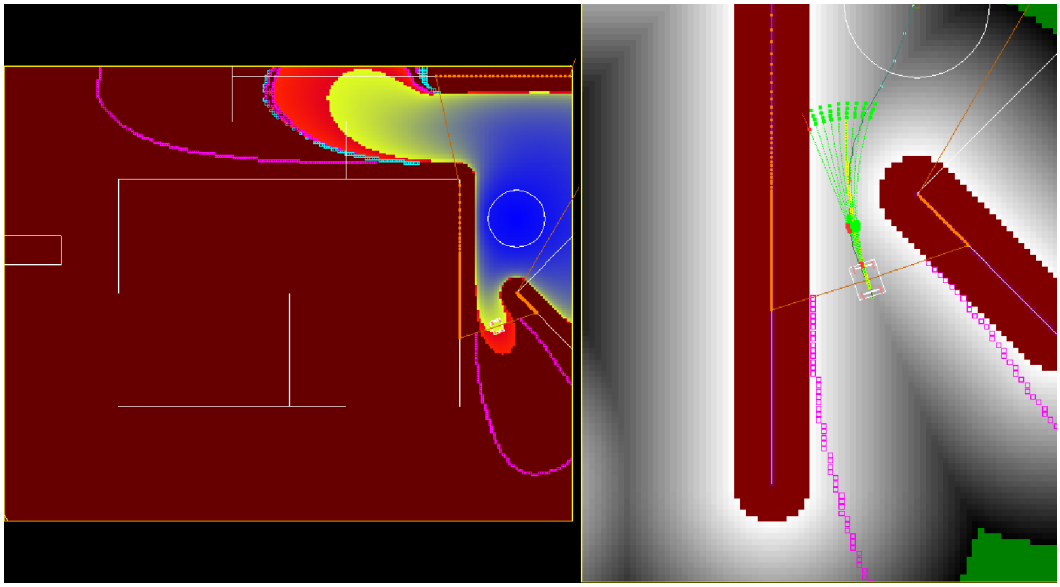


Figure 3.3: Hierarchical navigation with the TADPF controller II. Left: Global environment with the planning phase. Right: Local trajectory view.

sense a prioritized-type navigation can be exerted.

The formentioned global planners are designed to seamlessly integrate new information about the environment, but without considering the temporal aspect, i.e. the future configurations of moving objects. Attempts to include temporal aspect in global planning can be found in [Coué *et al.*, 2006], [Ferguson and Stentz,

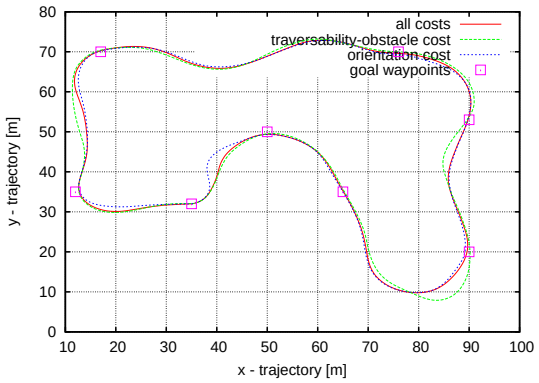


Figure 3.4: Trajectory x-y comparison for the TADPF controller with individual and combined cost criteria.

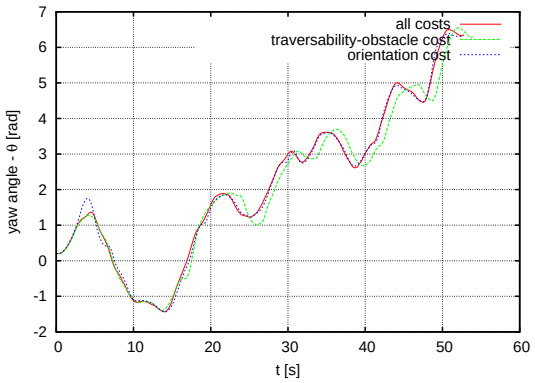


Figure 3.5: Orientation θ comparison for the TADPF controller with individual and combined cost criteria.

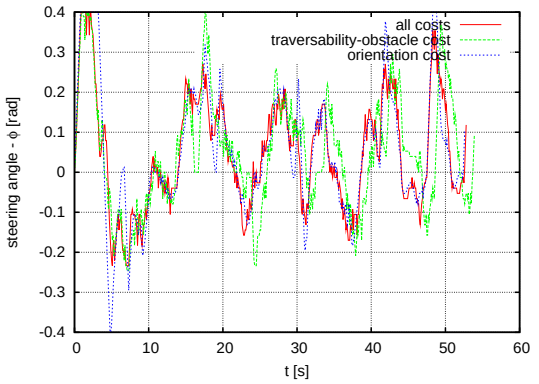


Figure 3.6: Steering angle ϕ comparison for the TADPF controller with individual and combined cost criteria.

Figure 3.7: Longitudinal velocity v_l for the TADPF controller with combined traversability Γ_t and obstacle Γ_o cost criteria.

2007], [Seder and Petrovic, 2007]. However, since the cycle of the vehicle control loop should be constant and real-time compliant, relying solely on traversability information from global planning is not sufficient, since the replanning stage may take considerable time in complex environments. Therefore, in order to assure safety conditions our system relies exclusively on the collision checking scheme of Sec. 3.2.3.1 based on Mapper (MP) information and considers global planner information only as preferred direction for the vehicle to navigate to.

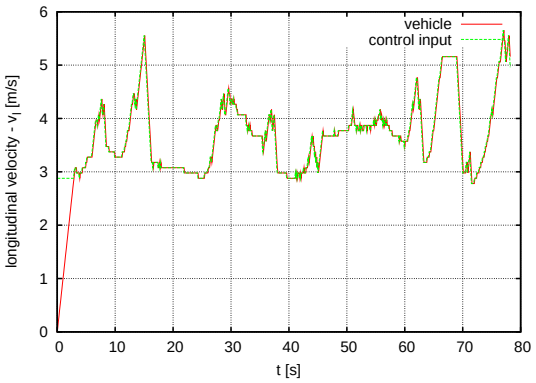


Figure 3.8: Velocity v_l for the TADPF controller with path orientation Γ_r cost criterion.

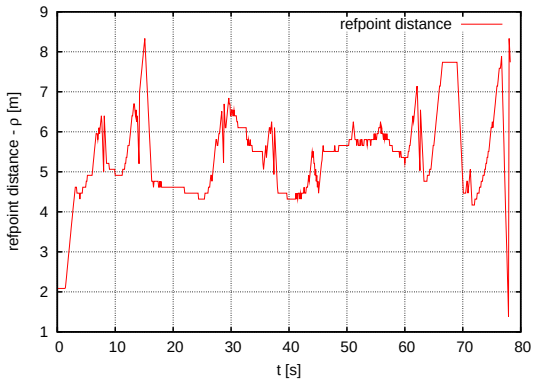


Figure 3.9: Refpoint distance ρ for TADPF controller with path orientation Γ_r cost criterion.

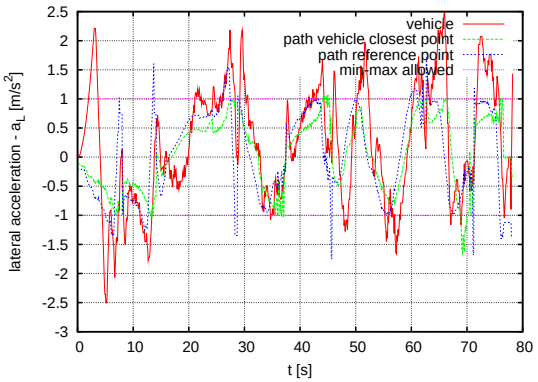


Figure 3.10: Lateral acceleration a_L for TADPF controller with path orientation Γ_r cost criterion.

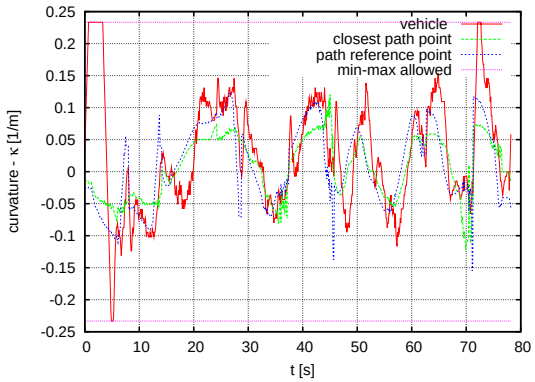


Figure 3.11: Curvature κ for TADPF controller with path orientation Γ_r cost criterion (reference point, closest path point and vehicle itself).

The newly proposed path following scheme TADPF and the overall navigation framework of Sec. 3.2.1 have been tested extensively experimentally on the Smart autonomous test vehicle at constant longitudinal speeds of up to $20km/h$. Fig. 3.12 shows a sequence of snapshots taken from an experimental run with the testing vehicle in an lane structured environment. The obstacle representation was based on generic occupancy grid map used by the $FD^\star$ global planner to calculate the traversability costs with the lane structure rasterized to a cell size of $20cm$.

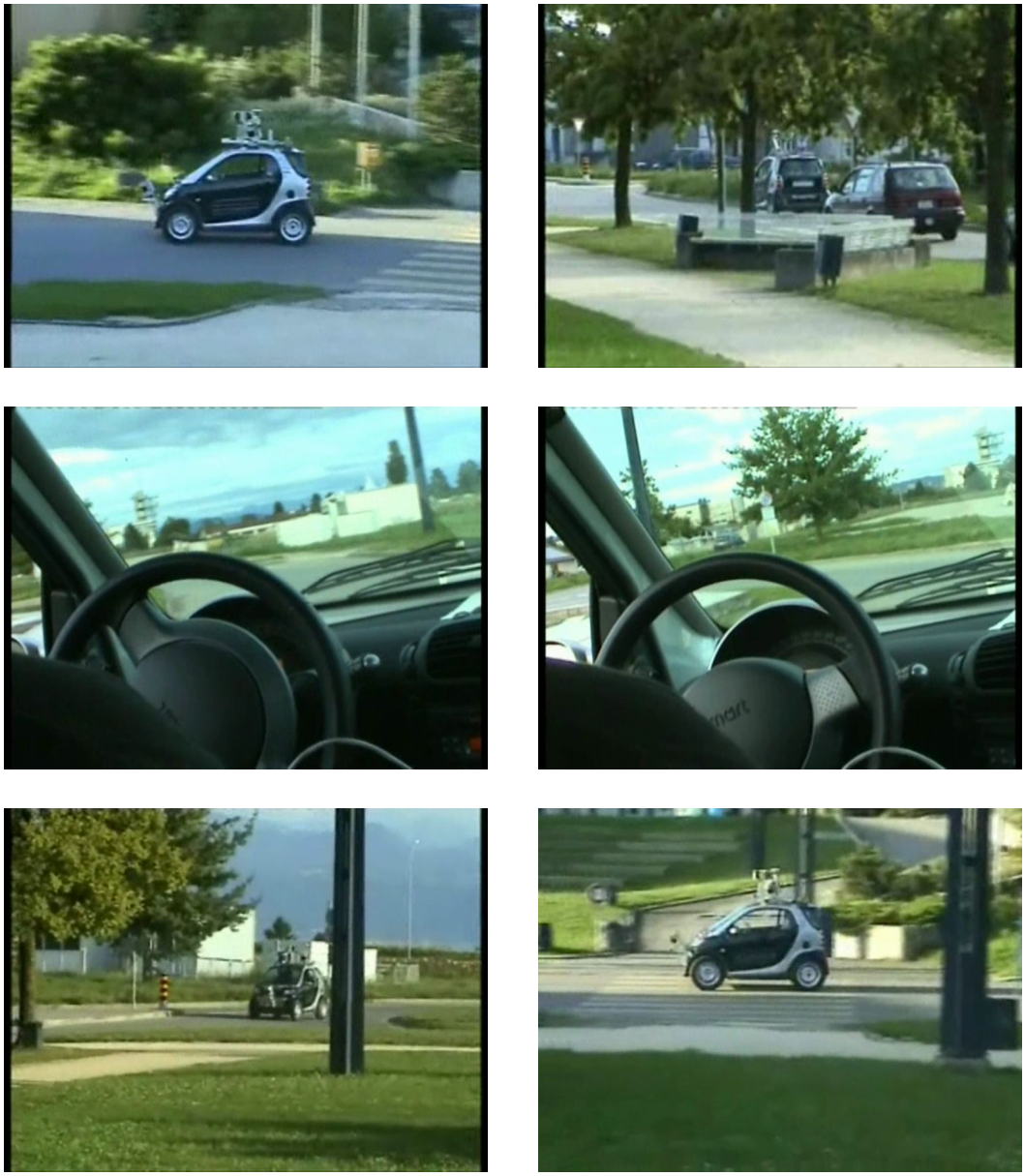


Figure 3.12: An experimental snapshot sequence of the testing vehicle navigating in a lane structured environment using the hierarchical navigation architecture with TADPF controller. The vehicle is following a path computed by the global planner based on global waypoints and a grid lane obstacle representation.

3.2.4 Sliding Mode Path Following (SMPF) controller

By querying the minimal total cost to goal, the global planners can extract the global geometric path $\mathcal{C}_r$, to goal from current vehicle location. Furthermore, the global geometric path can also be given by the structure of the environment itself as is the case in lane based navigation where the path to follow is given by the lane's spine or in the case of negotiating intersections which is an generalization of lane following with predefined traffic rules. In case when global geometric path is available, a control theory based path following scheme can be used.

In [Solea and Nunes, 2006] a sliding mode controller was developed for path tracking of Ackermann-like vehicles in order to address the issues such as fast response, good transient and robustness with respect to system uncertainties and external disturbances. The path following errors are described by y_e and θ_e, where y_e is the lateral distance from the vehicle reference point on the middle of the rear axis $\{x_r, y_r\}$, to the closest point $\{x_d, y_d\}$ on the reference curve $\mathcal{C}_r(s)$, denoted as virtual vehicle position. Angular error θ_e is the difference between the vehicle orientation θ_r and the tangent curve angle at the closest point θ_d, therefore the errors can be written as:

$$\begin{bmatrix} y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta_d & \sin\theta_d \\ -\sin\theta_d & cos\theta_d \end{bmatrix} \cdot \begin{bmatrix} x_r - x_d \\ y_r - y_d \\ \theta_r - \theta_d \end{bmatrix} . \tag{3.16}$$

Assuming the kinematics of the vehicle being the same as in Eq. 4.36, the corresponding error dynamics are:

$$\dot{y}_e = v_l \cdot \sin\theta_e \ , \ \dot{\theta}_e = \dot{\theta}_r = \frac{v_l}{L} tan\phi_r \, , \tag{3.17}$$

noting that $\dot{\theta}_d = 0$ in the path following problem.

In [Solea and Nunes, 2006] a sliding surface was proposed which couples

together the lateral error y_e and the angular error θ_e:

$$s = \dot{y}_e + k_1 \cdot y_e + k_0 \cdot sgn\,(y_e) \cdot \theta_e\,. \tag{3.18}$$

The dynamics of the sliding surface is defined as:

$$\dot{s} = -Q \cdot s - P \cdot sgn\,(s)\ , \tag{3.19}$$

where Q and P are scalars.

The Eq. 3.18 can also be expressed as:

$$\dot{s} = v_l \cdot \dot{\theta}_e \cos\theta_e + k_1 \cdot v_l \cdot \sin_{\theta_e} + k_0 \cdot sgn\,(y_e) \cdot \dot{\theta}_e\,. \tag{3.20}$$

Combining Eq. 3.17, 3.19 and 3.20 yields the desired steering command:

$$\phi_c = \arctan\left(\frac{L}{v_l} \cdot \frac{-Qs - P \cdot sgn\,(s) - k_1 \cdot v_l \cdot \sin\theta_e}{v_l \cdot \theta_e + k_0 \cdot sgn\,(y_e)}\right)\,. \tag{3.21}$$

It was proven in [Solea and Nunes, 2006] by Lyapunov analysis that it suffices for the control law to be stable that $Q, P \geq 0$. Note also, that the longitudinal velocity v_l in this case is not controlled, but is given as an input parameter to the control law of Eq. 3.21. It may be constant or variable according to a predefined velocity profile along the reference path $\mathcal{C}_r(s)$.

3.2.5 KinoDynamic Sliding Mode Path Following (KDSMPF) controller

Although the sliding mode controller of Sec. 3.2.5 provides a fast transient response and small path following error, it has two basic disadvantages. Firstly, it does not take explicitly into account the constraints on the control inputs, such as maximal steering angle ϕ_{max}, nor the dynamic constraints $\dot{v}_{l,max}$ and $\dot{\phi}_{max}$ as described in Sec. 3.2.2.1.

Secondly, there is no check for collision of the chosen vehicle trajectory arising from a given control input ϕ_c at v_l, as is done for the TADPF controller in Sec. 3.2.3.1. This effectively means that the vehicle may hit an obstacle or end in other potentially hazardous situation if there is a significant deviation from the pre-planned reference path, due to path following error, external disturbances or other non-modeled effects. In order to resolve both these aspects and render the navigation of the vehicle safe while taking into account the control and dynamic limitations of the vehicle, the proposed new controller combines the SMPF control strategy with the trajectory feasibility both in kinodynamic and configuration space sense.

As mentioned in Sec. on the SMPF controller, the longitudinal vehicle velocity is considered a parameter, given by an external (hierarchically higher) velocity profile module. Eventhough the interpolated traversability information of the global planner may not be available, the global geometrical path and the binary obstacle traversability map must be calculated. Therefore, the obstacle cost (Sec. 3.2.3.1) and the path orientation cost (Sec. 3.2.3.1), that are partial measures used in the TADPF controller to determine the longitudinal velocity and the steering angle, can still be used here, removing the need of an external velocity profile module. Rather, the longitudinal velocity is adjusted according to the local obstacle configuration according to the Eq. 3.15, denoted $v_l^\star$ as explained in Sec. 3.2.3.3.

The steering command ϕ_c is given by the SMPF controller of Eq. 3.21. In order to enforce the constraints of Sec. 3.2.2.1 and Sec. 3.2.3.1, the steering command at each instant is calculated as:

$$\phi_c^\star = \underset{\phi_{c,j}}{\operatorname{argmin}} \left\{ \|\phi_{c,j} - \phi_c\| \right\} . \tag{3.22}$$

Thus, the optimal control chosen is defined by the kinodynamic arc of the $\{v_l^\star, \phi_c^\star\}$ that is also checked to be configuration space feasible and matches the closest the input command given by the SMPF control strategy $\{v_l, \phi_c\}$. The resulting path

following control strategy is termed KDSMPF - "KinoDynamic Sliding Mode Path Following".

In order to enable on-line adaptation to the changes in the environment the global reference path that is recomputed by the global planner should be close to the steering control cycle rate (in our application up to 10Hz). However, in presence of newly detected or moving obstacles, the global reference path $\mathcal{C}_r$ may become invalid. Indeed, due to the replanning phase of the global planner a new reference path may be obtained by delay or the reference path is globally static (lane based structure of the environment). In this case, the obstacle avoidance is achieved entirely on the reactive level by inhibiting the infeasible vehicle trajectories due to potential collision with obstacles.

3.2.5.1 Simulation and experimental results

This results section deals with comparison of the TADPF, SMPF controller and the proposed synergic version KDSMPF as given in Fig. 3.13 to Fig. 3.18. Since the SMPF control scheme does not provide longitudinal velocity commands, the lateral acceleration based constraints along the path of the TADPF control scheme (Eq. 3.14 and Eq. 3.15) were used to generate the longitudinal velocity control inputs in all three cases (Fig. 3.15). When analyzing the steering angle, curvature and lateral acceleration of the vehicle in the Fig. 3.16, 3.18 and 3.17, respectively, it can be seen that the peak values of the TADPF method are smaller, due to the fact that a reference point look-ahead distance is used, combined with the global navigation function in the configuration space. On the other hand, the SMPF and KDSMPF rely exclusively on the vehicle reference point, which is effectively the closest point on the path with respect to the vehicle rear-axes middle point. As a consequence, abrupt changes in path curvature and tangent direction may cause steering angle control peaks, which is the case when the vicinity of a global waypoint is reached and the global planner recalculates the path to the next one in the list. This issue could be resolved by introducing a middle layer for smooth

waypoint transition, however, this is not considered here.

The KDSMPF method provides smaller path following errors with respect to the path generated by the global planner and can be used efficiently and with less control effort if the global path is smooth enough. In comparison, the TADPF method alone tends to smoothen the vehicle trajectory through the free space because it's based on both global navigation function and current trajectory as can be seen in Figs. 3.13 and 3.14.

If the smoothness and non-holonomicity of the on-line generated global path is not guaranteed, the TADPF method alone is preferred, since it produces less steering signal effort, while still following the global path direction. In contrast, if a global path is smooth, possibly optimized for lesser curvature change along the whole path, the preferred method is KDSMPF that provides smaller path following errors and is invariant on the choice of the reference point lookahead distance, since it bound to the closest point on the path. Note that the SMPF controller alone is not suitable for real application, since it provides no guarantees on safety with comparison to TADPF or KDSMPF method, which include collision checks at each cycle. Moreover, in the pure SMPF scheme the dynamic constraints are not taken into account, which can cause larger delays or even instabilities in the cases of limited actuators, in this case particularly the steering rate $\dot{\phi}$.

The newly proposed path following scheme KDSMPF and the overall navigation framework of Sec. 3.2.1 have been tested extensively experimentally on the Smart autonomous test vehicle at constant longitudinal speeds of up to $18km/h$. Fig. 3.21 shows a sequence of snapshots taken from an experimental run with the testing vehicle in a parking lot unstructured environment in the presence of static or slowly moving vehicle and pedestrians. The obstacle representation used by the $FD^{\star}$ global planned was based on generic occupancy grid map with a cell size of $20cm$. However, the local feasible trajectory collision checks were performed on a occupancy grid map (Mapper module) with reduced cell size of $10cm$ in order to better account for arbitrary obstacle shapes and orientations.

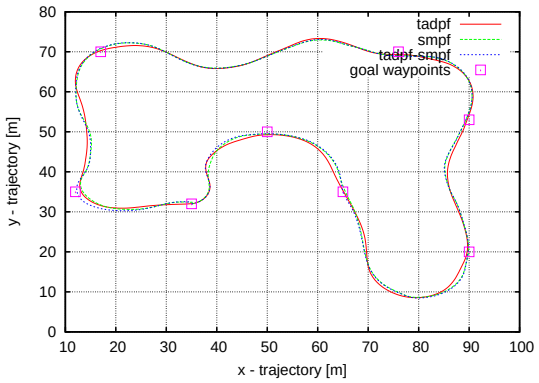


Figure 3.13: Trajectory x-y comparison for the TADPF, SMPF and KDSMPF.

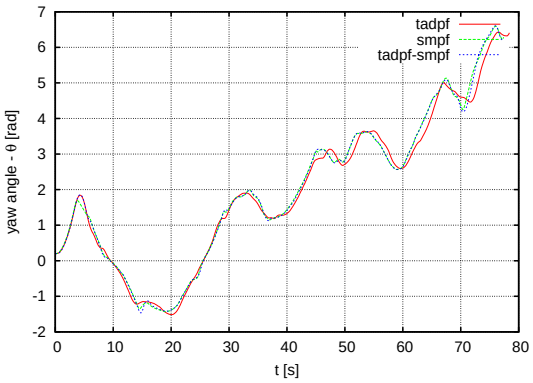


Figure 3.14: Orientation θ comparison for the TADPF, SMPF and KDSMPF.

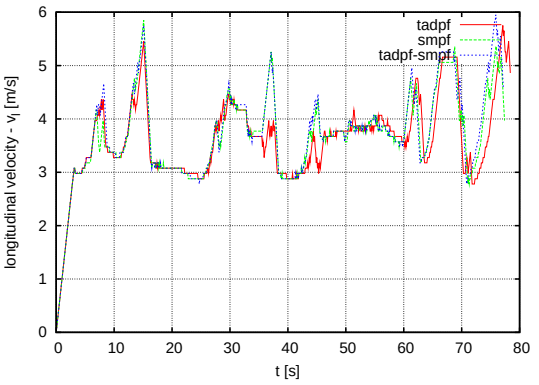


Figure 3.15: Velocity v_l comparison for the TADPF, SMPF and KDSMPF.

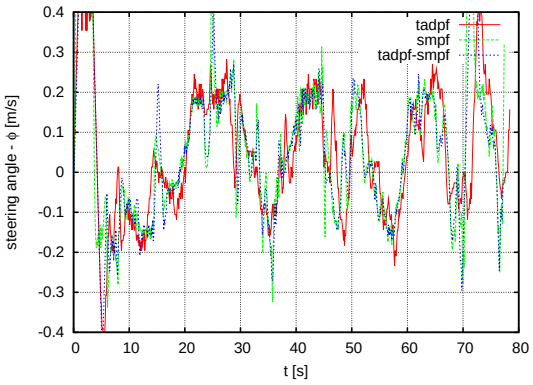


Figure 3.16: Steering angle ϕ comparison for the TADPF, SMPF and KDSMPF.

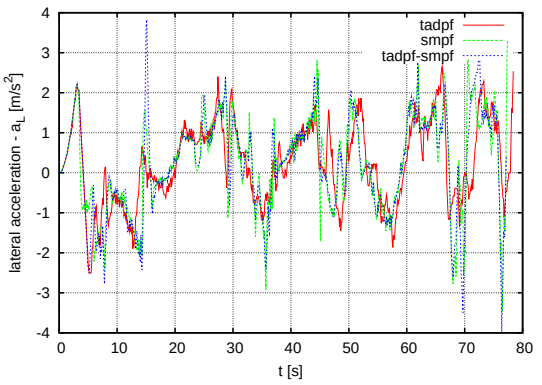


Figure 3.17: Lateral acceleration a_L comparison for the TADPF, SMPF and KDSMPF.

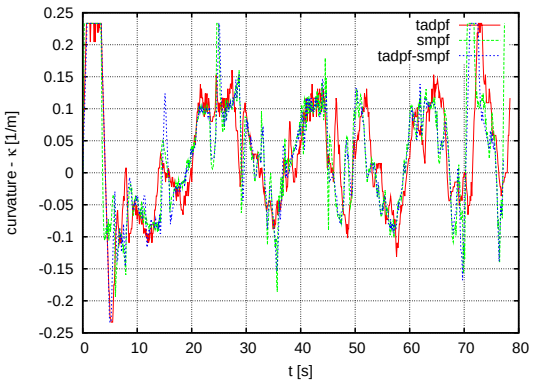


Figure 3.18: Curvature κ comparison for the TADPF, SMPF and KDSMPF.

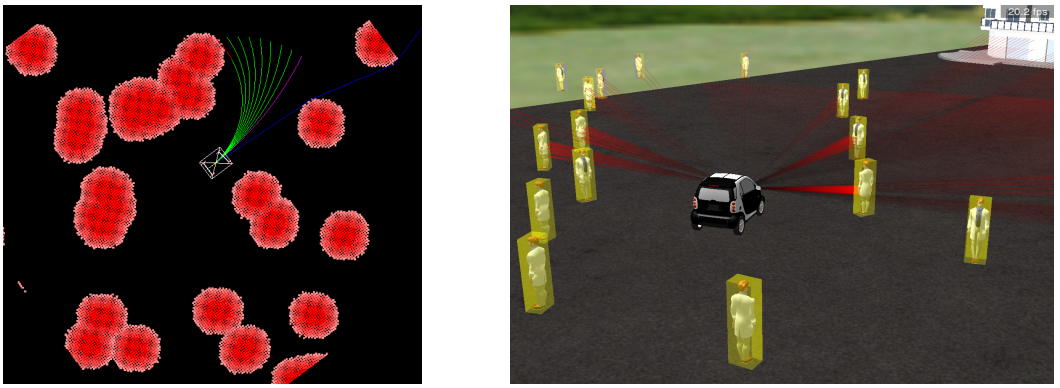


Figure 3.19: Hierarchical navigation with KDSMPF controller - configuration space. Left: Configuration space snapshot of the KDSMPF navigation scheme where the global planner is used for on-line global path replanning. Red obstacles are configuration space expanded obstacles. Feasible robot trajectories are marked in green, the currently optimal robot trajectory according to the global path following marked in magenta. Right: Snapshot of the environment.

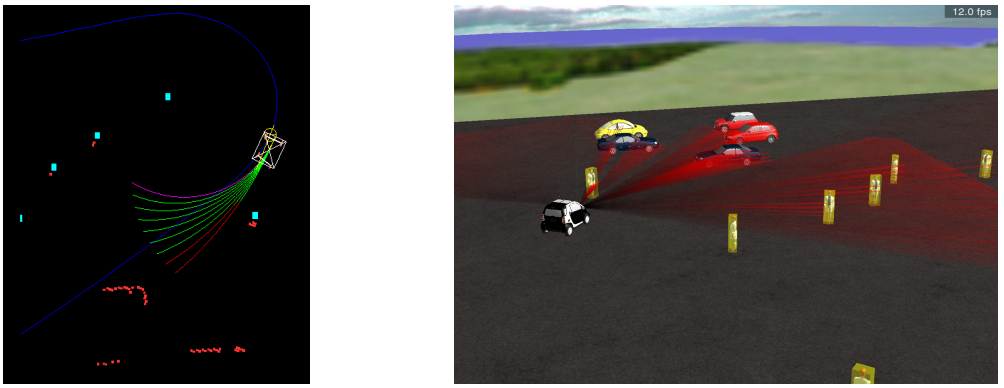


Figure 3.20: Hierarchical navigation with KDSMPF controller - workspace.
Left: Workspace snapshot of the KDSMPF navigation scheme among pedestrians (labelled in cyan) and vehicles in a parking lot scenario. The underlying generic obstacle representation is based on an occupancy grid map (cells marked in red). Feasible robot trajectories are marked in green, prohibited in red, the currently optimal robot trajectory in magenta. Right: Snapshot of the environment.

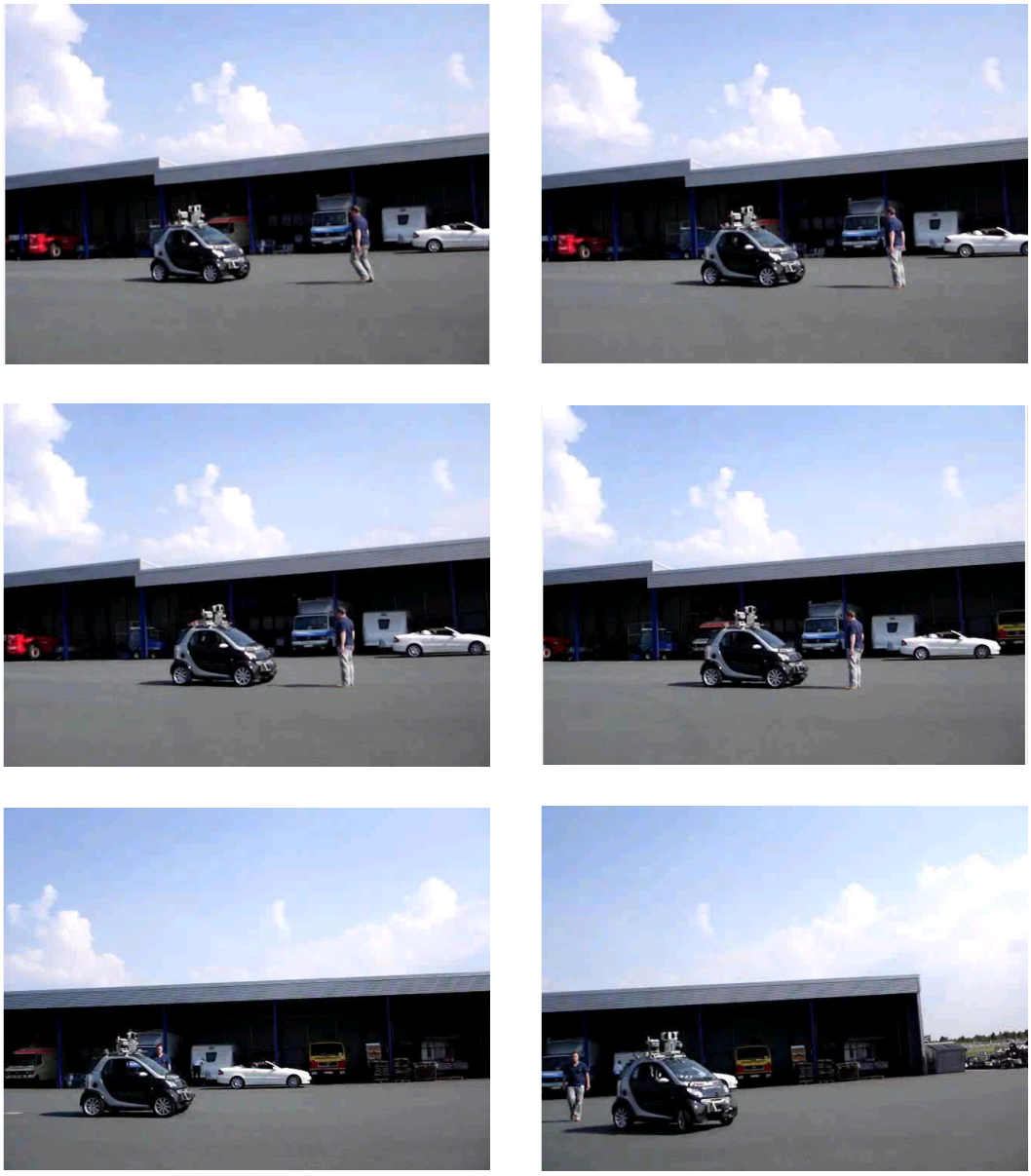


Figure 3.21: Experimental snapshots illustrating the obstacle (pedestrian) avoiding behavior of the hierarchical navigation scheme using KDSMPF controller. If there exists a feasible trajectory given the current obstacle configuration and kinodynamic limitations, the vehicle follows the on-line replanned global path with obstacle avoidance maneuvers. If no feasible trajectory exists within the prediction horizon, the vehicle performs a complete stop.

3.3 Probabilistic obstacle avoidance

The reactive obstacle avoidance system presented in this section implements a bayesian obstacle avoider module where the steering angle of the vehicle is the control output variable modeled according to the reactive sensor-action model with respect to measurement variables which are the longitudinal vehicle velocity and the distance to obstacles in predefined sectors. Sensor-action model is initially implemented as a simple a-priori knowledge based obstacle avoider. However, by tuning the probabilistic model or even learning the model from initially unknown knowledge base according to the human driver actions to the obstacles in scope the performance of the controller can be increased. Therefore, the probabilistic obstacle avoidance can be divided into the "reasoning" part where the proper control action is inferred for a given situation and into the "learning" (adaptation) part where new behaviors can be learned according to the experience data provided by the driver control inputs.

3.3.1 Obstacle avoidance architecture

The task of obstacle avoidance requires a closed loop from sensing the environment, motion planning and execution on the actuators of the vehicles. The implementation of the obstacle avoider for the experimental vehicle is presented in Fig. 3.22 and relies on the software framework *Genom* (Generator of Modules) developed by LAAS, Toulouse [Fleury *et al.*, 1997].

The upper layer illustrates the sensing and acting modules of the architecture. Beside the range data delivered by the Sick LMS-291 attached to the system, the inertial measurement unit delivers angular and translational velocities and accelerations of the car as well. The vehicle's internal bus provides an interface for issuing input control commands, in this case the steering wheel angle.

The vehicle environment is described via an angular clearance map that is a form of occupancy grid, generated from the laser sensor range data where the

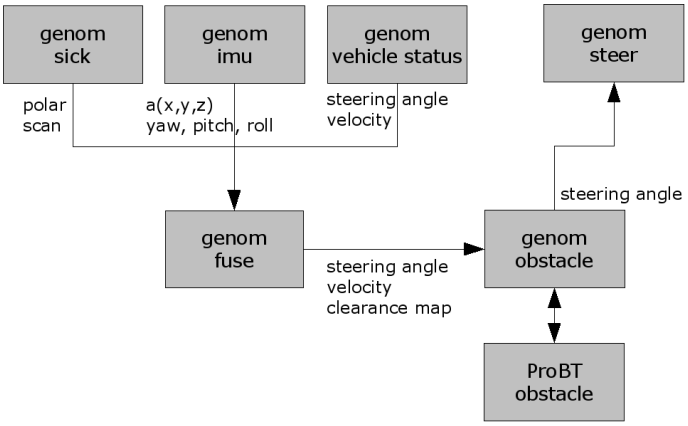


Figure 3.22: Overview of the obstacle avoidance architecture.

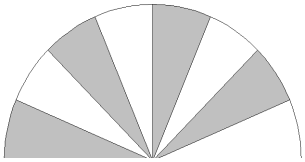


Figure 3.23: Clearance map for obstacle avoidance.

clearance in each of angular sectors is defined by means of the minimum free distance among all range points within that sector and is illustrated in Fig. 3.23.

This raw polar obstacle information is fed into the *ProBT* bayesian probabilistic inference engine [Mekhnacha *et al.*, 2006]. Based on a obstacle avoidance bayesian program that will be presented in more detail Sec. 3.3.2, a new control input steering angle is calculated at each servo cycle.

3.3.2 Obstacle avoidance bayesian programming

The obstacle avoidance algorithm itself is based on the work in [Pradalier *et al.*, 2005] and [Koike *et al.*, 2003] showing the avoidance of obstacles given a trajectory by a centralized path-planner of human driver. This work focusses on the obstacle avoidance module exclusively. However, since the longitudinal velocity is not yet implemented in the car, the obstacle avoidance should be performed based only

on the automated steering module of the car, i.e. the steering angle Φ_T of the car. To robustly account for different measured longitudinal velocities, the input (evidence) to the inference scheme is the current longitudinal velocity V_m and the minimal laser sensor distances within each of the angular sectors D_i where $i = 1, \ldots, N_s$.

Effectively, since there are N_s different angular sectors, there are also N_s different control variables Φ_i possible. It is therefore mandatory to fuse the Φ_i commands to a single final Φ_T value. There are different possibilities to achieve that by e.g. fusing as a weighted sum N_n neighboring commands Φ_i, where possible neighbor numbers are $N_n = 1, \ldots, N_s$ and the weights for each command corresponding to the angular distance from the sector with the minimal object distance for all D_i. However, a switching scheme is implemented here similar to [Pradalier *et al.*, 2005], [Koike *et al.*, 2003] where only a single probabilistic sensor-action model $P(\Phi_i|V_m D_i)$ is taken into account at a given time instant. With a sufficiently high sampling rate and continuity of the environment, the control transitions between different sectors should be smooth.

All the input variables, i.e. sector distances and measured vehicle velocity, and the output steering angles are discretized into integer variables. The actual mapping to the vehicle variables scale (e.g. bipolarity of steering angles) is done separately. A single clearance map sector Bayesian inference program is presented in Fig. 3.24. Since the a-priori knowledge about the environment $P(D_i|\pi_i)$ and the dynamics of the longitudinal velocity $P(V_m|\pi_i)$ is not known (i.e. the uniform pdf is assumed) the joint distribution $P(\Phi_i V_m D_i|\pi_i)$ pdf is equal to the question itself. However, if we assume that the obstacle avoidance is designed for a certain longitudinal velocity with a variation margin, this can be easily incorporated into the program by taking the pdf of $P(V_m|\pi_i)$ to be normally distributed around the longitudinal velocity setpoint.

$$
\left\{
\begin{array}{l}
\left\{
\begin{array}{l}
\left\{
\begin{array}{l}
\text{Relevant Variables:} \\
\Phi_i \in \{1, N_c\}, |\Phi_i| = 13 \\
V_m \in \{1, N_v\}, |V_m| = 5 \\
D_i \in \{1, N_d\}, |D_i| = 20 \\
\text{Decomposition:} \\
P(\Phi_i V_m D_i | \pi_i) = P(D_i|\pi_i) P(V_m|\pi_i)\ P(\Phi_i|V_m D_i \pi_i) \\
\text{Parametric Forms:} \\
P(D_i|\pi_i) = \text{Uniform} \\
P(V_m|\pi_i) = \text{Uniform} \\
P(\Phi_i|V_m D_i \pi_i) = G_{\mu_\Phi(Vm,D_i);\sigma_\Phi(Vm,D_i)}(\Phi_i)
\end{array}
\right. \\
\text{Identification:} \\
\text{A Priori}
\end{array}
\right. \\
\text{Question} \\
P(\Phi_i|V_m D_i \pi_i)
\end{array}
\right.
$$

Figure 3.24: Single clearance map sector Bayesian inference.

Vehicle steering command fusion program for fused steering angle Φ_T is presented in Fig. 3.25, where a sector variable S is introduced that models the switching between different sector steering angle models. The parametric forms $P(S|\vec{D}\pi)$ and $P(\Phi_T|\vec{\Phi}S\ \pi)$ are implemented as functional Diracs within the ProBT framework. The first is defined by choosing the sector that has minimal distance among all sectors at a given instant with probability 1. The second represents the one-to-one probabilistic mapping between the fused steering angle Φ_T and the Φ_i steering angle in the i-th sector that is chosen. Alternatively, the sector random variable S with pdf $P(S|\vec{D}\pi)$ could be also modelled with increased probability of occurrence $[S = i]$ as the distance D_i within sector i decreases, as was implemented in [Koike *et al.*, 2003] according to e.g. sigmoid-shaped pdf.

3.3.3 Command learning (on-line adaptation)

Once the presented sector and command fusion bayesian programs are set, the interested issue is to tune the probabilistic models in an on-line fashion while driving. From the bayesian programming perspective this is the a-priori identification phase, that can actually be performed continuously during the periods where the

$$
\left\{
\begin{array}{l}
\left\{
\begin{array}{l}
\left\{
\begin{array}{l}
\text{Relevant Variables:} \\
\text{Number of sectors } Ns = 8 \\
\Phi_T \in \{1, N_c\}, |\Phi_T| = 13 \\
V_m \in \{1, N_v\}, |V_m| = 5 \\
\Phi_i \in \{1, N_c\}, |\Phi_i| = 13, i = 1 \ldots 8 \\
D_i \in \{1, N_d\}, |D_i| = 20, i = 1 \ldots 8 \\
S \in \{1, Ns\}, |S| = 8 \\
\text{Decomposition:} \\
P(\Phi_T V_m \vec{D} \vec{\Phi} S | \pi) = \begin{array}{l} P(\vec{D}|\pi) P(V_m|\pi) P(S|\vec{D}\pi) \\ P(\Phi_T|V_m \vec{D} \vec{\Phi} \pi) P(\vec{\Phi}|\pi) \end{array} \\
\text{where:} \\
P(\Phi_T|V_m \vec{D} \vec{\Phi} \pi) P(\vec{\Phi}|\pi) = \prod_{i=1}^{N_s} P(\Phi_i|V_m \vec{D} \pi_i) P(\Phi_T|\vec{\Phi} S\ \pi) \\
\\
\text{Parametric Forms:} \\
P(\vec{D}|\pi) = \text{Uniform} \\
P(V_m|\pi) = \text{Uniform} \\
P([S = i]|D_i = min\{\vec{D}\}\ \pi) = 1 \\
P(\Phi_T|\vec{\Phi}\ [S = i]\ \pi) = P(\Phi_i|V_m \vec{D} \pi_i)
\end{array}
\right. \\
\text{Identification:} \\
\text{A Priori}
\end{array}
\right. \\
\text{Question} \\
P(\Phi_T|V_m \vec{D} \pi) = \begin{array}{l} \sum_S P(S|\vec{D}\pi) \\ \sum_{\vec{\Phi}} P(\Phi_T|\vec{\Phi} S\ \pi) \prod_{i=1}^{N_s} P(\Phi_i|V_m \vec{D} \pi_i) \end{array}
\end{array}
\right.
$$

Figure 3.25: Vehicle steering command fusion Bayesian program.

active obstacle avoider mode is switched off and the human driver executes obstacle avoidance maneuvers in different scenarios (i.e. obstacle configurations) but within the longitudinal velocity range taken into account by the bayesian models. Since the initial implementation of the sensor action models $P(\Phi_i|V_m D_i \pi_i)$ is in this case a simple obstacle avoider where the shape of gaussian functions $G_{\mu_\Phi(Vm,D_i);\sigma_\Phi(Vm,D_i)}(\Phi_i)$ is set in linear dependency fashion according to obstacle distance and longitudinal velocity within a sector, it is expected that fine-tuning these models could significantly improve the overall performance. The resultant obstacle avoidance controller would then mimic the human driver. Although, the implementation assumes choosing always a single sector model at each instant, the learning phase would include tuning all the probabilistic models for Φ_i within

each sector according to the overall Φ_T that in the learning phase represents the human driver action. The intercorrelations between the sectors would be encoded in the variance for each possible Φ_i value given a certain measurement evidence.

3.3.4 Experimental results

The proposed probabilistic obstacle avoider was tested on the experimental platform. Time snapshots of a vehicle run can be seen in Fig. 3.26 where the environmental obstacles include a pedestrian and the confines of the surrounding area. Note that the test was run in purely reactive mode, i.e. no global objective was present. The vehicle motion was therefore governed by the occupancy presence in the map sectors (*cf* Fig. 3.23) with the default behavior being to steer away from currently activated obstacle sectors.

3.3.5 Conclusions

The Bayesian reasoning obstacle avoider presented here operates at the basic reactive level of autonomous vehicle navigation based on sector map of environmental obstacles. The experimental verification is made for a very basic scenario and if compared to the reactive level module of Sec. 3.2.1 it does not include an explicit vehicle motion model nor any collision checking in future timesteps. However, it's implicit advantage lies in the generic formulation based on the Bayesian reasoning. The underlying probabilistic inference engine is represented with Gaussian kernel pdfs that can be adapted on-the-fly according to the input mapping between the sensory data and for instance human expert steering control. In that sense, although not validated here, new navigation pattern could be "learned" by the vehicle from scratch.

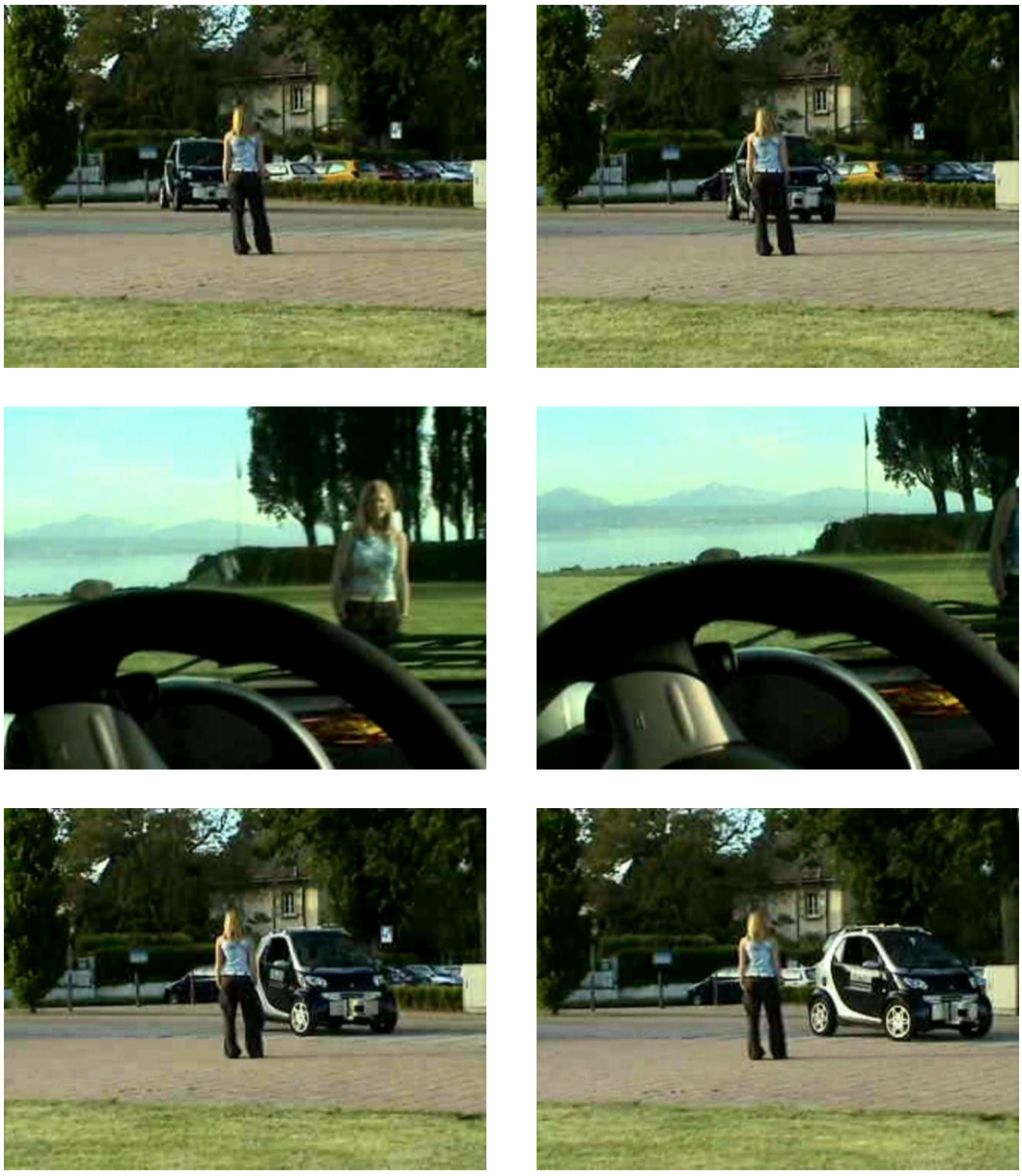


Figure 3.26: An experimental snapshot sequence of the testing vehicle with the probabilistic reactive obstacle avoider. The vehicle is steering away from sectors of occupied angular clearance map (pedestrian and static environmental objects).

3.4 Optimal trajectory planning

The focus of this section is on motion planning of non-holonomic mobile platforms with particular application to autonomous parking. The motion planning problem is formulated as a nonlinear optimal control problem with implicit non-holonomic constraints stemming from the car-like steering vehicle (system equations). Explicit constraints are imposed due to the obstacles in the environment and limited accelerations and velocities of the mobile platform. The optimization objective is minimum time from a start to a goal configuration. Further motivation of the project is to apply nonlinear programming techniques which are generic and widely used in different domains. In particular, direct collocation method is to be used for system discretization. Thereafter, sequential quadratic programming is to be used as the nonlinear programming engine. The approach is motivated by [Kondak and Hommel, 2001] where the problem of motion planning is formulated as a non-linear constrained optimization problem. In contrast to geometric path planning techniques in configuration space [Latombe, 1991], this approach accounts for non-holonomic constraints by considering the kinematic system equations within the path planning phase itself.

3.4.1 Model of a non-holonomic mobile platform (system description)

A typical non-holonomic mobile platform with car-like steering is considered where $\mathbf{s} = (x, y, \theta)$ describes the pose of the system (reference point on the center of the rear axle) and $\mathbf{u} = (v, \psi)$ is the control input, v the longitudinal velocity and ψ the steering angle. The kinematic system equations are:

$$\dot{x} = v\ cos(\theta) \qquad \dot{y} = v\ sin(\theta) \qquad \dot{\theta} = \frac{v}{L}\ tan(\psi) \tag{3.23}$$

where L is the distance between the front and rear axes.

3.4.2 Constraints on control inputs (path constraints)

The bounding box constraints on the control inputs and their rate of change for the mobile platform are:

$$\begin{aligned} \|v(t)\| &\leq v_{max} & \|\psi(t)\| &\leq \psi_{max} \\ \|\dot{v}(t)\| &\leq a_{v_{max}} & \|\dot{\psi}(t)\| &\leq v_{\psi_{max}} \end{aligned} \tag{3.24}$$

In terms of optimal control description the constraints on control inputs throughout the system evolution represent path constraints.

3.4.3 Constraints on system state due to obstacles in the environment (path constraints)

In order to obtain collision-free paths in the environment, obstacles impose additional path constrains on the system state, i.e. a current mobile platform configuration cannot be inside a given obstacle. The obstacles are modeled as a set of points that each contribute to the artificial potential field of obstacles that influences the mobile platform movement. The potential field exerted by a particular obstacle point is according to [Kondak and Hommel, 2001]:

$$P^i(\mathbf{s}) = \begin{cases} P^i_x(\mathbf{s}) & : \quad R_y - x^i_R \geq R_x - y^i_R \\ P^i_y(\mathbf{s}) & : \quad R_y - x^i_R < R_x - y^i_R \end{cases} \tag{3.25}$$

where:

$$\begin{aligned} P^i_x &= P_{max}(1 - \frac{x^i_R}{R_x}) \\ P^i_y &= P_{max}(1 - \frac{y^i_R}{R_y}) \end{aligned} \tag{3.26}$$

if the obstacle is inside the platform and

$$P_x^i(\mathbf{s}) = P_y^i(\mathbf{s}) = 0 \tag{3.27}$$

if the obstacle i is outside the platform. The coordinates (x_R^i, y_R^i) of obstacle i are taken with respect to the platform reference point (which can be arbitrary but for simplicity taken to correspond with the point of rotation of the platform in the center of the rear axis). P_x and P_y are distances from the reference point to the platform boundaries (rectangular shape assumed) and P_{max} is a scaling factor. A collision-free movement is ensured if for each obstacle i:

$$P_i(s) \leq 0 \qquad (i = 1, 2, \ldots N) \tag{3.28}$$

where N is the number of obstacle points in the environment. Thus defined constraints on each of the obstacles $i = 1, 2, \ldots N$ allows for superposition of constraints into a single one:

$$S(\mathbf{s}) = \sum_{i=1}^{N} P_i(\mathbf{s}) \leq 0 \tag{3.29}$$

3.4.4 Objective function

The objective function that minimizes the final position error only may be studied (feasibility path):

$$J_1(\mathbf{u}, \mathbf{s}, t_f) = -\left(\mathbf{s}_{ref}(t_f) - \mathbf{s}(t_f)\right)^2 \tag{3.30}$$

where t_f is a free parameter and $\mathbf{s}_{ref}(t_f)$ reference pose at the final time.

In order to obtain a time-optimal solution the objective function is simply:

$$J_2(\mathbf{u}, \mathbf{s}, t_f) = -t_f \tag{3.31}$$

.

3.4.5 Complete optimization problem formulation

Given the initial and final pose of the system $s_{start} = s_0$ and $s_{goal} = s_{t_f}$ find a control profile $\mathbf{u}(t)$ that minimizes the objective function $J(\mathbf{u}, \mathbf{s}, t_f)$ subject to path constraints of (3.24) and (3.29).

In order to obtain a numerical optimization solution, the state space and input variables will be discretized and cast as a static nonlinear optimization problem [Kondak and Hommel, 2001], [Srinivasan, 2005]:

$$\min_{\mathbf{u},\mathbf{s},t_f} J(\mathbf{u}, \mathbf{s}, t_f) = \min_{\tilde{\mathbf{x}}} J(\tilde{\mathbf{x}}) \tag{3.32}$$

subject to:

$$c_{eq_i}(\tilde{\mathbf{x}}) = 0,\ i \in E \tag{3.33}$$

$$c_{in_i}(\tilde{\mathbf{x}}) \leq 0,\ i \in I \tag{3.34}$$

where the vector $\tilde{\mathbf{x}} = (\mathbf{s}_1^T, \mathbf{u}_1^T, \mathbf{s}_2^T, \mathbf{u}_2^T, \ldots, \mathbf{s}_M^T, \mathbf{u}_M^T, t_f)^T$ contains values of the states and the control vector at discrete time points t_k:

$$\mathbf{u}_k^T = (v(t_k), \psi(t_k)) \tag{3.35}$$

$$\mathbf{s}_k^T = (x(t_k), y(t_k), \theta(t_k)) \tag{3.36}$$

for $k = 1, 2, \ldots, M$. The equality constraint functions c_{eq_i} are obtained after discretization of state equations (3.23) and the inequality constraint functions c_{in_i} after discretization of path constraints (3.24) and (3.29).

3.4.6 Simulation results for the parking problem of a non-holonomic vehicle platform

According to the autonomous parking problem, three major scenarios have been studied:

- **Parallel parking with obstacles**
- **Row parking with obstacles**
- **Slide parking with obstacles**

Scenarios of motion planning without obstacles are also referred to as steering problems.

In all simulation runs the clearance to obstacles according to (3.25) was calculated such that the size of the vehicle in the optimization problem corresponds exactly to the "physical" size. Therefore, constraints are satisfied already at the vehicle-obstacle surface distance zero. In order to increase the safety, the vehicle dimensions could be artificially increased for a safety margin in the optimization task.

3.4.6.1 Matlab simulation results

In order to check the optimization idea the Matlab function "fmincon" was used with equidistant time discretization. Due to the calculation time and size limitations only a small number of time discretization points was used (i.e. 15), therefore, the nonlinear constraints due to the obstacles on the path was not completely satisfied. Due to the very coarse discretization based on kinematic input, i.e. $\mathbf{u} = (v, \psi)$, the bounding box constraints on the rate of change of the signals, i.e. $\dot{\mathbf{u}} = (\dot{v}, \dot{\psi})$, (Eq. (3.24)) were not considered in the Matlab simulations whereas in the DIRCOL simulations this was the case.

3.4.6.1.1 Parallel parking Results include figures Fig. 3.27 to Fig. 3.32.

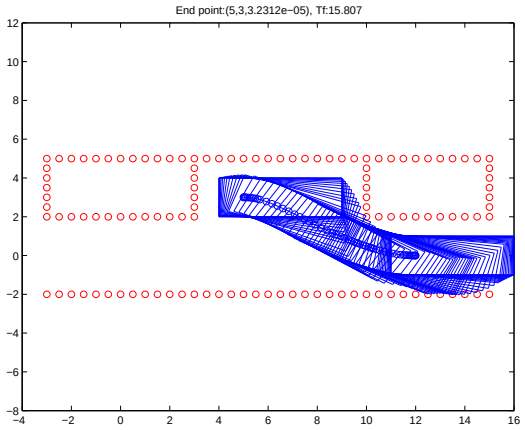


Figure 3.27: Parallel parking - robot motion (feasibility path) (J1)

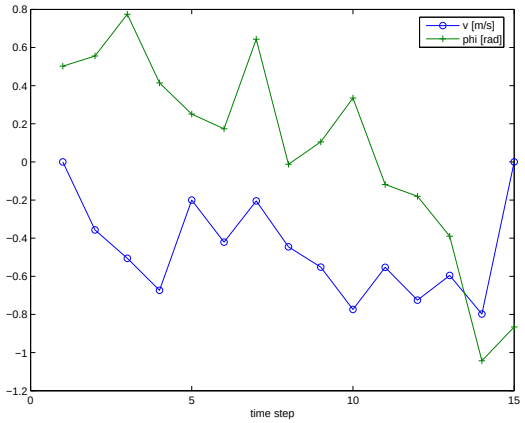


Figure 3.28: Parallel parking - inputs kinematic level (feasibility path) (J1)

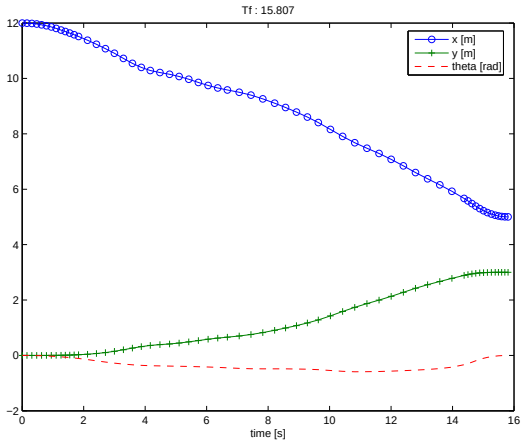


Figure 3.29: Parallel parking - vehicle pose (feasibility path) (J1)

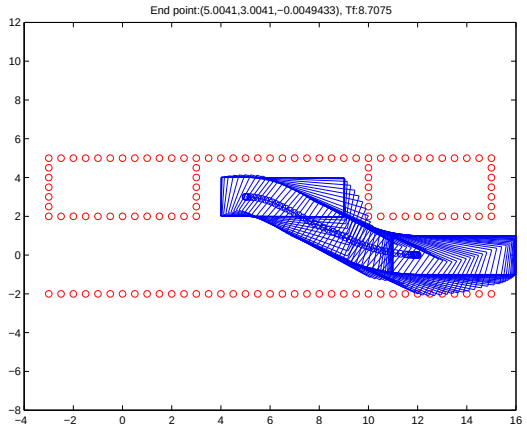


Figure 3.30: Parallel parking - robot motion (time optimal) (J2)

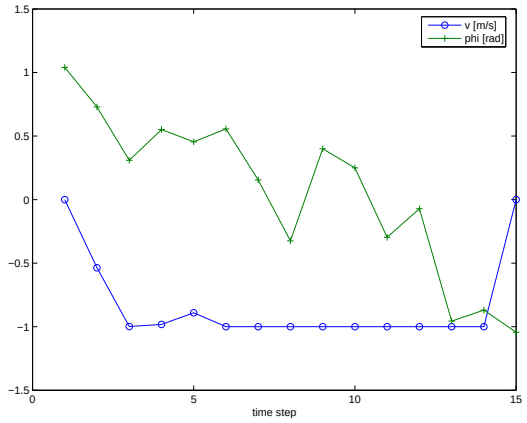


Figure 3.31: Parallel parking - inputs kinematic level (time optimal) (J2)

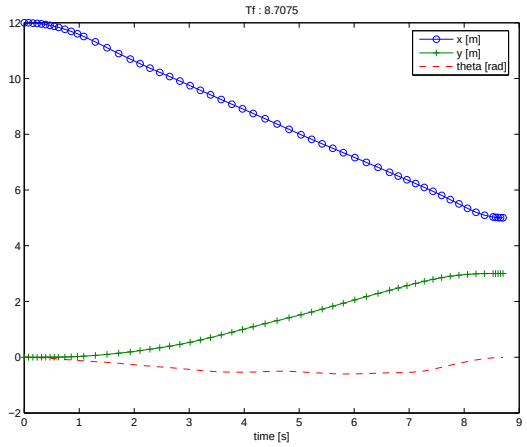


Figure 3.32: Parallel parking - vehicle pose (time optimal) (J2)

3.4.6.1.2 **Row parking** Results include figures Fig. 3.33 to Fig. 3.38.

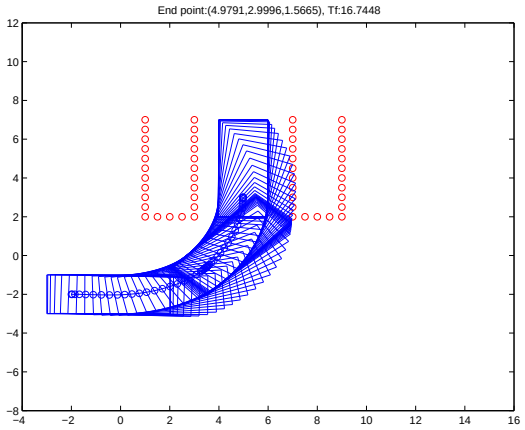


Figure 3.33: Row parking - robot motion (feasibility path) (J1)

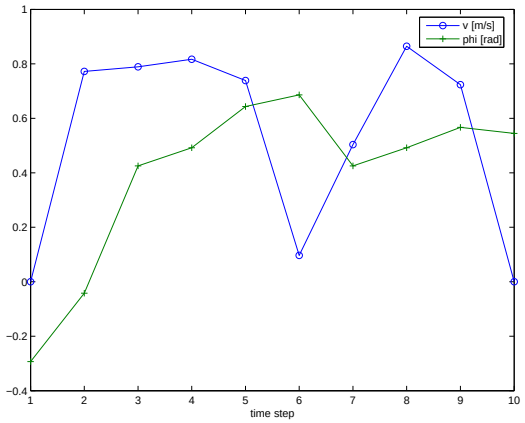


Figure 3.34: Row parking - inputs kinematic level (feasibility path) (J1)

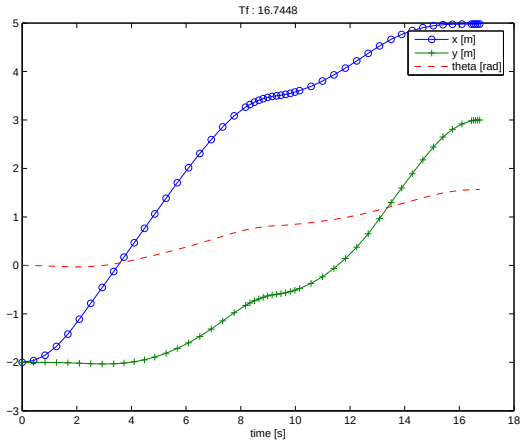


Figure 3.35: Row parking - vehicle pose (feasibility path) (J1)

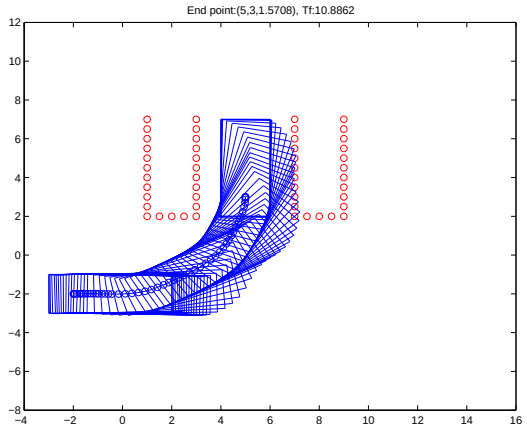


Figure 3.36: Row parking - robot motion (time optimal) (J2)

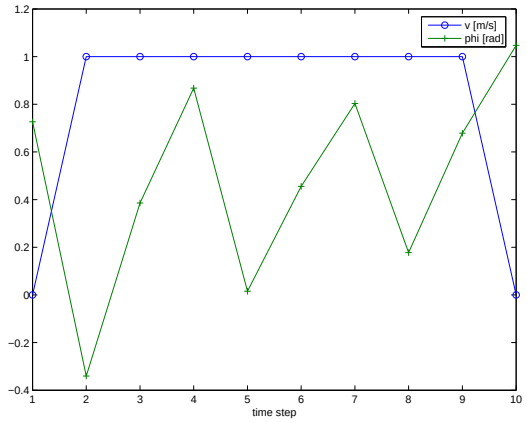


Figure 3.37: Row parking - inputs kinematic level (time optimal) (J2)

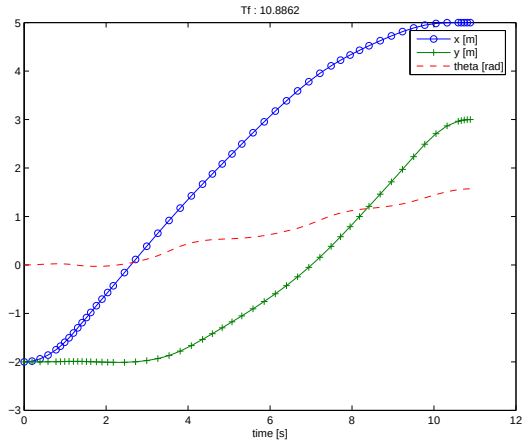


Figure 3.38: Row parking - vehicle pose (time optimal) (J2)

3.4.6.1.3 Slide parking

Results include figures Fig. 3.39 to Fig. 3.44.

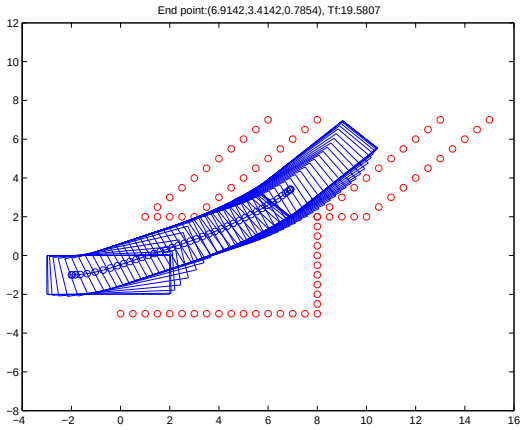


Figure 3.39: Slide parking - robot motion (feasibility path) (J1)

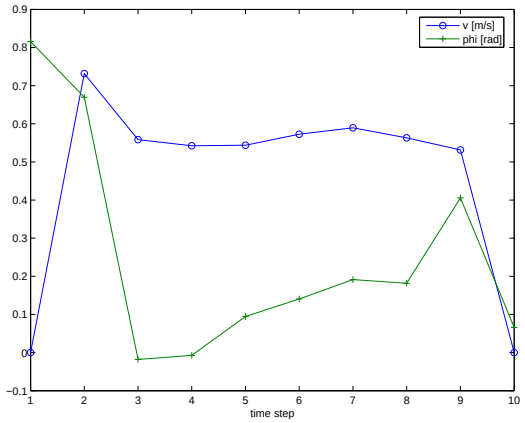


Figure 3.40: Slide parking - inputs kinematic level (feasibility path) (J1)

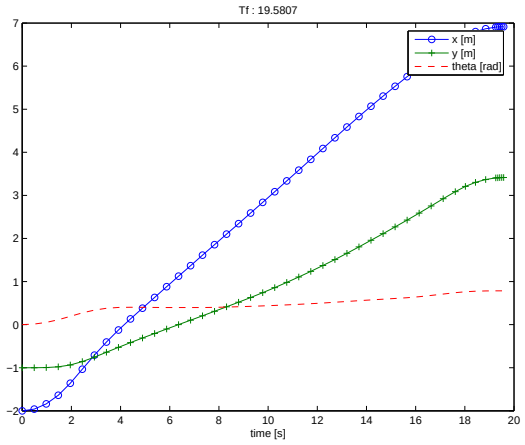


Figure 3.41: Slide parking - vehicle pose (feasibility path) (J1)

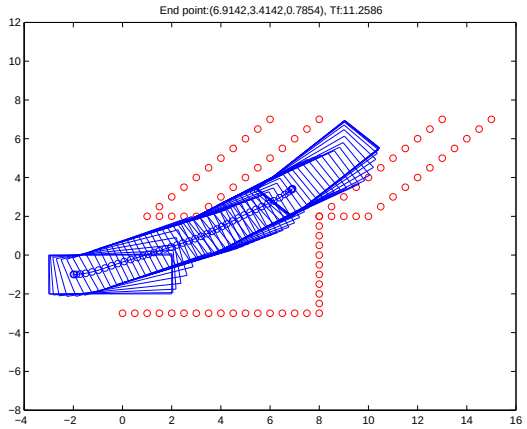


Figure 3.42: Slide parking - robot motion (time optimal) (J2)

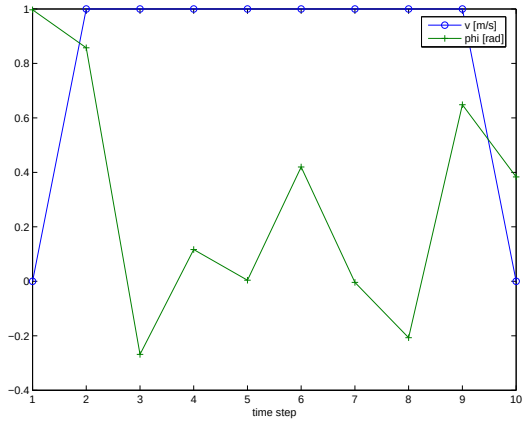


Figure 3.43: Slide parking - inputs kinematic level (time optimal) (J2)

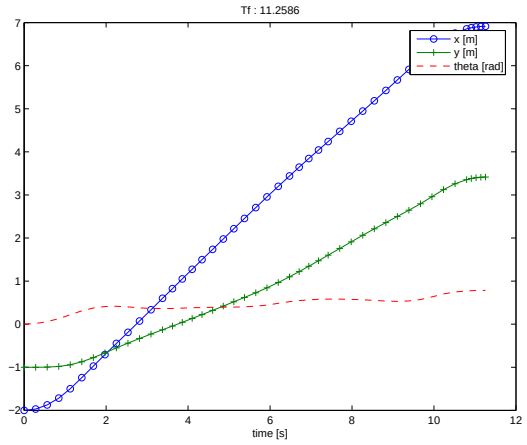


Figure 3.44: Slide parking - vehicle pose (time optimal) (J2)

3.4.6.2 DIRCOL-SNOPT simulation results

For an optimized discretization of the continuous (infinite-dimensional) nonlinear system which takes into account dynamics of the system (approximation of the system dynamics with cubic polynomials) as well as the stability of the optimization procedure (stability of the adjoint variables), the DIRCOL software package [von Stryk,] which implements the direct collocation method [von Stryk, 1993] was used. The software is coupled with SNOPT sequential quadratic programming (SQP) package [P.E. Gill and Saunders, 2005] which resolves the static nonlinear optimization problem of (3.32).

3.4.6.2.1 Parallel parking Results include figures Fig. 3.45 to Fig. 3.54.

3.4.6.2.2 Row parking The successful row parking was shown on figures Fig. 3.55 to Fig. 3.59. A failure to meet the nonlinear obstacle constraints is shown in Fig. 3.60 to Fig. 3.64 where the parking space is much more confined. A possible remedy would be a 3-stage maneuver with forward move -backward steering-forward parking stages. The solution to this problem would require a more sophisticated initial input/state estimate.

3.4.6.2.3 Slide parking The results are shown in Fig. 3.65 to Fig. 3.69.

3.4.7 Conclusion

In this section the non-holonomic path planning, obstacle avoidance and control of autonomous vehicles was approached in a simultaneous manner. The particular problem studied was autonomous parking mode. The system dynamic equations which describe the kinematic level control of the vehicle discretized on equidistant time scale. Since the system itself exhibits no discontinuities in motion, this is a reasonable assumption. However, the nonlinear constrains imposed on the system by the environmental obstacles may require more refined discretization in the crit-

ical areas near the obstacles. This is a critical issue in successful optimization, i.e. in acquiring a feasible robot/vehicle path. The refinement step was not possible in the Matlab simulation runs, unless increasing the number of discrete steps (grid nodes) to a large number. Nevertheless,this grid increase may become prohibitive to the Matlab simulation altogether. Therefore, an alternative approach which enables adaptive grid insertion based on the system dynamics and constraints was also implemented using direct collocation method DIRCOL in coupling with the SNOPT nonlinear solver. The results shown both in Matlab and DIRCOL simulations show successful and failure scenarios of parallel, row and slide parking where either the feasibility path is shown (objective function minimizes the final position error) or the time optimal path. In the time optimal case, the vehicle actuators are brought to the saturation level whenever possible. It became clear during the simulation tests that an initial good estimation of the system dynamics and inputs may be a critical issue also. In the Matlab simulations the optimization runs where in some cases restarted with the resulting input/time discretization values from the previous runs. In the DIRCOL approach, a linear initial estimation from the start to the goal position was assumed, whereas the inputs where left as free parameters (no initial estimation). An promising approach to solve the problem of initial estimation may be the kinodynamic planning in [Fraichard, 1992] which consideres both structural and dynamic constraints as a search in discrete space. Thereafter, the DIRCOL-SNOPT nonlinear optimization step could be performed, in order to obtain a refined, time-optimal path, which is a subject of future work. From the practical implementation point of view, the choice of the particular optimization technique and dynamic system discretization is crucial. For comparison, the Matlab optimization engine that uses SQP (Sequential Quadratic Programming) optimization method computed a very coarse control sequence solution in an order of 1 min whereas the DIRCOL-SNOPT performed the computation for the same initial conditions at a much finer resolution within a 2 sec to 3 sec range which is an order of magnitude better performance.

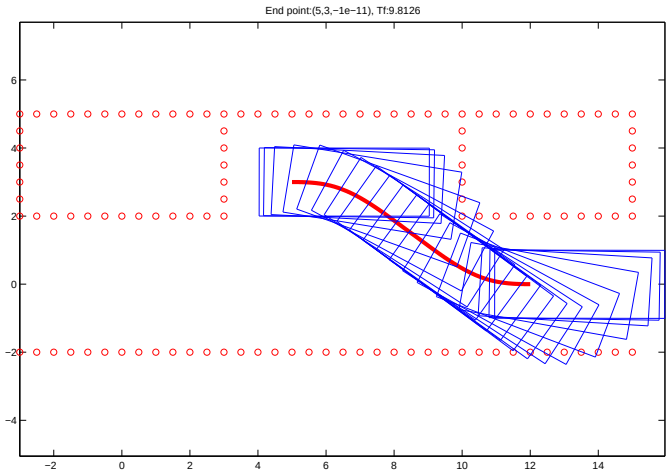


Figure 3.45: Parallel parking (no constraints) - robot motion (time optimal) (J2)

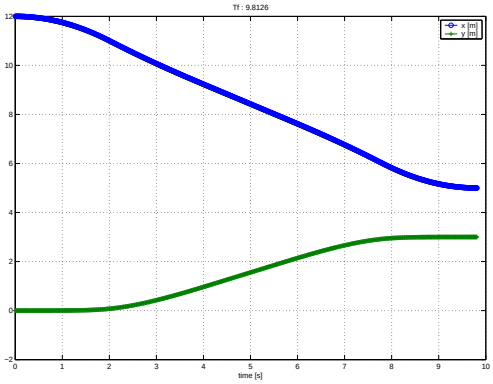


Figure 3.46: Parallel parking (no constraints) - xy coordinates (time optimal) (J2)

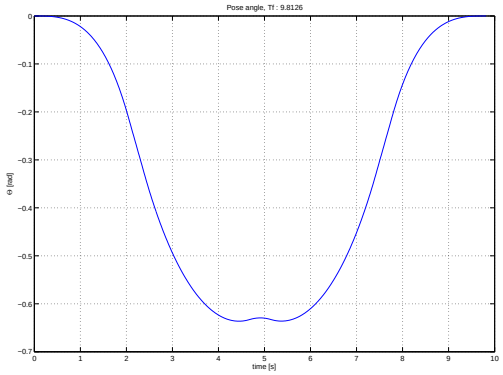


Figure 3.47: Parallel parking (no constraints) - vehicle heading (time optimal) (J2)

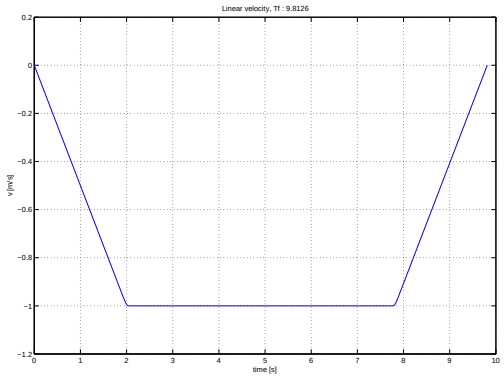


Figure 3.48: Parallel parking (no constraints) - linear velocity (time optimal) (J2)

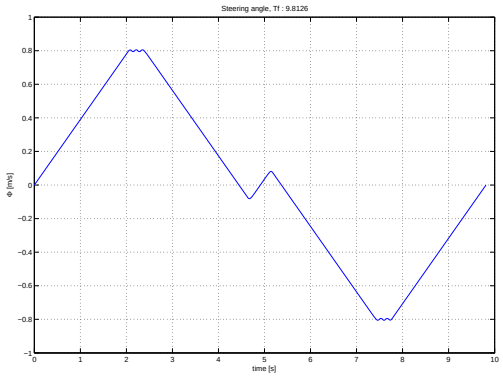


Figure 3.49: Parallel parking (no constraints) - steering angle (time optimal) (J2)

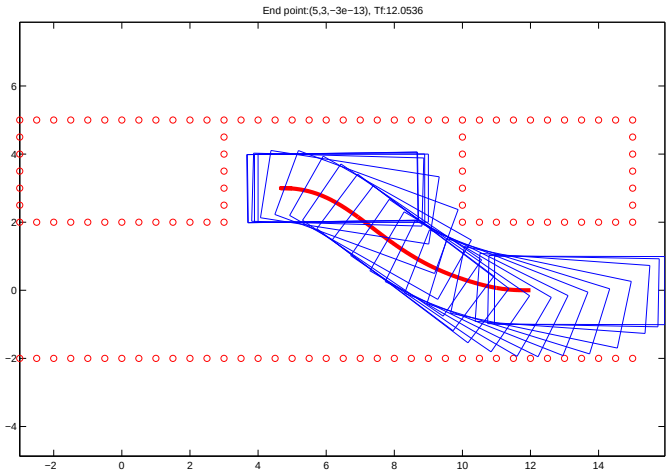


Figure 3.50: Parallel parking (obstacle constraints) - robot motion (time optimal) (J2)

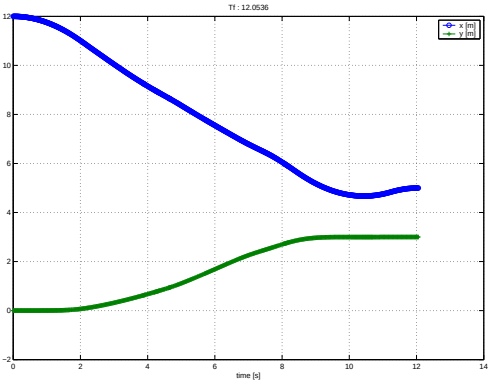


Figure 3.51: Parallel parking (obstacle constraints) - xy coordinates (time optimal) (J2)

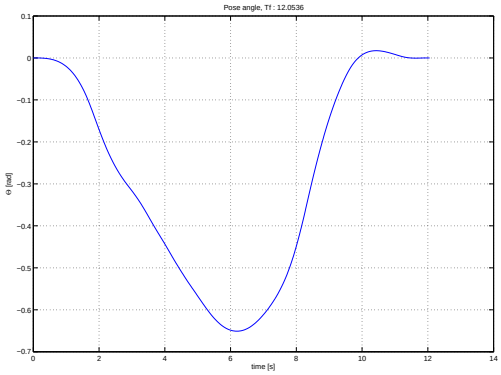


Figure 3.52: Parallel parking (obstacle constraints) - vehicle heading (time optimal) (J2)

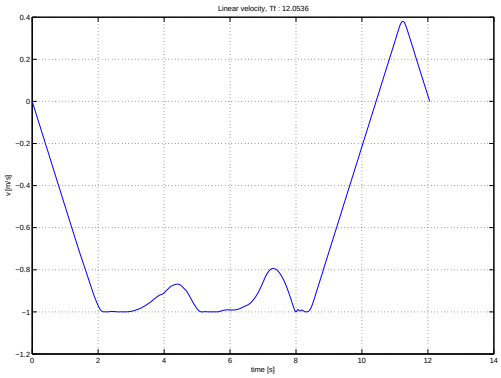


Figure 3.53: Parallel parking (obstacle constraints) - linear velocity (time optimal) (J2)

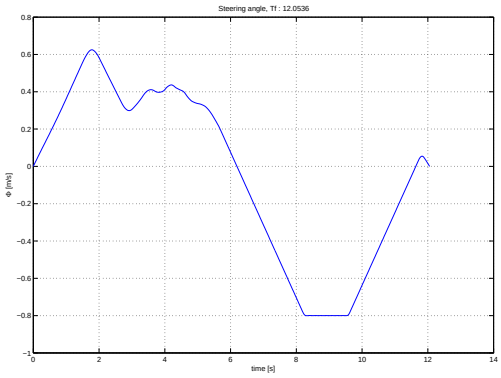


Figure 3.54: Parallel parking (obstacle constraints) - steering angle (time optimal) (J2)

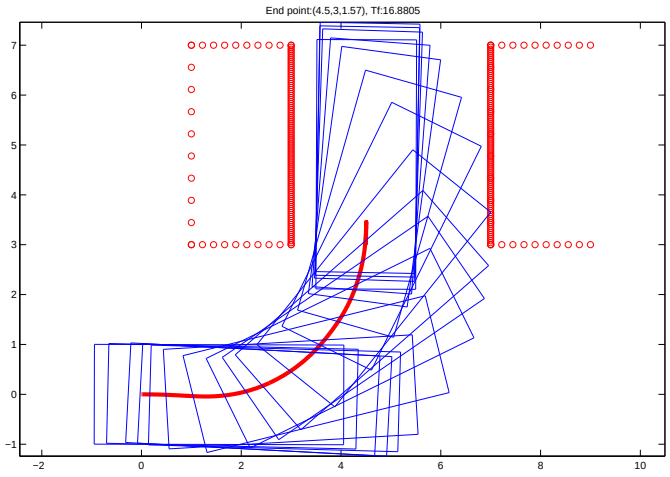


Figure 3.55: Row parking (obstacle constraints) - robot motion (time optimal) (J2)

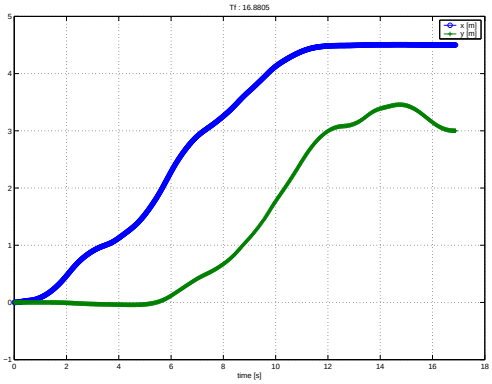


Figure 3.56: Row parking (obstacle constraints) - xy coordinates (time optimal) (J2)

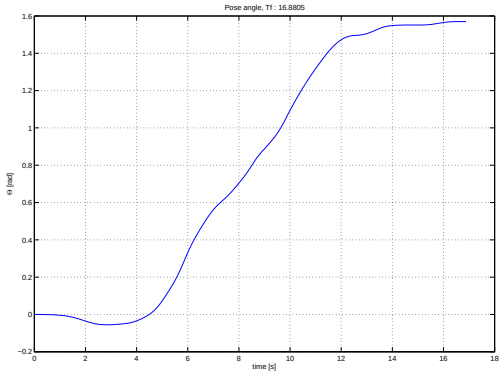


Figure 3.57: Row parking (obstacle constraints) - vehicle heading (time optimal) (J2)

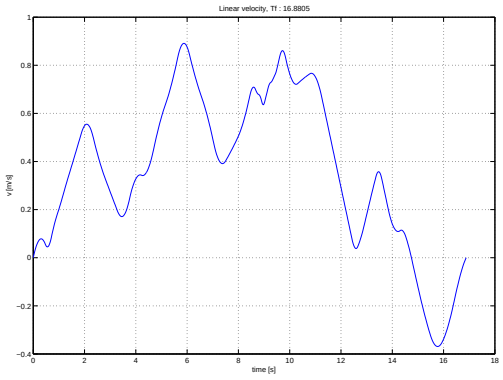


Figure 3.58: Row parking (obstacle constraints) - linear velocity (time optimal) (J2)

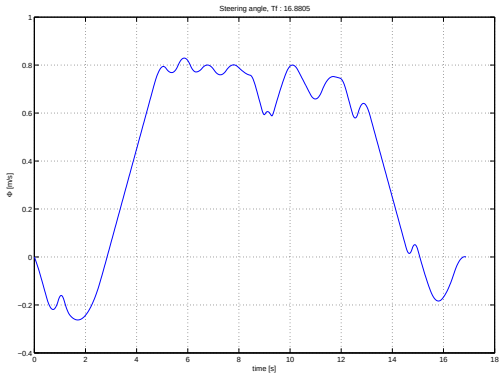


Figure 3.59: Row parking (obstacle constraints) - steering angle (time optimal) (J2)

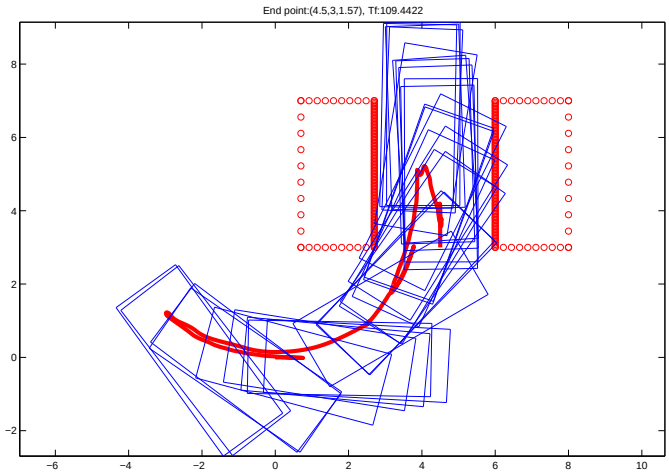


Figure 3.60: Row parking failure (obstacle constraints) - robot motion (time optimal) (J2)

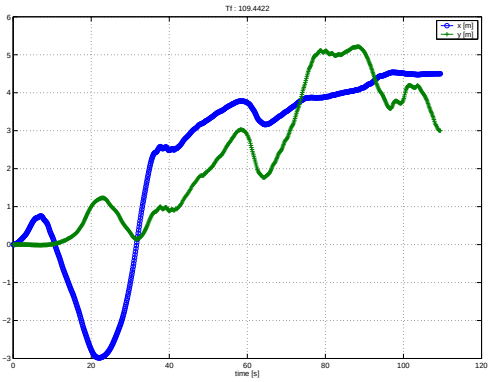


Figure 3.61: Row parking failure (obstacle constraints) - xy (time optimal) (J2)

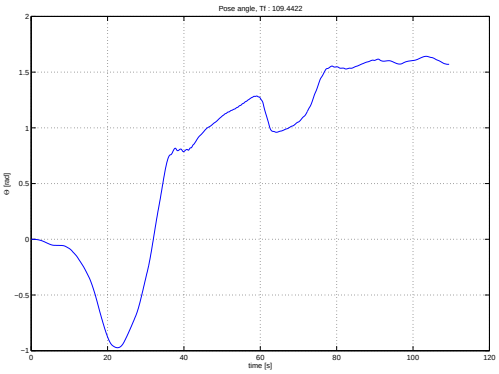


Figure 3.62: Row parking failure (obstacle constraints) - vehicle heading (time optimal) (J2)

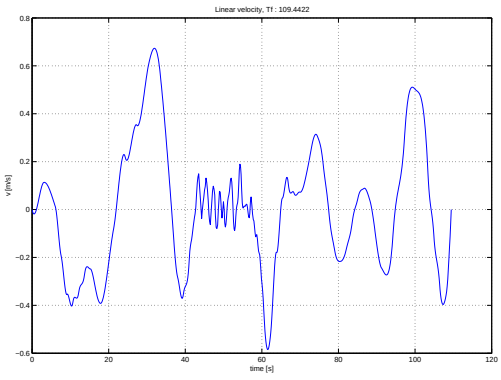


Figure 3.63: Row parking failure (obstacle constraints) - linear velocity (time optimal) (J2)

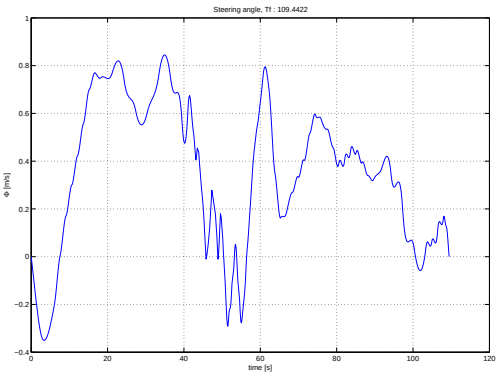


Figure 3.64: Row parking failure (obstacle constraints) - steering angle (time optimal) (J2)

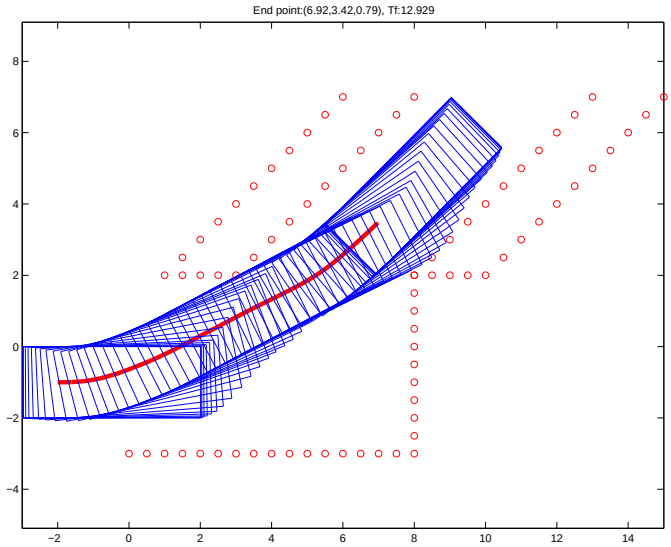


Figure 3.65: Slide parking (obstacle constraints) - robot motion (time optimal) (J2)

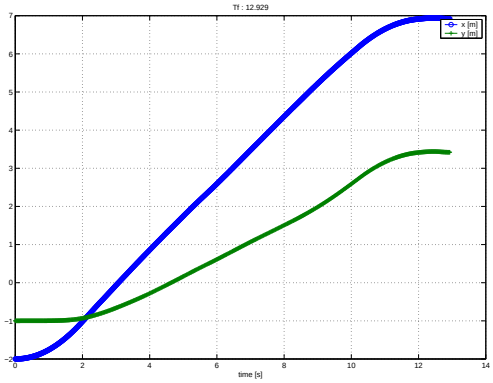


Figure 3.66: Slide parking (obstacle constraints) - xy coordinates (time optimal) (J2)

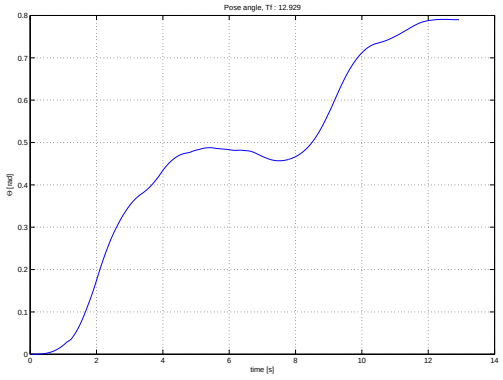


Figure 3.67: Slide parking (obstacle constraints) - vehicle heading (time optimal) (J2)

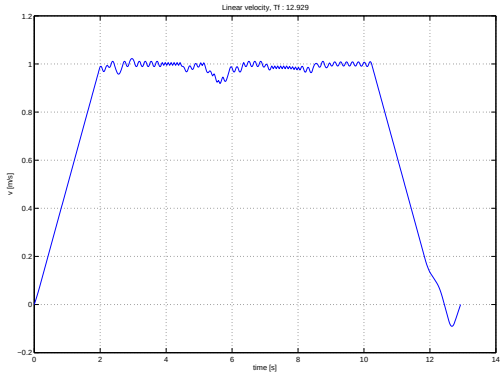


Figure 3.68: Slide parking (obstacle constraints) - linear velocity (time optimal) (J2)

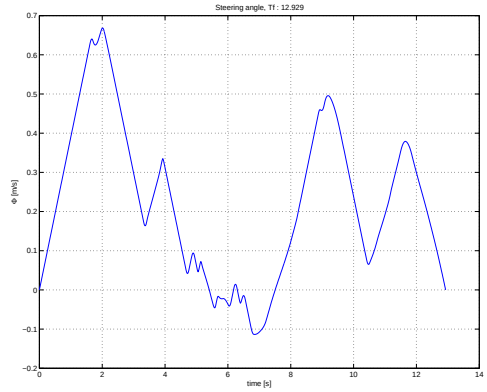


Figure 3.69: Slide parking (obstacle constraints) - steering (time optimal) (J2)

3.5 Conclusion

The presented chapter dealt with autonomous navigation approaches applicable to static, unstructured environments. The hierarchical navigation scheme presented in Sec. 3.2 is complete in the sense that it contains all the necessary levels of deliberation from global goal objective to reactive avoidance level and was well tested experimentally in parking lot conditions.

The motivation behind the development of a probabilistic obstacle avoider of Sec. 3.3 or the motion planning using optimization techniques in Sec. 3.4 was to explore the possibilities of very generic dynamic system description and inference engines within the scope of autonomous vehicle navigation. The presented results are limited to very simple tasks such as instantaneous obstacle avoidance or parking maneuvers, however, the inclusion of more complex tasks should be fairly straightforward due to the flexibility of the system description while taking into account the additional computation burden.

Chapter 4

Motion planning in dynamic environments

4.1 Introduction

Assuming the presence of moving obstacles with arbitrary dynamics, a motion planning technique is to be proposed that would enable the navigation of the ego-vehicle from a start to a goal configuration. The motion planning consists in an incremental trajectory search from the start to goal configuration, given the kino-dynamic constraints of the ego-vehicle itself and the motion of the obstacles in the surrounding environment/workspace. The main distinction to the motion planning problem in a static environment is the constraint that at each instant of time the trajectory of the ego-vehicle must be collision-free with respect to the static as well as dynamic obstacles. From the implementation point of view, this implies that the whole ego-vehicle trajectory must be discretized in time and checked against collision against each of static as well as dynamic objects. The aim of this chapter is to derive motion planning strategies in presence of dynamic obstacles that would comply to the timing structure regarding the motion prediction, motion planning and execution proposed here. Furthermore, explore different possibilities of the trajectory exploration phase in the sense of state branching techniques (trajectory

diffusion) and directedness towards the goal objective in form of cost functions. Furthermore, given the ego-vehicle motion planning scheme and various levels of dynamic obstacle movement, different motion safety levels can be identified and issued as safety guarantees.

Firstly, the general time constraints for motion planning in dynamic environments are identified and analyzed with respect to each other in Sec. 4.2.

Sec. 4.3 proposes motion planning solutions for the ego-vehicle whose motion is described in the transformed state space. This enables the description of dynamic obstacle traces in time and ego-vehicle trajectory increments in the trajectory exploration phase both linear which in turn renders the collision checks against many moving obstacles and trajectory exploration updates fast. Under constrained ego-vehicle motion along one transformed state axes, the final time of arrival at the goal configuration can be predicted, enabling a bidirectional trajectory search from start and goal configuration simultaneously. The proposed motion planning technique in analyzed in a generic dynamic environments with many moving objects in Sec. 4.3.5. An unconstrained version of this motion planning technique that performes a randomized search also in the longitudinal velocity space is analyzed in Sec. 4.3.6 and applied to a urban dynamic scenario problem.

A generic motion planning scheme that is independent of a particular ego-motion model is presented in Sec. 4.4, where the proposed motion planning scheme can be applied to any dynamic model of the ego-platform as well as the predicted motion of the dynamic obstacles and static configuration of the environment. All the timing constraints for a generic motion problem in a dynamic environment of Sec. 4.2 are inherently integrated in the scheme. Different trajectory exploration/diffusion techniques are explored along with the integration of global navigation level information about the static part of the environment structure.

Sec. 4.5 defines possible levels of motion planning safety each with an increasing guaranty with respect to the ego-motion and dynamic obstacles movement deliberation. In particular, safety guaranty of the proposed motion planning scheme in

Sec. 4.4 is analyzed in detail. Formulating clear safety guaranty levels determine the bounds of utilization of an autonomous vehicle platform.

4.2 Time constraints in dynamic environments

Since time is the key variable in the motion planning problem in dynamic environments, there are several important time variables intrinsically associated with it. The first is the motion planning horizon T_m, which denotes how far into the future the motion of the ego-vehicle has been explored (the plan). The second is the obstacle motion prediction horizon T_p, which denotes how far into the future the motion of the environmental obstacles have been modeled and predicted (the perception). Clearly, the fundamental requirement is that $T_m \leq T_p$, that is one can only plan so far into the future as the environment evolution prediction allows for it.

Thirdly, another important aspect is the time allowed for computation of future ego-vehicle motions, here called the decision time T_d. Regardless of the computational details (whether the motion planning is a dedicated task run in parallel or executed in succession with respect to other running tasks), the fundamental limitation is $T_d \leq T_m$. Assuming that the motion planning task is run in cycles, i.e. re-planning cycles, then the current motion planning cycle's control execution is based on the previous cycle's planning result. This further implies that if the execution of the previous motion plan ("planned in the past for the current time") lasts T_m of time, than the decision time T_d cannot exceed the motion prediction horizon itself ("otherwise the ego-vehicle would run out of a valid plan for the next execution cycle"). Note however that there is no lower bound for the decision time, in fact it can be that $T_d \ll T_m$, depending on the complexity of the motion planning exploration itself. Fourthly, there is a perception update cycle time T_s, which gives a global update on the motion prediction of the moving obstacles, i.e. change in their future trajectories.

In an ideal world, the motion of the ego-vehicle could be predicted without any uncertainty up to the motion planning horizon T_m. This holds true also for the future trajectories of the moving obstacles up to the motion prediction horizon T_p. Whereas, the former assumption could be assumed valid if a high-fidelity motion model for the ego-vehicle is available (since motion prediction equals the computed motion plan here), the latter assumption is more critical. Motion prediction for the environment obstacles underlies inherently to greater uncertainty due to the fact they are in principle not controllable by the ego-vehicle. Therefore, it is sensible to set the perception update cycle smaller than the motion prediction horizon, i.e. $T_s < T_p$. For instance, in an urban-like traffic environment the motion of the participating vehicles, pedestrians, cyclists could be predicted up to $T_p \approx 5\,\text{sec}$, whereas the perception update cycle that recomputes the trajectory assumptions based on the new sensory data could be set at $T_s \approx 1\,\text{sec}$. Clearly, these values depend strongly on the dynamicity of the environment itself. As a consequence of the environment modeling uncertainty, the execution duration of the current motion plan, notated T_e, beyond T_s may not be sensible, due to the fact that the obstacle trajectories may have unpredictably changed "in-the-meantime", invalidating the motion plan which was originally computed up to motion planning horizon T_m.

Environmental modeling uncertainty therefore imposed an addition limitation to the motion planning problem, namely the control execution of the current motion plan cannot exceed the perception update cycle, in fact is equal to $T_e = T_s$ (assuming unpredictable, unmodeled environmental changes could happen). The restriction on the execution duration T_e further imposes that the decision time T_d (time to compute a new motion plan) be $T_d \leq T_e$. In fact, if the computation of the motion planning is run parallel as an independent task with respect to other computational processes it is beneficial to set $T_d = T_e$ in order to evaluate as much as possible solutions in the motion exploration phase. Therefore, it is important to conclude that the decision-execution cycle and the perception cycle are all run

in synchronous mode, yielding $T_d = T_e = T_s$.

In addition to the above, it should be stated that all of the motion processes are naturally continuous in time, but are emulated at discrete integration steps according to standardized numerical techniques. Additionally, since the collision checking can in general not be solved analytically, a time discretization T_l is introduced for all the trajectories in order to resolve the geometric relations of the ego-vehicle with respect to obstacles at each instant, as mentioned earlier. Typically, T_l is much smaller than other time constants introduced above. Moreover, the low-level control execution of the nominal trajectory computed for the ego-vehicle implies other (shorter) time cycles which will be omitted here for simplicity.

In summary, the timing relations identified above allow for a very generic description of the motion planning problem in the dynamic environments, that takes into account the interplay between motion prediction, motion planning and motion execution. Different computational implementations (parallel, serial) can be described by the time constants relations as well as the overall modeling uncertainty in the form of time limit constraints. The paradigm could be resumed as "predict/plan as far as possible" but "re-observe/execute only close enough" according to certainty limitations. A question that might arise is, why it may be important to predict and plan beyond T_s and T_e, respectively. The answer concerns primarily the completeness of the motion planning problem with respect to the goal objective and the fact that the uncertainty increases with respect to time. Therefore, an far-sighted uncertain motion plan that potentially directs the ego-vehicle to the goal is better than a plan that considers all future motions as equally likely or unlikely beyond a given time limit.

4.3 Motion planning in transformed state space

4.3.1 Kinematic modeling and chained form state-space transformation

Car-like kinematic model can be described by configuration coordinates $q = [x\, y\, \theta\, \phi]$, where the Cartesian coordinates (x, y) correspond to the midpoint between front and rear axle centers which are a separated by a length l, vehicle orientation θ and ϕ being the steering angle. Denoting ρ the wheel radius, u_1 the angular velocity of the rear driving wheels, u_2 the steering rate of the front wheels, then the kinematic car-like model is:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} \rho\cos(\theta) - \frac{\rho}{2}\tan(\phi)\sin(\theta) & 0 \\ \rho\sin(\theta) + \frac{\rho}{2}\tan(\phi)\cos(\theta) & 0 \\ \frac{\rho}{l}\tan(\phi) & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} . \tag{4.1}$$

According to [Laumond, 1998] and [Qu *et al.*, 2004] the kinematic model of Eq. 4.1 can be transformed into the canonical chained form where derivatives of transformed states depend only on lower states in the chain and the inputs:

$$\begin{aligned} \dot{z}_1 &= v_{c1}\,, \qquad \dot{z}_3 = z_2 v_{c1}\,, \\ \dot{z}_2 &= v_{c2}\,, \qquad \dot{z}_4 = z_3 v_{c1}\,, \end{aligned} \tag{4.2}$$

where the configuration coordinates $q = [x\, y\, \theta\, \phi]$ and inputs u_1, u_2 are transformed to new state space of $z = [z_1, z_2, z_3, z_4]$ and inputs v_{c1}, v_{c2}:

$$\begin{aligned} z_1 &= x - \tfrac{l}{2}\cos(\theta)\,, \quad & z_2 &= \tfrac{\tan(\phi)}{l\cos^3(\theta)}\,, \\ z_3 &= \tan(\theta)\,, & z_4 &= y - \tfrac{l}{2}\sin(\theta)\,, \\ u_1 &= \tfrac{v_{c1}}{\rho\cos(\theta)}\,, \\ u_2 &= -\tfrac{3\sin(\theta)}{l\cos^2(\theta)} sin^2(\phi) v_{c1} + l\cos^3(\theta)\cos^2(\phi) v_{c2}\,. \end{aligned} \tag{4.3}$$

The state and control transformation of Eq. 4.3 is valid for angles $\theta \neq \pm\pi/2$.

This singularity is not a hard limitation, since the $x-y$ plane can be pre-rotated for guaranteeing $\theta(t) \in (-\pi/2\,, \pi/2)$ or by introducing an intermediate goal position.

The transformed states z_2 and z_3 can be further expressed as:

$$z_3 = \frac{\mathrm{d}z_4}{\mathrm{d}z_1}\,, \qquad z_2 = \frac{\mathrm{d}^2 z_4}{\mathrm{d}z_1^2}\,. \tag{4.4}$$

4.3.2 Collision avoidance criterion for moving obstacles

The criterion for collision avoidance between the vehicle and a single moving obstacle is adopted from [Qu *et al.*, 2004]and is exposed here as follows:

- The physical limits of the vehicle are denoted by a 2D circle of radius R at the center $O(t) = (x(t)\,, y(t))$. The choice of the reference point being on the midpoint of the connecting line between the axes renders the R size less conservative. The translational vehicle velocity to be determined is $v_r = [\dot{x}(t)\, \dot{y}(t)]^\top$.
- A set of $i = 1, \ldots, n_o$ objects is represented by circles at points $O_i(t)$ and radius r_i denoted by $\mathcal{O}_i\,(O_i(t)\,, r_i)$ and moving at linear velocities $v_i(t)$.

The state of the environment in terms of obstacle positions and directions of movement is resampled at a rate T_s. Starting at sample instant k, the ith obstacle's description in the interval $t \in [kT_s\,, (k+1)T_s]$ is assumed to be evolving linearly from position $O_i = \left(x_i^k\,, y_i^k\right)$ with velocity $v_i^k = \left[v_{i,x}^k\, v_{i,y}^k\right]^\top$.

The relative vehicle velocity to the ith obstacle is defined as:

$$v_{r,i}^k = v_r - v_i^k = \begin{bmatrix} v_{r,i,x}^k \\ v_{r,i,y}^k \end{bmatrix} = \begin{bmatrix} \dot{x} - v_{i,x}^k \\ \dot{y} - v_{i,y}^k \end{bmatrix}\,. \tag{4.5}$$

Taking into account the physical sizes of the vehicle and the ith obstacle, the avoidance criterion to a single obstacle can be formulated as:

$$\left(x_i' - x_i^k\right)^2 + \left(y_i' - y_i^k\right)^2 \geq (r_i + R)^2\,, \tag{4.6}$$

where $x'_i = x - v^k_{i,x}\tau$ and $y'_i = y - v^k_{i,y}\tau$, with τ being $\tau = t - kT_s$.

The avoidance relation according to Eq. 4.6 is based on relative vehicle motion to a virtually static ith obstacle. The minimum distance between the two is a radius of $(r_i + R)$ whenever the obstacle's x^k_i coordinate is in the following interval in the $x - y$ plane:

$$x^k_i \in [x'_i - r_i - R, x'_i + r_i + R] \ . \tag{4.7}$$

In order to obtain the avoidance criterion for the transformed state space, the relations from Eq. 4.3 combining variable x with z_1 and y with z_4 are used. Therefore, the avoidance criterion of Eq. 4.6 can be described in the transformed plane $z_4 - z_1$ as:

$$\begin{aligned} &\left(z_1 + \frac{l}{2}\cos\theta - v^k_{i,x}\tau - x^k_i\right)^2 + \\ &+ \left(z_4 + \frac{l}{2}\sin\theta - v^k_{i,y}\tau - y^k_i\right)^2 \geq (r_i + R)^2 \ , \end{aligned} \tag{4.8}$$

or conversely in the relative motion plane $z'_{4,i} - z'_{1,i}$ as:

$$\begin{aligned} \left(z'_{1,i} + \frac{l}{2}\cos\theta - x^k_i\right)^2 + \left(z'_{4,i} + \frac{l}{2}\sin\theta - y^k_i\right)^2 \\ \geq (r_i + R)^2 \ , \end{aligned} \tag{4.9}$$

where:

$$z'_{1,i} = z_1 - v^k_{i,x}\tau \ , \ z'_{4,i} = z_4 - v^k_{i,y}\tau \ . \tag{4.10}$$

The criterion of Eq. 4.9 represents a snapshot at time t (or τ) and sample interval k of the relative position of the vehicle with transformed coordinates $\left(z'_{1,i}\,, z'_{4,i}\right)$ to a virtually static obstacle at position $O^k_i = \left(x^k_i\,, y^k_i\right)$ in the interval:

$$x^k_i \in \left[z'_{1,i} + \frac{l}{2}\cos\theta - r_i - R\,, z'_{1,i} + \frac{l}{2}\cos\theta + r_i + R\right] \tag{4.11}$$

This criterion includes dependence on the current vehicle orientation θ which can be undesirable if the orientation of the vehicle in the planning stage is not known beforehand. By further geometric inspection an orientation independent criterion was developed in [Qu *et al.*, 2004] only in terms of relative transformed variables $z'_{1,i}$, $z'_{4,i}$, vehicle and obstacles sizes.

Firstly, as the relative motion variables x'_i and y'_i can be expressed with the relative transformed variables $z'_{1,i}$, $z'_{4,i}$ as:

$$x'_i = z'_{1,i} + \tfrac{l}{2}\cos(\theta)\,, \quad y'_i = z'_{4,i} + \tfrac{l}{2}\sin(\theta) \tag{4.12}$$

and the fact that the vehicle orientation angle θ can only take values $\theta \in (-\pi/2\,, \pi/2)$, then it follows that the points $(x'_i\,, y'_i)$ can only be located at the right semicircle located at $\left(z'_{1,i}\,, z'_{4,i}\right)$ with radius $\frac{l}{2}$ in the $z'_{4,i} - z'_{1,i}$ plane.

Furthermore, the Eq. 4.6 gives the condition on minimum distance of $(r_i + R)$ between the possible relative vehicle point $(x'_i\,, y'_i)$ and the ith obstacle's virtual static position $\left(x_i^k\,, y_i^k\right)$. Therefore, by drawing circles of radius $(r_i + R)$ at all possible $(x'_i\,, y'_i)$ loci, a circular prohibitive region Υ_i for i*th* obstacle center $O_i^k = \left(x_i^k\,, y_i^k\right)$ is defined and is of radius $\tilde{R} = \left(r_i + R + \frac{l}{2}\right)$, giving the final vehicle to single ith obstacle collision avoidance criterion:

$$\left(z'_{1,i} - x_i^k\right)^2 + \left(z'_{4,i} - y_i^k\right)^2 \geq \left(r_i + R + \frac{l}{2}\right)^2 \tag{4.13}$$

with no vehicle orientation θ dependence. Because of the limited θ range, the potential collision region for the $z'_{1,i}$ axis is defined as:

$$x_i^k \in \left[z'_{1,i} - r_i - R\,, z'_{1,i} + \frac{l}{2} + r_i + R\right]\,. \tag{4.14}$$

A criterion for the $z'_{4,i}$ axis that considers any possible orientation θ, results in a rectangular region Ξ_i for an i-th obstacle:

$$y_i^k \in \left[z'_{4,i} - \frac{l}{2} - r_i - R \, , z'_{4,i} + \frac{l}{2} + r_i + R \right] . \tag{4.15}$$

for the $z'_{4,i}$ axis.

Although this modified collision avoidance criterion neglects some allowed robot configurations, it renders the testing of collision-free regions linear, compared to the circular prohibitive region of Eq. 4.13 which is quadratic in terms of inequality testing. This facilitates computation, which can be important in presence of many moving obstacles.

4.3.3 Trajectory representation

To generate a feasible trajectory, i.e. a trajectory that complies to the kinematic model of Eq. 4.1 that involves nonholonomic constraints and that is collision-free, a certain family of curves is assumed. In [Qu *et al.*, 2004] this family were polynomial curves where it was stipulated that the variable z_4 is dependent on z_1. In this work, the family of curves chosen are B-splines [de Boor, 2001], since they enable better local control of the trajectory shape. Their local controllability/flexibility may be important in confined environments, so problems like large detours can be avoided.

Using the chained form transformation of Eq.4.2 for the vehicle kinematic model allows to describe the variable z_4 in a direct functional relation to z_1:

$$z_4(t) = b(z_1(t)) = \sum_{j=0}^{m} B_j N_{j,d}(z_1(t)) \, , \tag{4.16}$$

where B_j are $m+1$ spline control points, N_j^d denote the basis functions of degree d defined by a knot sequence $z_{1_0} \leq z_{1_1} \leq \cdots \leq z_{1_{m+d+1}}$. The variable z_1 represents the spline parameter that can take any real value in the range $\left[z_{1_0}, z_{1_{m+d+1}} \right]$.

In order to calculate the state variables z_2 and z_3, the B-spline curve of degree

$d = 4$ is derived as:

$$\begin{aligned} z_3 &= \tfrac{\mathrm{d}z_4}{\mathrm{d}z_1} = b'(z_1) = \textstyle\sum_{j=0}^{m} B_j N'_{j,d}(z_1)\,, \\ z_2 &= \tfrac{\mathrm{d}^2 z_4}{\mathrm{d}{z_1}^2} = b''(z_1) = \textstyle\sum_{j=0}^{m} B_j N''_{j,d}(z_1)\,, \\ \dot{z}_2 &= \tfrac{\mathrm{d}^3 z_4}{\mathrm{d}{z_1}^3} \cdot \dot{z}_1 = b'''(z_1) \cdot \dot{z}_1 = \textstyle\sum_{j=0}^{m} B_j N'''_{j,d}(z_1) \cdot \dot{z}_1\,. \end{aligned} \tag{4.17}$$

Each B-spline curve has a control polygon associated to it, which is a polyline with vertices defined as spline control points B_j. In general, the B-spline $b(z_1(t))$ curve may or may not pass through the control points, depending on control points alignment and knot vector (spline parameter) distribution. The B-spline curve implementation in this work takes initial and final robot configurations exactly into account. From the recursion relations for the basis functions and derivatives [de Boor, 2001] this implies that the control points B_j, $j = 0, \ldots, 3$ and $j = m-3, \ldots, m$ are determined from boundary conditions.The degree of the B-spline curve is determined as the minimum degree curve where the third degree derivations of the basis functions of Eq. 4.17 are still continuous, hence $d = 4$.

At each point on the B-spline curve given by the knot vector (spline parameter) z_{1_i}, there are exactly $d + 1$ control points B_j that locally determine the curve's shape, contained in the polygon spanned by these control points. The search for a feasible trajectory in general involves determining both position of the control points and the discretization/enumeration of the knot vector. The value of the knot vector z_{1_i} where a particular control point B_j exerts maximum influence is given by the so called Greville abscissae [de Boor, 2001]:

$$z^*_{1_j} = \sum_{i=j+1}^{j+d} \frac{z_{1_i}}{d}\,. \tag{4.18}$$

Collocation (discretization) of the B-spline curve based on these knot vector values enables a minimalistic knot vector representation which can be calculated a-priori, whereas on the other hand enabling the maximum influence on the B-spline curve's shape by the control points B_j, which are in our case the unknowns

to be determined by the search for a feasible trajectory.

As will be described in more detail in Sec. 4.3.5 and Sec. 4.3.6, the control points B_j in this work represent vertices of a RRT tree (Rapidly-Exploring Random Tree). The RRT-s were first introducted in [Kuffner and LaValle, 2000], where the state transitions representing the leafs of the tree are determined as direct robot state integration, given a current control input depending on the motion model used. In this work, however, the transitions between the states (leafs) are linear segments that rather determine the control polygon of the spline then the curve itself. This enables a much faster calculation of the tree expansion during the search phase, since all the expansions are linear segments. Moreover, the collision checking with the obstacle traces which are polygonal segments according to Sec. 4.3.2 and these linear expansion is extremely fast.

4.3.4 Motion planning problem

The motion planning problem can be stated as problem of moving the vehicle from an initial configuration $q_0 = \left[x_0, y_0, \theta_0, \phi_0, \dot{\phi}_0\right]$ to a final configuration $q_f = \left[x_f, y_f, \theta_f, \phi_f, \dot{\phi}_f\right]$ while avoiding the linearly moving obstacles $\mathcal{O}_i\left(O_i(t), r_i\right)$, $i = 1, \ldots, n_o$, as described in Sec. 4.3.2. The configuration boundary conditions define the set of transformed state boundary conditions $\left[z_1^k, \dot{z}_2^k, z_2^k, z_3^k, z_4^k\right]$ in $k = 0, \ldots, N_s$ global replanning sample intervals, where final time is $T_f = N_s \cdot T_s$.

The task of finding the vehicle trajectory is divided into three phases:

- defining potential collision areas with obstacles;
- performing a RRT search expansion within the obstacle free regions of the transformed space;
- smoothing of the linear segment based path with B-splines to obtain the final smooth trajectory.

The search for a feasible path with RRT expansion in the transformed state space spanned by $z_4 - z_1$ plane can be considered as an analogy to a $y - x$ plane

search. A tree leaf of the RRT is a line segment that represents a path increment, where p, q are connected tree nodes indices. Due to dynamic obstacles, the collision regions in $z_4 - z_1$ plane where the tree leafs cannot be expanded, depend directly on the vehicle motion and therefore control inputs beforehand, so that this space-time combination results in a $3D$ search problem.

Nevertheless, the evolution of the state z_1 can be analyzed independently from the rest of the vehicle configuration, due to chained transform (Eq. 4.2) where $\dot{z}_1$ is an input control variable. If the evolution of $z_1(t)$ is known by some defined profile, the time component is embedded in z_1 and a pure $2D$ geometric search can be performed with moving obstacle traces in $z_4 - z_1$ plane related to their respective velocities and z_1 component. This fact is used in Sec. 4.3.5 and 4.3.6 where a bidirectional and a unidirectional trajectory search is performed in the transformed space, respectfully.

The bidirectional trajectory search case [Macek and Siegwart, 2006], where a RRT tree is expanded from the starting and the goal configuration simultaneously assumes an a-priori know profile on $\dot{z}_1$, therefore also the final time T_f is a fixed parameter. Eventhough this trajectory search version affects both linear and rotational velocity of the robot, due to the nature of chained form of Eq. 4.2 and Eq. 4.3, the control input $\dot{z}_1$ primarily influences the linear robot velocity. If $\dot{z}_1$ is then kept constant, as will be elaborated further in Sec. 4.3.5, the bidirectional method can be considered as a form of steering navigation method.

The unidirectional trajectory search case [Macek *et al.*, 2006], in Sec. 4.3.6 makes no a-priori assumption on the profile of the transformed input $\dot{z}_1$, which has to be determined during the trajectory search. Consequently the final time T_f is a free parameter, the search is performed only from the starting configuration using incremental expansion of one RRT tree. The resulting trajectory is general in the sense that the longitudinal and rotational robot velocities are independent.

4.3.5 Bidirectional trajectory search

4.3.5.1 Control input time profile

To obtain a collision-free movement of the robot, it has to be guaranteed at each instant t that the avoidance condition discussed in Sec. 4.3.2 for a single obstacle be fulfilled for all moving obstacles. In order to predict the possible collision zones with obstacles for all future times τ, where $\tau \in [kT_s, T_f]$, the main assumption introduced in [Qu *et al.*, 2004] was that the transformed velocity along the z_1 axis is kept constant:

$$v_{c1}(t) = C = \frac{z_1^f - z_1^0}{T_f}\,. \tag{4.19}$$

According to the chained form transformation of Eq. 4.2 this effectively means that the state z_1 can be analyzed independently from the rest of the robot transformed configuration.

Since the robot is on a constant forward move along the z_1 axis and due to the fact that obstacles move in a linear fashion on a kth sample interval, the robot can collide with an ith obstacle only on *one* interval from its current configuration to the goal. Naturally, this is based on condition that the state of the environment, i.e. the obstacle velocities and their directions remain unchanged from time interval kT_s to T_f. If at sample interval $k+1$ the environment conditions change, the possible collision intervals have to be revised.

4.3.5.2 Potential collision areas with obstacles

To determine the time limits of these collision intervals, the Eq. 4.14 can be used since it implicitly contains the temporal information in the variable $z'_{1,i}(\tau) = z_1(\tau) - v_{i,x}^k \tau$, where $\tau = t - kT_s$. Assuming there exist $n_o^k \leq n_o$ obstacles which can collide with the robot on the interval $\tau \in [kT_s, T_f]$, the exact time boundaries $\tau_{i,1}^k \leq \tau_{i,2}^k$, $i = 1, \ldots, n_o^k$ can be calculated in the region where the robot and the ith obstacle share the same z_1 transformed coordinate. According to Eq. 4.14 it

can be written:

$$\begin{aligned} \tilde{\tau}_{i,1}^k &= \frac{z_1(\tilde{\tau}_{i,1}^k) - r_i - R - x_i^k}{v_{i,x}^k}, \\ \tilde{\tau}_{i,2}^k &= \frac{z_1(\tilde{\tau}_{i,2}^k) + \frac{l}{2} + r_i + R - x_i^k}{v_{i,x}^k}. \end{aligned} \tag{4.20}$$

Since the z_1 coordinate is dependent only on constant transformed velocity input v_{c1} it can be expressed as:

$$\begin{aligned} z_1(\tilde{\tau}_{i,1}^k) &= z_1^0 + C \cdot (kT_s + \tilde{\tau}_{i,1}^k), \\ z_1(\tilde{\tau}_{i,2}^k) &= z_1^0 + C \cdot (kT_s + \tilde{\tau}_{i,2}^k) \end{aligned} \tag{4.21}$$

and it follows that:

$$\begin{aligned} \tilde{\tau}_{i,1}^k &= \frac{z_1^0 + CkT_s - r_i - R - x_i^k}{v_{i,x}^k - C}, \\ \tilde{\tau}_{i,2}^k &= \frac{z_1^0 + CkT_s + \frac{l}{2} + r_i + R - x_i^k}{v_{i,x}^k - C}, \end{aligned} \tag{4.22}$$

where $v_{i,x}^k \neq C$. According to the obstacles direction, the time intervals $\tau_{i,1}^k$ and $\tau_{i,1}^k$ are obtained as:

$$\tau_{i,1}^k = min\left\{\tilde{\tau}_{i,1}^k, \tilde{\tau}_{i,2}^k\right\} \quad , \quad \tau_{i,2}^k = max\left\{\tilde{\tau}_{i,1}^k, \tilde{\tau}_{i,2}^k\right\} \ . \tag{4.23}$$

Such decoupling of one state variable z_1 reduces the search for a collision-free path among all moving obstacles from a geometric space-time combination ($2D + 1D = 3D$) to only geometric space search ($2D$). In [Qu *et al.*, 2004] an analytic solution to the collision avoidance problem was thereafter developed based on recursive calculation of polynomial coefficients for each sample interval k. The collision-free area was defined by a set of $n_o^k \leq n_o$ quadratic inequalities based on Eq. 4.13 for the obstacles sharing the same z_1 transformed coordinate with the robot.

Although, the polynomial approach was appealing since it provided a closed-form solution to the trajectory generation, the sensitivity of the path shape to only one coefficient proved to be important in our simulations. In particular, by a small change in its value, or a choice of a different sign (as an alternative solution to the quadratic inequalities), the path was changed importantly and could exhibit large detours with respect to a nominal "straight-line" path to the goal. As mentioned earlier in Sec. 4.3.3, in order to remedy this problem, the B-spline curve family introduced here (Eq. 4.16) enables better local control of the trajectory shape. However, the advantage of the state decomposition and clear separation of geometric and time criterion for collision avoidance as presented in [Qu *et al.*, 2004] is still preserved.

To account for all moving obstacles, the approach taken here is to combine all prohibitive collision areas defined by the rectangular areas Ξ_i, $i = 1, \ldots, n_o^k$, for a sample interval k as discussed in Sec. 4.3.2 for every $\tau \in \left[\tau_{i,1}^k, \tau_{i,2}^k\right]$ and calculate the prohibited regions in the $z_4 - z_1$ plane. Each obstacle that potentially collides with the robot leaves a "trace" in the $z_4 - z_1$ in the shape of a parallelogram defined by the points $(z_{1,i,1}^k, z_{4,i,1}^k)$, $(z_{1,i,1}^k, z_{4,i,2}^k)$, $(z_{1,i,2}^k, z_{4,i,3}^k)$ and $(z_{1,i,2}^k, z_{4,i,4}^k)$ as:

$$
\begin{aligned}
z_{1,i,1}^k &= z_1^k + C \cdot \tau_{i,1}^k\,, \qquad z_{1,i,2}^k = z_1^k + C \cdot \tau_{i,2}^k\,,\\
z_{4,i,1}^k &= y_i^k - v_{i,y}^k \tau_{i,1}^k - \tfrac{l}{2} - r_i - R\,,\\
z_{4,i,2}^k &= y_i^k - v_{i,y}^k \tau_{i,1} + \tfrac{l}{2} + r_i + R\,,\\
z_{4,i,3}^k &= y_i^k - v_{i,y}^k \tau_{i,2} - \tfrac{l}{2} - r_i - R\,,\\
z_{4,i,4}^k &= y_i^k - v_{i,y}^k \tau_{i,2} + \tfrac{l}{2} + r_i + R\,.
\end{aligned}
\tag{4.24}
$$

4.3.5.3 Bidirectional RRT expansion

By introducing the B-spline curves, there is no unique solution to the trajectory generation problem, since there can be many degrees of freedom as to the choice of the spline control points B_j and knot vector. Therefore, the control points to the B-spline are found by Rapidly-exploring Random Trees (RRT) based on

[LaValle and Kuffner, 2001] for the geometric space spanned by the $z_4 - z_1$ plane and the prohibited areas included. The start and end control points are identical to the $z_{init} = (z_1^k, z_4^k)$ and $z_{goal} = (z_1^f, z_4^f)$ configurations. Since the coordinate z_1 already represents the knot parameter, a $1D$ spline approximation is required. A bidirectional RRT search is employed here, with one tree $\mathcal{T}_a$ starting from the (z_1^k, z_4^k) configuration and the $\mathcal{T}_b$ starting from the goal configuration.

The trees are expanded through the EXTEND function in a random z_{rand} direction from the nearest z_{near} point on the tree obtained by NEW_NEIGHBOR function. An increment from z_{near} along the z_1 axis is calculated based on the Greville abscissae from Eq. 4.18. The possible new state is tested for collision against all obstacles in the NEW_STATE. Collision-free point z_{new} is connected to z_{new_conn}, if one exists, on the opposite tree, at a max distance $z_{1_j}^*$ along z_1, where j represents the relative position from the root node of the tree.

A separate list of connected leaves $\mathcal{L}_{a,b,cost}$ from both $\mathcal{T}_a$ and $\mathcal{T}_b$ is kept. Along the connections, the Euclidean traversal cost from the root parent node (z_1^k, z_4^k) of $\mathcal{T}_a$ to the goal root node $z_{goal} = (z_1^f, z_4^f)$ of $\mathcal{T}_b$ is kept, which enables choosing the connected path between the two trees defined by the connected leafs $l_{a,b}^\star = \min_{cost}\{\mathcal{L}_{a,b,cost}\}$ with shortest overall distance on their corresponding backpointer nodes along the path.

The RRT search results in a polyline polygon $z_4 = l(z_1)$ connecting the MIN_PATH based tree leaf vertices. These tree leaf vertices represent the B-spline control points $\mathcal{B} = \{B_0, \ldots, B_m\}$ of the spline control polygon:

$$l(z_{1_j}^*) = B_j\,,\ j = 0, \ldots, m\,. \tag{4.25}$$

from which the final B-spline trajectory can be obtained according to Eq. 4.16.

RRT_BIDIRECTIONAL_SEARCH(z_{init}, z_{goal})
1 $\mathcal{T}_a$.init(z_{init}); $\mathcal{T}_b$.init(z_{goal}) ; $direction = 1$;
2 **for** $k = 1$ **to** K **do**
3 $z_{new} \leftarrow$ EXTEND($\mathcal{T}_a$, z_{rand}, $direction$);
4 **if not** ($z_{new} = \varnothing$) **then**
5 **if not**(CONNECT($\mathcal{T}_b$, z_{new})=*Failed*) **then**
6 $\mathcal{L}_{a,b,cost} \leftarrow \mathcal{L}_{a,b,cost} \cup (z_{new}, z_{new\_conn}, cost)$;
7 $direction \leftarrow$ SWAP($\mathcal{T}_a$, $\mathcal{T}_b$);
8 **end**
9 **if not** $\mathcal{L}_{a,b,cost} = \varnothing$ **then**
10 **return** MIN_PATH($\mathcal{T}_a$, $\mathcal{T}_b$);
11 **else return** *Failure*;

EXTEND($\mathcal{T}$, z, $direction$)
1 $z_{near} \leftarrow$ NEAREST_NEIGHBOR(z,$\mathcal{T}$, $direction$);
2 $z_{new} \leftarrow$ NEW_STATE(z, z_{near}, $direction$);
3 **if not** ($z_{new} = \varnothing$) **then**
4 $\mathcal{T} \leftarrow \mathcal{T} \cup z_{new}$;
5 **return** z_{new};
6 **else return** $\varnothing$;

Table 4.1: Bidirectional RRT search in the $z_4 - z_1$ space.

4.3.5.4 Simulation results in a highly populated dynamic obstacle environment

Simulation results show a sequence of snapshots of an environment highly populated with moving obstacles in Fig. 4.1, at the obstacle state and replanning update rate $T_s = 5s$. The initial and final conditions are $x(0) = 0$, $y(0) = -3$, $\theta(0) = -\pi/4$, $\phi(0) = \pi/6$, $\dot{\phi}(0) = 0$, $x(T_f) = 15$, $y(T_f) = 7$, $\theta(T_f) = \pi/6$, $\phi(T_f) = 0$, $\dot{\phi}(T_f) = 0$ and $T_f = 15s$. The obstacles are allowed to randomly change velocity and direction at the sample rate $T_s = 5s$, however, their maximum velocity is limited to the vehicle velocity towards goal $v_{max} = C$, since the vehicle should always be at least as fast as the obstacles in order to prevent

collision. Obstacles that can potentially collide with the vehicle on a certain segment are marked with full line paths (green), others are marked with dashed line paths (blue). The obstacle trajectory traces and search for the ego-vehicle feasible trajectory is depicted in Fig. 4.2.

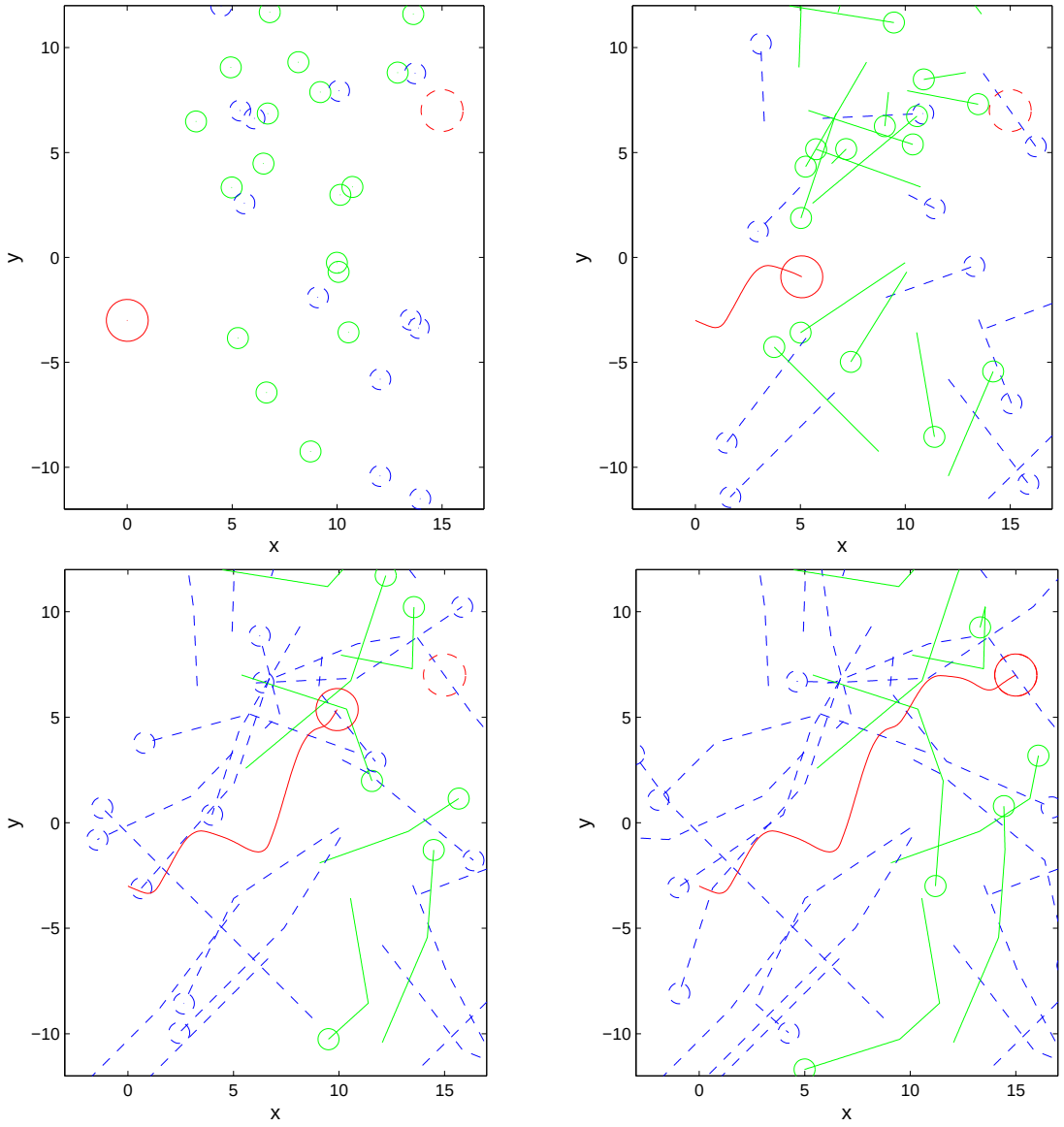


Figure 4.1: Vehicle motion simulation among moving obstacles. Vehicle and obstacle paths are depicted at global replanning instants $t = 0, 5, 10, 15s$, respectively. Obstacle trajectories are re-evaluated for each replanning cycle, where the obstacles that can potentially collide with the vehicle are depicted in solid (green).

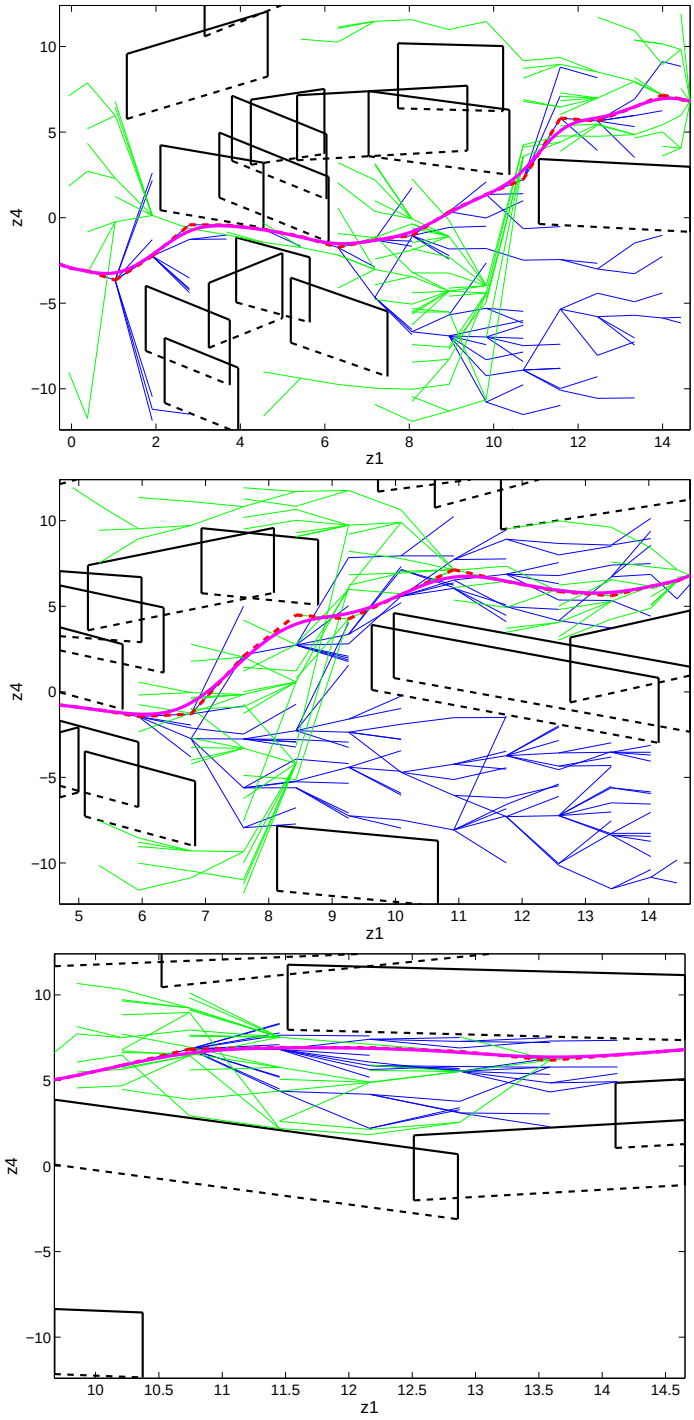
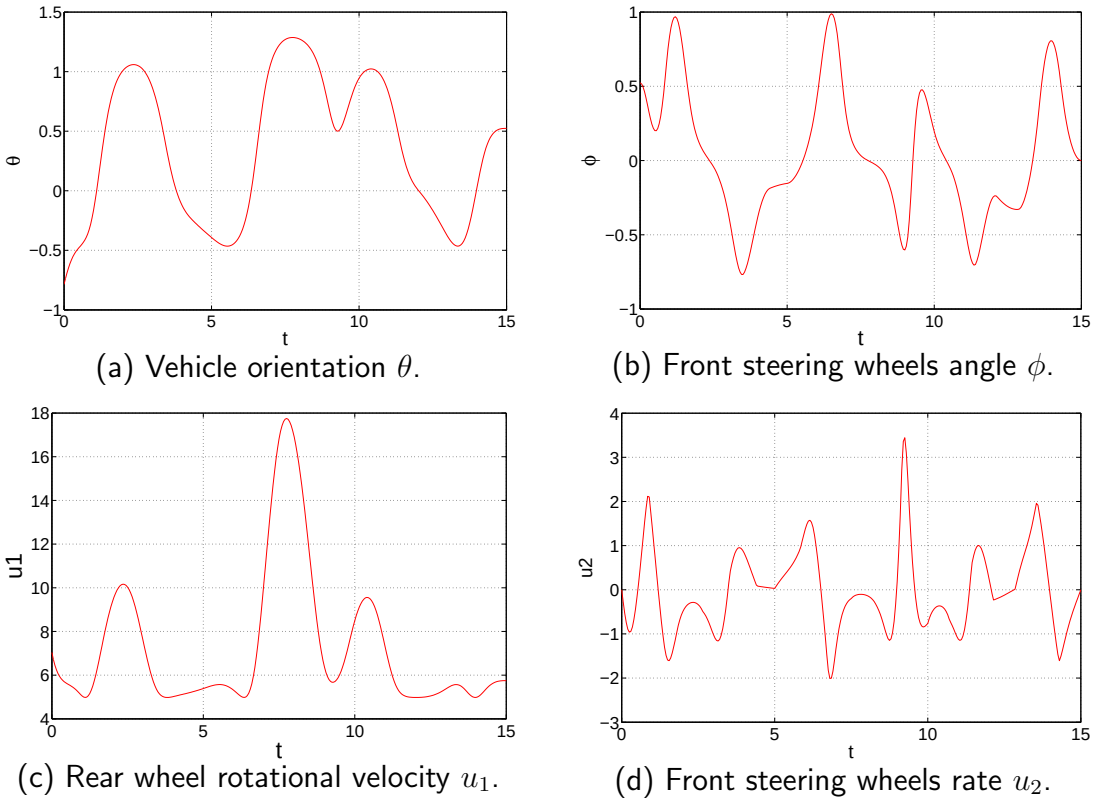


Figure 4.2: Vehicle motion planning among moving obstacles. Bidirectional RRT search (from start and goal configuration) and root B-spline trajectory generation in transformed $z_4 - z_1$ space is depicted at instants $t = 0, 5, 10s$, respectively. Trajectory traces of obstacles on potential collision course are depicted as polygons.



(a) Vehicle orientation θ.

(b) Front steering wheels angle ϕ.

(c) Rear wheel rotational velocity u_1.

(d) Front steering wheels rate u_2.

Figure 4.3: Vehicle state and commands.

Due to the inherent randomness of the obstacles there is no guarantee that a free path exists for the vehicle, however, it was shown through the simulations that a feasible trajectory was found, if one existed, even in a densely cluttered environment. At each resampling instant $k = 0, 1, 2$, the current obstacle motion and the state of the vehicle at the end of the previous cycle was taken into account in order to ensure continuous control inputs, in particular the steering wheels rate u_2 where the third derivative of the B-spline curve $b(z_1)'''$ must be continuous as can be seen on Fig. 4.3.

In summary, the presented section deals with trajectory generation of a non-holonomic platform in an dynamic environment with linearly moving obstacles. Kinematic state transformation introduced in an earlier work enables predicting potential collision with obstacles both in terms of time and geometric criteria. The search of a feasible path among many moving obstacles is here performed as

a bidirectional RRT search in transformed state space and is further smoothed by B-splines, resulting in a smooth, locally adjustable trajectory for the vehicle. The boundary poses (states) of the vehicle can be precisely defined and the control inputs are continuous. Simulation results of vehicle motion in an environment with densely populated obstacles verify the approach.

4.3.6 Unidirectional trajectory search

4.3.6.1 Control input time profile

Let the RRT tree in the replanning phase $t \in [kT_s\,, (k+1)T_s]$ be $\mathcal{T}^k$ and let tree nodes $\mathcal{T}_p^k$, $\mathcal{T}_q^k$ define a connected tree leaf $\mathcal{T}_{p,q}^k = \left\{ \left(z_{1,p}^k, z_{4,p}^k\right) \left(z_{1,q}^k, z_{4,q}^k\right) \right\}$, where $p, q \in [0 \ldots N_{\mathcal{T}^k}]$ and $N_{\mathcal{T}^k}$ total number of connected leafs. The proposed $\dot{z}_{1,p,q}^k$ velocity profile can be expressed as:

$$\dot{z}_{1,p,q}^k(\tau) = \dot{z}_{1,p}^k + \ddot{z}_{1,p,q}^k \cdot \tau\,, \tag{4.26}$$

as linear acceleration phase in $\tau \in [T_{s,p}^k, T_{s,p}^k + T_{a,p,q}^k]$ with $\ddot{z}_{1,p,q}^k$ and then moving forward with a constant speed of:

$$\dot{z}_{1,p,q}^k(\tau) = \dot{z}_{1,q}^k\,, \tag{4.27}$$

in the interval $\tau \in [T_{s,p}^k + T_{a,p,q}^k, T_{s,p}^k + T_{d,p,q}^k]$. For full control search capabilities the acceleration/deceleration $\ddot{z}_{1,p,q}^k$ is randomized enabling full longitudinal velocity control. The acceleration period $T_{a,p,q}^k$ is also randomized changing the width of the trapezoidal velocity shape. The length of the $\mathcal{T}_{p,q}^k$ leaf depends also on the randomized time interval $T_{d,p,q}^k$ on which the velocity profile is applied. This is particularly interesting if the vehicle has to circumvent obstacles and a finer, i.e. smaller leaf length is needed for tree search. It is then evident that the total time $T_{s,p}^k$ up from the tree root to the node $\mathcal{T}_p^k$ depends on the history of profiles along the previous connected nodes. In terms of achieving a goal position, the final time

τ_f^k in k-th resample period is therefore left as a free parameter in the search.

The velocity $\dot{z}_1$ can be expressed also as:

$$\dot{z}_1(\tau) = \dot{x}(\tau) + \frac{l}{2}\sin(\theta(\tau)) \cdot \dot{\theta}(\tau)\,. \tag{4.28}$$

By approximating the motion along leaf $\mathcal{T}_{p,q}^k$ as straight-line motion, the acceleration $\ddot{z}_{1,p,q}^k$ is bound by:

$$\|\ddot{z}_{1,p,q}^k\| = acc_{max} \cdot \cos\theta_{p,q}^k\,, \tag{4.29}$$

where acc_{max} denotes the maximal vehicle acceleration. $\theta_{p,q}^k$ is a constant vehicle orientation along the $\mathcal{T}_{p,q}^k$ leaf, according to Eq. 4.2 and Eq. 4.4 where $z_{3,p,q}(\tau) = \tan(\theta_{p,q}(\tau))$, representing the slope of a RRT line segment. Constant angle approximation is valid if the leaf length and therefore the interval sample time $T_{d,p,q}^k$ is small enough.

4.3.6.2 Potential collision areas with obstacles

To calculate possible prohibited collision regions for a leaf $\mathcal{T}_{p,q}^k$ with obstacles $i = 1, \ldots, n_o^k$, time intervals $\tau \in \left[\tau_{i,p,q,1}^k\,, \tau_{i,p,q,2}^k\right]$ must be calculated. If any of $\tau_{i,p,q,1}^k$ or $\tau_{i,p,q,2}^k$ lies in interval $\tau \in [T_{s,p}^k, T_{s,p}^k + T_{d,p,q}^k]$ the i-th obstacle has a potential collision course with the vehicle.

By combining Eq. 4.10 with velocity profiles of Eq. 4.26 and Eq. 4.27 the following four $z_{1,p,q,1}^k$, $z_{1,p,q,2}^k$, $z_{1,p,q,3}^k$, $z_{1,p,q,4}^k$ potential bound positions are calculated for each i-th obstacle. On the linear acceleration part of the velocity profile it is obtained:

$$\begin{aligned} z_{1,p,q,\{1,2\}}^k(\tilde{\tau}_{i,p,q,\{1,2\}}^k) = \;& z_{1,p}^k + \dot{z}_{1,p}^k \cdot \tilde{\tau}_{i,p,q,\{1,2\}}^k \\ & + \ddot{z}_{1,p,q}^k \cdot \frac{\left(\tilde{\tau}_{i,p,q,\{1,2\}}^k\right)^2}{2} \end{aligned} \tag{4.30}$$

and on the constant part of the velocity profile:

$$z^k_{1,p,q,\{3,4\}}(\tilde{\tau}^k_{i,p,q,\{3,4\}}) = z^k_{1,p} + \dot{z}^k_{1,p} \cdot T^k_{a,p,q} + \ddot{z}^k_{1,p,q} \cdot \frac{(T_{a,p,q})^2}{2} + \left(\tilde{\tau}^k_{i,p,q,\{3,4\}} - T^k_{a,p,q}\right) \cdot \dot{z}^k_{1,q} \,. \tag{4.31}$$

Calculating the collision avoidance interval bounds from Eq. 4.14 yields:

$$\begin{aligned} x^k_i &= z^k_{1,p,q,\{1,3\}}(\tilde{\tau}^k_{i,p,q,\{1,3\}}) - r_i - R \\ x^k_i &= z^k_{1,p,q,\{2,4\}}(\tilde{\tau}^k_{i,p,q,\{2,4\}}) + \frac{l}{2} + r_i + R \,. \end{aligned} \tag{4.32}$$

By solving Eq. 4.30 and 4.31 with Eq. 4.32, the collision time bounds for i-th obstacle are obtained as:

$$\begin{aligned} \tau^k_{i,p,q,1} &= min\left\{\tilde{\tau}^k_{i,p,q,1}, \tilde{\tau}^k_{i,p,q,2}\right\} \\ \tau^k_{i,p,q,2} &= max\left\{\tilde{\tau}^k_{i,p,q,3}, \tilde{\tau}^k_{i,p,q,4}\right\} \,, \end{aligned} \tag{4.33}$$

if any of the time bounds is within the interval $\tau \in [T^k_{s,p}, T^k_{s,p} + T^k_{d,p,q}]$. Additionally, if entry into collision with i-th obstacle was detected in an adjacent previous leaf or before and no change is detected on the current leaf, the time bounds are set to $\tau^k_{i,p,q,1} = T^k_{s,p}$ and $\tau^k_{i,p,q,2} = T^k_{s,p} + T^k_{d,p,q}$.

As mentioned earlier, the vehicle angle $\theta^k_{p,q}$ can be considered constant for small leaf lengths (profiles). Since the leaf orientation is known in the search phase, the collision interval bounds for z_4 of Eq. 4.15 where any possible orientation was assumed, can be less strict. If the i-th obstacle can potentially collide with the

vehicle on the interval, it's trace in the $z_4 - z_1$ plane is a parallelogram:

$$\begin{aligned}
z^k_{1,i,p,q,1} &= z_1(\tau^k_{i,p,q,1})\,, \qquad z^k_{1,i,p,q,2} = z_1(\tau^k_{i,p,q,2}) \\
z^k_{4,i,p,q,1} &= y^k_i + v^k_{i,y}\tau^k_{i,p,q,1} - \tfrac{l}{2}\sin(\theta^k_{p,q}) - r_i - R\,, \\
z^k_{4,i,p,q,2} &= y^k_i + v^k_{i,y}\tau^k_{i,p,q,1} + \tfrac{l}{2}\sin(\theta^k_{p,q}) + r_i + R\,, \\
z^k_{4,i,p,q,3} &= y^k_i + v^k_{i,y}\tau^k_{i,p,q,2} - \tfrac{l}{2}\sin(\theta^k_{p,q}) - r_i - R\,, \\
z^k_{4,i,p,q,4} &= y^k_i + v^k_{i,y}\tau^k_{i,p,q,2} + \tfrac{l}{2}\sin(\theta^k_{p,q}) + r_i + R\,,
\end{aligned} \tag{4.34}$$

where $z^k_{1,i,p,q,1}$ and $z^k_{1,i,p,q,2}$ are calculated depending on the velocity profile of Eq. 4.30 or 4.31.

4.3.6.3 Unidirectional RRT expansion

In Sec. 4.3.6.1 it was stated that four different parameters of each tree leaf $\mathcal{T}^k_{p,q}$ are randomized, namely, velocity profile's acceleration $\ddot{z}^k_{1,p,q}$, acceleration duration $T^k_{a,p,q}$, total leaf sample time $T^k_{d,p,q}$ and the orientation angle $\theta^k_{p,q}$. The first three parameters are related to the longitudinal control search, whereas the orientation angle is related to the steering control search. At each replanning sample instant k a tree $\mathcal{T}^k$ is grown from the current vehicle position $z_{init} = (z^k_1(0), z^k_4(0))$ to a goal region $\mathcal{U}(z^k_{goal}, z^k_{1,goal_{max}}, z^k_{4,goal_{max}}) = \left[z^k_1(\tau^k_f) \pm z_{1,goal_{max}}, z^k_4(\tau^k_f) \pm z_{4,goal_{max}}\right]$. The subgoal position z^k_{goal} is defined by higher level meta-planner based on global scenario waypoints. $z^k_{1,goal_{max}}$ and $z^k_{4,goal_{max}}$ are subgoal region bounds dependent on current vehicle velocity. The subgoal region is not bound to a single point, because it might be too restrictive in some cases, e.g. when a vehicle in front is just positioned "on the subgoal", therefore more maneuvering flexibility is given to the vehicle on intermediate goals, whereas the final goal $z_{goal}(T_f)$ is a point in state space. The time to reach subgoal τ^k_f and the final time T_f for the whole run are free parameters. Note that replanning must be started either at sample rate T_s from the k-th interval or even before, if $\tau^k_f < T_s$.

The trees are expanded through the EXTEND function in a random z_{rand} direc-

RRT_SEARCH(z_{init}, $\mathcal{U}(z_{goal}, z_{1,goal_{max}}, z_{4,goal_{max}})$)
1 $\mathcal{T}$.init(z_{init});
2 **for** $n = 1$ **to** N **do**
3 $z_{new} \leftarrow$ EXTEND($\mathcal{T}$, z_{rand});
4 **if not** ($z_{new} = \varnothing$) **then**
5 **if**($z_{new} \in \mathcal{U}$) **then**
6 $\mathcal{L} \leftarrow \mathcal{L} \cup (z_{new}, cost)$;
7 **end**
8 **if not** $\mathcal{L} = \varnothing$ **then**
9 **return** MIN_PATH($\mathcal{T}$);
10 **else return** *Failure*;

EXTEND($\mathcal{T}$, z)
1 $z_{near} \leftarrow$ NEAREST_NEIGHBOR(z,$\mathcal{T}$);
2 $z_{new} \leftarrow$ NEW_STATE(z, z_{near});
3 **if not** ($z_{new} = \varnothing$) **then**
4 $\mathcal{T} \leftarrow \mathcal{T} \cup z_{new}$;
5 **return** z_{new};
6 **else return** $\varnothing$;

Table 4.2: Unidirectional RRT search in the $z_4 - z_1$ space.

tion from the nearest z_{near} point on the tree obtained by NEAREST_NEIGHBOR function. An increment from z_{near} along the z_1 axis is based on the randomized velocity profile (Sec. 4.3.6.1). The possible new state is tested for collision against all obstacles in the NEW_STATE.

A list $\mathcal{L}$ of final nodes that reached the goal region is kept separately. The node with minimum cost and its backpointers form the final RRT path, which is a polyline. The cost of a single node $\mathcal{T}_p$ on the list $\mathcal{L}$ is pondered as:

$$\Gamma(\mathcal{T}_p) = \omega_{time} \cdot \Gamma_{time}(\mathcal{T}_p) + \omega_{\theta_{diff}} \cdot \Gamma_{\theta_{diff}}(\mathcal{T}_p) \tag{4.35}$$

where Γ_{time} is the total time cost from the k-th resample time and $\Gamma_{\theta_{diff}}$ the total absolute orientation change of the vehicle along the path to $\mathcal{T}_p$ that should

be minimized.

As in the case of Sec. 4.3.5 the RRT polyline represents the control points $z_4 = \mathcal{B}(z_1)$ of the spline polygon with the z_1 coordinate sampled at Greville abscissae (Eq. 4.18), from which the final B-spline robot trajectory can be determined.

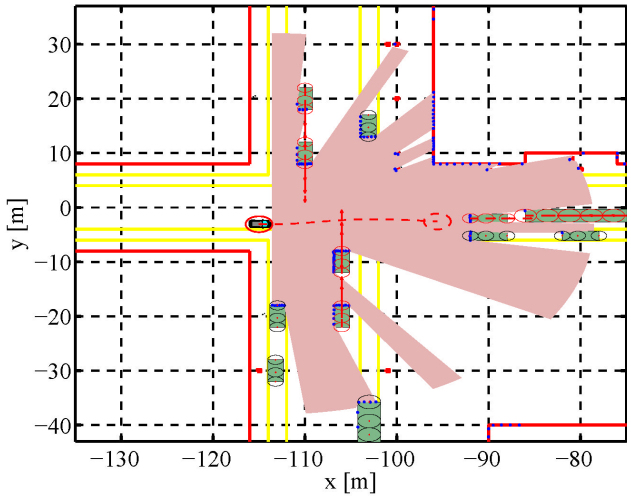
4.3.6.4 Simulation results in an dynamic urban scenario environment

The Smart Car simulator used in the results of this section was developed in MatLab. It reproduces a simple 2D urban-like environment (approximately 800 m x 800 m) with parked and moving cars, buses, trucks, and people, buildings, walls, streets, and trees. Using the simulator one may reproduce the 2D kinematic behavior of a car-like vehicle. In simulator all environment static and dynamic features are represented by lines and/or arcs. The sensor data are extracted from the environment based on its geometrical description and used as input data for the algorithms. The simulator uses the global position of the Smart Car in the environment for selecting a feature-window that contains all lines and/or arcs close to the vehicle according to the exteroceptive sensor measurement. The Dijkstra algorithm [Dijkstra, 1959] is used for calculating the shortest global path by taking into account the right-hand traffic convention. A sequence of intermediate positions from the vehicle's initial position to the goal is thus generated.

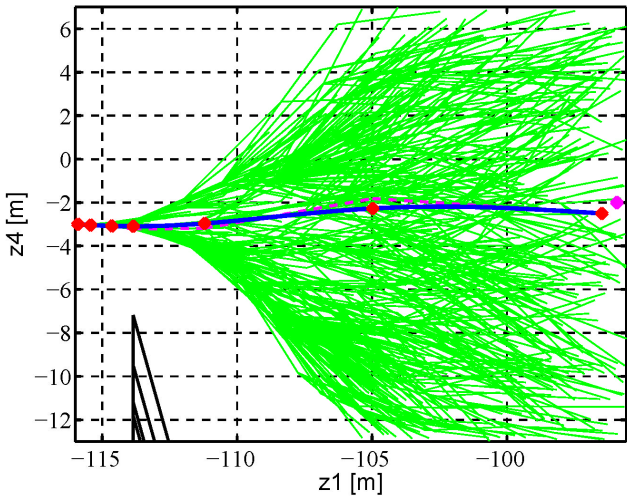
Sequences in Fig. 4.4, 4.5 and 4.6 show different scenarios with intersection crossing of the vehicle (waiting for other vehicles to pass by), speeding up in free space area and slowing down while approaching slowly moving vehicles, respectively. The RRT search is shown in Fig. 4.4(b), 4.5(b) and 4.6(b) with potential collision obstacle traces and the B-spline final trajectory which is obstacle free.The vehicle commands, i.e. the speed and the steering angle, are shown in Fig. 4.4(c), 4.5(c) and 4.6(c). Vehicle specifications are: max. speed $v_{max} = 10m/sec$, max. steering angle $\phi_{max} = 0.4rad$, max. acceleration $acc_{max} = 1.5m/sec^2$, the RRT leaf randomized time interval $T_{d,p,q} \in [1, 2]\, sec$ and global planning sample rate of $T_s = 2sec$. The RRT structure is built incrementally, so the obstacle traces drawn

are valid only for one path variant. In Fig. 4.4(b) it can be seen that the only obstacle trace within the search area (defined by the sensor range) is the one of the up most vehicle moving down, whereas the other moving vehicles have already passed by the time the ego-vehicle reaches the collision zone. In the particular case, there was no solution found to drive into the intersection before the other vehicles due to the acceleration limits. In Fig. 4.6(b) there is no trace of the obstacle in the front shown, since it is already gone from the search area by the time the ego-vehicle reaches the subgoal position. However, the ego-vehicle is still slowed down as a result of search for a feasible trajectory.

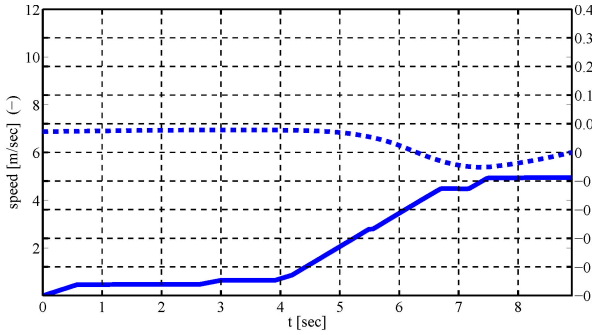
In summary, this section presents a motion planning algorithm for a nonholonomic car-like platform in dynamic environments with linearly moving obstacles. The potential collision check is performed in transformed space where time component is resolved in a criterion based on relative vehicle to obstacle motion. In consequence, generating a feasible path is a geometric search which is performed here using unidirectional RRT expansion. Final smooth trajectory is obtained using B-spline interpolation. The approach enables full control of a vehicle both in terms of longitudinal and rotational velocity (i.e. speed and steering). Simulation results are provided for a dynamic urban traffic scenario.



(a) Urban environment.

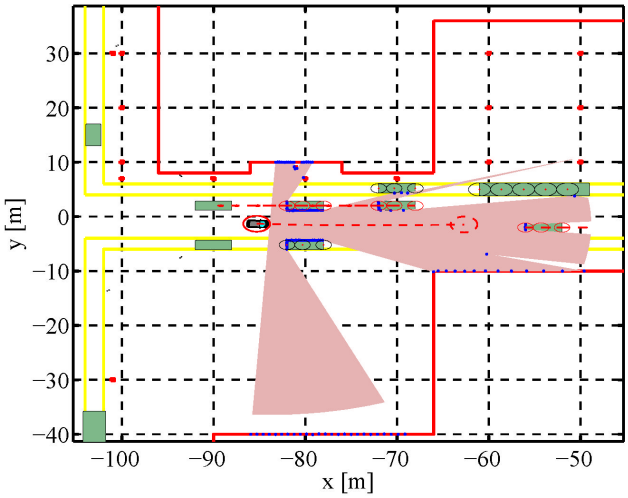


(b) Vehicle trajectory search and obstacle traces.

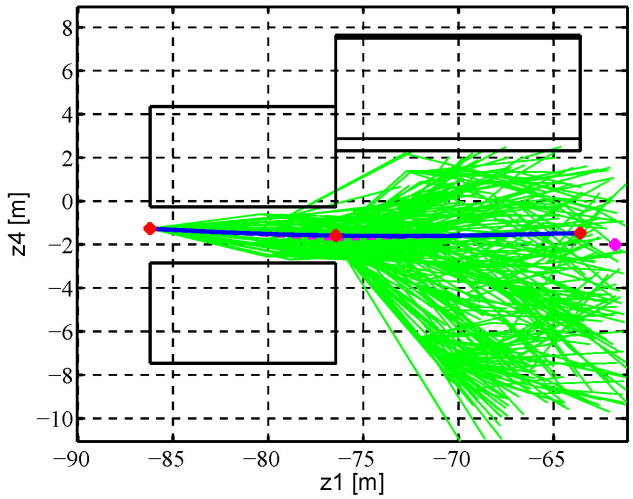


(c) Vehicle speed and steering angle.

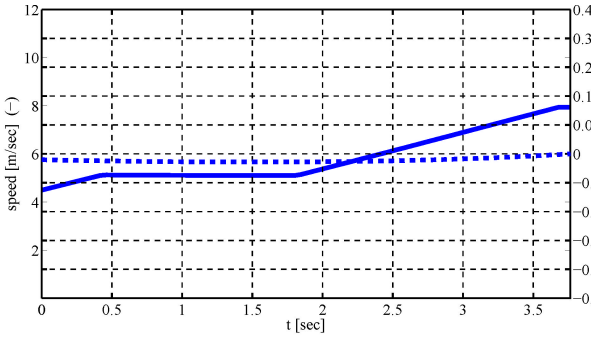
Figure 4.4: Ego-vehicle intersection crossing with automatic waiting for vehicles to pass.



(a) Urban environment.

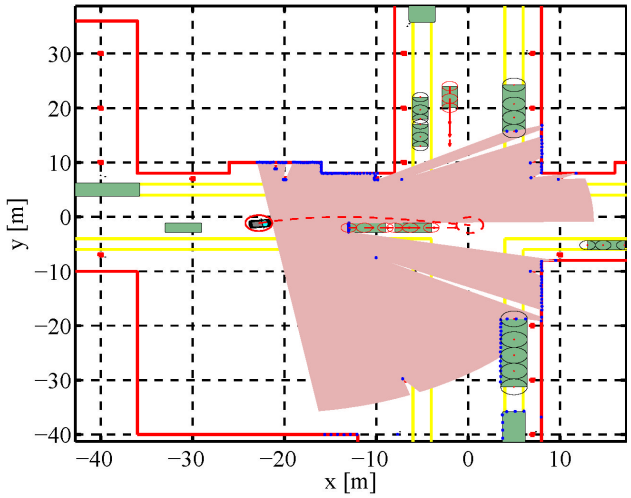


(b) Vehicle trajectory search and obstacle traces.

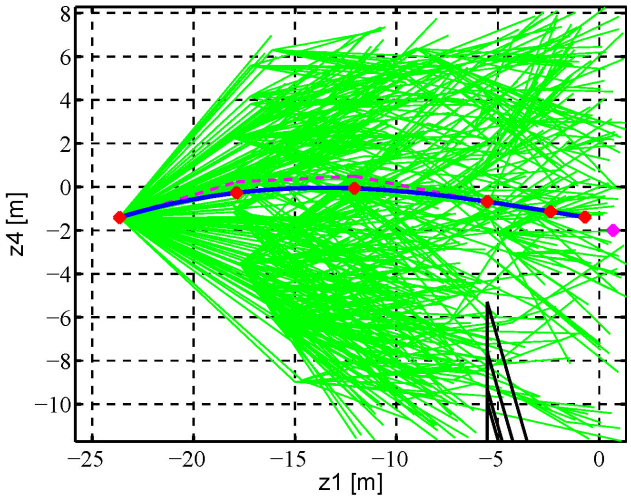


(c) Vehicle speed and steering angle.

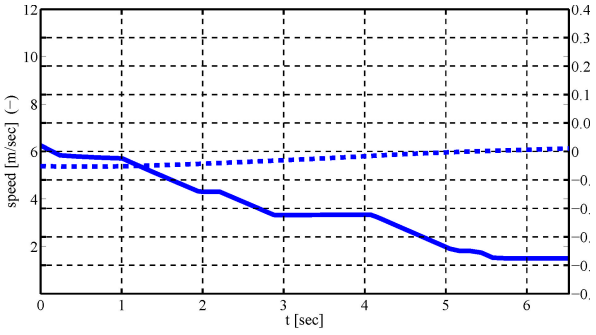
Figure 4.5: Ego-vehicle accelerating to nominal speed in free space area.



(a) Urban environment.



(b) Vehicle trajectory search and obstacle traces.



(c) Vehicle speed and steering angle.

Figure 4.6: Ego-vehicle decelerating after approaching slowly moving vehicles.

4.4 Motion planning in trajectory space

4.4.1 Introduction

The motion planning analyzed in this section is part of a larger navigation scheme oriented towards autonomous vehicle driving in dynamic urban environments with a particular focus on the motion safety issue.

The deliberative part of the architecture features two key modules working together in a hierarchical fashion: the *Route Planner* (global waypoints level) and the *Partial Motion Planner* (trajectory level).

The purpose of the Route Planner is to provide the Partial Motion Planner with a valid route towards a given goal. A route can be vary from a set of global waypoints (checkpoints) to visit or can be comprised of a geometric path augmented with additional information based on the structure of the environment considered. Such a route should comply with the standard regulations for vehicles driving in a urban setting. This means that factors such as speed limits and stop signs should be taken into account. The Route Planner will be presented in more detail in Sec. 6.2 and 6.3.

Suffice to say at this point that the output of the Route Planner module consists of a list of configurations, i.e. waypoints $\mathcal{Q}_g = \{q_g^1, \cdots, q_g^{N_g}\}$ that the vehicle should reach. These configurations are passed on to the Partial Motion Planner which takes care of all the details of the actual driving. It relies upon the route, obstacle information, i.e. the local vehicle's environment (with up to date information about the fixed and the moving objects) in order to determine the next motion command to apply to the vehicle.

As the name suggests, a Partial Motion Planning scheme is used [Petti and Fraichard, 2005], [Macek *et al.*, 2006], [Macek *et al.*, 2008b] in order to:

- take into account the *decision time constraint* imposed by dynamic environments;

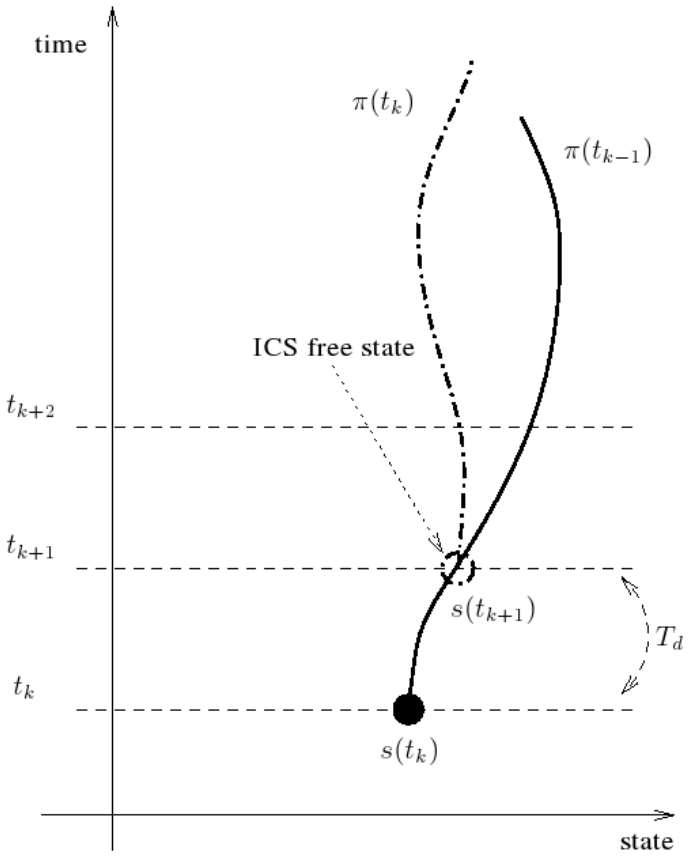


Figure 4.7: Partial motion planning iterative scheme.

- search for a feasible trajectory given the current vehicle's environment information;
- improve convergence towards the desired goal configuration.

4.4.2 Partial Motion Planning

As mentioned in the introduction (Sec. 4.4.1) the Partial Motion Planner (PMP) is the core navigational module of the deliberate navigation architecture for dynamic

urban scenarios.

Its primary purpose is to determine the motion command u that is sent to the vehicle controller at every time cycle. The motion command u must meet the following requirements:

- *Feasibility:* it must take into account the dynamic constraints of the vehicle;
- *Goal Convergence:* it must eventually drive the vehicle towards the desired goal;
- *Safety:* it must ensure the safety of the vehicle, *ie* its ability to avoid collisions.

According to the motion time constraints introducted in Sec. 4.2 PMP operates iteratively within the *decision time constraint* T_d which is determined primary by the environment dynamicity (in an environment featuring moving objects, you have a limited time only to decide upon your future course of action otherwise you run the risk of being hit by a moving object).

To determine u, PMP (as the name suggests) applies the Partial Motion Planning principle [Petti and Fraichard, 2005]: it tries to make the best possible use of the decision time T_d available by computing a partial motion towards the goal. To that end, a diffusion technique is used to explore the state×time space of the vehicle and determine a partial motion π that is used during the next time cycle to drive the vehicle towards its goal.

Fig. 4.7 illustrates how PMP operates. Let t_k denote the current time instant and the beginning of the k^{th} PMP cycle. The previous PMP cycle has computed the partial motion $\pi(t_{k-1})$ that starts at time t_k. The k^{th} PMP cycle then has to compute $\pi(t_k)$ that will start at time $t_{k+1} = t_k + T_d$. The process is repeated until the goal is reached.

At every cycle, PMP takes as input an updated model of the environment that comprises:

- the route computed by the Route Planner, *ie* the list $\mathcal{Q}_g = \{q_g^1, \cdots, q_g^{N_g}\}$ and the corresponding constraints $c^i, i = 1 \cdots N_g$ (*cf* Sec. 6.3),

- a list $\mathcal{O}_s$ of static objects: it is assumed that the static part of the environment is known from the road network structure and is described by a list of forbidden regions represented by closed polygons;

- a list $\mathcal{O}_d$ of dynamic objects: one fundamental task of the World Modelling module is to provide PMP with an updated model of the environment of the vehicle at every time cycle. PMP must know what are the moving objects present in the environment and, most important, what their future behavior will be. To that end, the World Modelling module features a Prediction module whose purpose is to estimate the future behavior of the moving objects. It is assumed that the prediction is valid over a given *prediction horizon* of duration T_p. The moving objects are described by a list of forbidden regions whose position varies over time. Their shape is modelled by rectangular bounding boxes which is suitable for vehicles and pedestrians alike (more general shapes could be used). The notation $\mathcal{O}_d(t)$ is used to indicated the fact that their position varies over time.

Each partial motion π computed respects the dynamic constraints of the vehicle considered thus meeting the Feasibility requirement.

By nature, PMP aims at maximizing the lookahead of the navigation process (the exploration of the future is carried out as far as possible given the decision time T_d available). In our opinion, this is one way to meet the Goal Convergence requirement.

Finally, each partial motion π computed will be safe in a predefined way (for instance, it will be guaranteed that the vehicle always have the possibility to brake down and stop before a collision occurs). This is the answer to the Safety requirement.

4.4.3 Diffusion Technique

The exploration phase of motion planning is based on an incremental search in the trajectory-space of the ego-vehicle. The resulting final trajectory from the starting to the goal configuration is a concatenation of trajectory chunks stemming from the ego-vehicle state branching and transitions in the exploration phase. These trajectory chunks can be understood as motion primitives of a single incremental search step. In principle, there are many possible transitions from a single state that are limited only by kino-dynamic characteristics of the vehicle and the configuration space obstacle motions. The transitions that were explored in a single incremental step can be branched further in the step, resulting in the exponential increase of possible states/transitions in an undirected search. Therefore, special attention must be payed on how the search and branching of possible states can be directed toward the goal objective.

Regarding the partial motion *ie*, the trajectory, that is to be computed for the vehicle it can be described as a single parametric curve, *eg* polynomial or spline curve, or a concatenation of several geometrical primitives such as arcs or clothoids. In the case of this presented trajectory diffusion here, the chunks of trajectory primitives are defined by the kinematic level velocity profiles. For instance, for a Ackermann vehicle the longitudinal and rotational velocity can be described by chunks of trapezoidal or triangular time profiles, where the rotational velocity profile is related to the steering wheel angle, since the kinematic vehicle model is described by the Ackermann model:

$$\dot{x} = \cos\theta\, v_l\,, \qquad \dot{y} = \sin\theta v_l\,, \qquad \dot{\theta} = \frac{v_l}{L}\tan\phi\,, \tag{4.36}$$

with $\{x, y, \theta\}$ being the robot pose and $\{v_l, \phi\}$ the longitudinal velocity and steering angle. Therefore the full vehicle state at the beginning of the current

replanning cycle t_k can be described as:

$$s(t_k) = \{x(t_k), y(t_k), \theta(t_k), v_l(t_k), \phi(t_k)\} \tag{4.37}$$

The dynamic update of the system can be described in the discrete general form:

$$\dot{s}(t_k) = f(s(t_k), u(t_{k-1})) \tag{4.38}$$

where the dynamic level control input vector u is the longitudinal acceleration $\dot{v}_l(t_{k-1})$ and the steering rate $\dot{\theta}(t_{k-1})$. The dynamic update function f encapsulates the dynamic model of the vehicle which can also include inertia and physical forces acting on the vehicle itself (emulated by ODE numeric integration).

If a low-level control is implemented separately (cascade control) which handles directly the actuators of the vehicle, *ie* gas pedal (longitudinal acceleration $\dot{v}_l$) and steering wheel torque (steering rate $\dot{\theta}$), the system function f represents the closed-loop response of the vehicle with low-level control. Therefore, the actual commands issued from the Partial Motion Planning level are the kinematic level control reference values:

$$u = \{v_{l,ref}, \phi_{ref}\} \tag{4.39}$$

whose time profile changes are limited by $\dot{v}_{l,max}$ and $\dot{\phi}_{max}$. In turn, this means that the control vector u needed for low-level control is derived directly from the planned trajectory since each vehicle configuration (state) on the trajectory represents not only the pose of the vehicle but also the kinematic control inputs, i.e. the steering angle and longitudinal velocity, according to Eq. 4.37.

In order to deliver a valid trajectory plan π_k at the end of the planning cycle k of duration T_d (based on the Partial Motion iterative scheme depicted in Fig. 4.7), the best feasible trajectory must represent vehicle motion at least of duration T_e, which is the execution cycle of the trajectory controller. Here it is assumed that $T_d = T_e$. Using a single kinematic level steering and velocity profile for the whole

duration of T_d would generate trajectories that lack enough complexity to find a feasible free space solution in a complex dynamic scene. Therefore, as already mentioned earlier, the trajectory consist of smaller trajectory primitive chunks each of duration T_h, the branching time. The whole exploration structure grows as a tree, where each node represents a vehicle state and the transitions between states are governed by the control inputs of Eq. 4.39.

The motion exploration phase starts with the current vehicle state $s(t_k)$ and the control trajectory from the previous cycle π_{k-1}, the goal configuration(s) $\mathcal{Q}_g$,the set of dynamic obstacles $\mathcal{O}_d$(t) and static obstacles $\mathcal{O}_s$. Firstly, the current state of the vehicle $s(t_k)$ is simulated to the beginning of the next execution cycle state $s(t_{k+1})$, according to the current available trajectory π_{k-1}. Then, the trajectory exploration phase starts as new states are inserted into the search tree $\mathcal{T}$. The available decision time for exploration is of duration T_d, after which a valid trajectory is given, if there exists a winning tree branch node that is at least a T_e time in the future with respect to the tree root node.

The whole diffusion process is depicted in Fig. 4.8. The node states being explored for the next execution cycle $k+1$ are a time $t_{k+1}^{(n)} = t_{k+1} + nT_h$ further in the future than the root node state $s(t_{k+1})$, where n is a particular node's tree depth and the T_h the branching time. The global index $k+1$ represents the execution cycle $k+1$, whereas the resultant trajectory π_k at the end of exploration is referenced to cycle k in which it was computed. The transition between states $s(t_{k+1}^{(n-1)})$ and $s(t_{k+1}^{(n)})$ is governed by control inputs $u(t_{k+1}^{(n-1)}, t_{k+1}^{(n)})$.

The pseudocode of trajectory diffusion is listed in Tab. 4.3. Firstly, the newly formed tree $\mathcal{T}$is initialized by its root node τ_{root}, where the information contained in each node is of the form:

$$\tau = \{s, u, w_\tau, t_\tau, n\} \tag{4.40}$$

The s represents the state of the vehicle, u the reference input that induced the state s, w_τ the overall cost to reach the node τ, t_τ the cumulative time with respect

to the root of the tree $\mathcal{T}$ and n the node depth within the tree. Therefore, τ_{root} contains the predicted state $s(t_{k+1})$ (PREDICT_STATE) for the given control trajectory π_{k-1} from the previous navigation cycle and zeroed cost, cumulative time and tree depth. The state prediction for the root node is necessary due to the fact that the currently computed trajectory $\pi(k)$ will be available only at the end of the current navigation cycle. The $\mathcal{L}_1$ contains all the nodes that are appended to the current tree depth $d_{\mathcal{T}}$. The overall computation time t_s can only be within the T_d span (Tab. 4.3. L5).

There are three node expansion methods implemented in the randomized scheme, inspired by the tree growing techniques of a RRT ("Rapidly-Exploring Random Tree") [LaValle and Kuffner Jr., 2001]:

1. *randomized exploration*: expansion towards a random configuration in the environment - q_{rand} (Tab. 4.3. L9-L13);

2. *greedy goal search*: expansion towards a goal configuration - q_{goal} (Tab. 4.3. L14-L18);

3. *randomized control input*: extension from a given tree node (state) using a randomized control input - u_{rand} (Tab. 4.3. L19-L23).

These cases are depicted further in Fig. 4.8. In case 1, the predicted state $s(t_{k+1})$ is expanded towards a random configuration q_{rand} in the workspace ($s^1(t_{k+1}^{(1)})$), whereas in case 2 the tree $\mathcal{T}$ is expanded towards the goal configuration q_{goal} ($s^2(t_{k+1}^{(1)})$). In both cases the function EXTEND_STATE_SPACE first finds the node τ_{near} in the tree $\mathcal{T}$which is the closest to the chosen configuration according to a predefined distance metrics and than chooses the nearest feasible command with NEAREST_COMMAND function. The case 3 represents the direct search in the control space of currently kino-dynamically feasible commands with RAND_COM_SPACE ($s^3(t_{k+1}^{(1)})$).

If the set of available control inputs is appropriately sized also the *exhaustive trajectory space* search can be performed on the level of the whole planner

```
PMP_SEARCH(s(t_k), π(t_{k-1}), Q_g, O_d(t), O_s, T_d)
1  t_s=0.0, d_T=0;
2  s(t_{k+1}) ← PREDICT_STATE(s(t_k), π(t_{k-1}));
3  τ_root.init(s(t_{k+1}), u(t_{k-1}), 0.0, 0.0, d_T);
4  T.init(τ_root), L_1.init(τ_root), L_2=∅, τ_⋆=∅;
5  while t_s ≤ T_d do
6     for n=1 to |L_1|
7     τ_ext ← L_1.pop();
8     p ← RAND();
9        if (p < T.p_r)
10          q_g^rand ← RAND_STATE_SPACE(O_s);
11          τ_new ← EXTEND_STATE_SPACE(q_g^rand,...);
12             if not (τ_new = ∅) then
13             INSERT_WITH_COST(τ_new, Q_g, T, L_2, d_T, τ_⋆);
14       if (T.p_r ≤ p < T.p_r + T.p_g)
15          q_g^rand ← RAND_SELECT(Q_g);
16          τ_new ← EXTEND_STATE_SPACE(q_g^rand,...);
17             if not (τ_new = ∅) then
18             INSERT_WITH_COST(τ_new, Q_g, T, L_2, d_T, τ_⋆);
19       else
20          u_rand ← RAND_COM_SPACE(u(t_{k-1}));
21          τ_new ← EXPAND_COM_SPACE(τ_ext, u_rand,...);
22             if not (τ_new = ∅) then
23             INSERT_WITH_COST(τ_new, Q_g, T, L_2, d_T, τ_⋆);
24    end
25 d_T=d_T+1;
26 swap(L_1, L_2);
27 UpdateTime(t_s);
28 end
29 if not τ_⋆ = ∅ then
30    return PATH(T, τ_⋆);
31 else return failure;
```

Table 4.3: Diffusion process of the Partial Motion Planner I.

tree. Such a case would for instance be that from each state node the current steering angle and steering left or right based on the steering rate parameter is performed for the duration of the arc primitive. This is a special case of the randomized control input in Tab. 4.3. L19-L23, where the whole available set of controls is explored, not only a randomly selected subsample. However, whereas

EXTEND_COM_SPACE(τ, u, $\mathcal{O}_d$, $\mathcal{O}_s$)
33 $\tau_{new} \leftarrow$ EXTEND_WITH_SAFETY_CHECK(τ, u,$\mathcal{O}_d$, $\mathcal{O}_s$);
34 **if not** (τ_{new}==$\varnothing$) **then**
35 $\tau_{new}.n$=$\tau.n$+1;
36 $\mathcal{T} \leftarrow \mathcal{T} \cup \tau_{new}$;
37 **return** τ_{new};
38 **else return** $\varnothing$;

EXTEND_STATE_SPACE(q_g, $\mathcal{T}$, $\mathcal{O}_d$, $\mathcal{O}_s$)
39 $\tau_{near} \leftarrow$ NEAREST_NEIGHBOR(q_g, $\mathcal{T}$);
40 $u_{near} \leftarrow$ NEAREST_COMMAND(τ_{near}, q_g);
41 **return** EXTEND_COM_SPACE(τ_{near}, $\mathcal{O}_d$, $\mathcal{O}_s$);

INSERT_WITH_COST(τ, $\mathcal{Q}_g$, $\mathcal{T}$, $\mathcal{L}$, $d_{\mathcal{T}}$, $\tau_\star$)
42 COMPUTE_COST(τ, $\mathcal{Q}_g$, $\tau_\star$);
43 $\mathcal{T} \leftarrow \mathcal{T} \cup \tau$;
44 **if** ($\tau.n$==$d_{\mathcal{T}}$+1) **then**
45 $\mathcal{L}$.push(τ_{new});

Table 4.4: Diffusion process of the Partial Motion Planner II.

the randomized control set can be arbitrarily finely discretized/large in the sense of control samples, the exhaustive expansion case is bound to a small set of control commands in order to keep the computation tractable.

For all the possible expansion methods, the newly obtained nodes/states and the trajectory primitives that connect them to their predecessor nodes in the tree have to be checked for collision against all the dynamic and static obstacles. This is done by discretizing the generated connecting trajectory primitives according to the state equation Eq. 4.37 in time and verifying possible polygonal intersections that could occur between the ego-vehicle and obstacles intermediate configurations in the EXTEND_WITH_SAFETY_CHECK (Tab. 4.4. L33), *ie* no configuration along a state transition should induce a collision and the vehicle must be able to

come to a full-stop without collision following the expanded node (further details safety issues will be given in Sec. 4.5). In Fig. 4.8 this is depicted by the example of the break-state $s^3(t_{k+1}^{(1)}+T_b)$ originating from state $s^3(t_{k+1}^{(1)})$, where the collision-check test is performed for the duration of the breaking maneuver. The brake time T_b is related to the dynamic capabilities of the vehicle, *ie* max linear deceleration $\dot{v}_{l,decc}$:

$$T_b = \frac{v_l}{\dot{v}_{l,decc}} \tag{4.41}$$

where the longitudinal velocity depends on each given state, in this case $s^3(t_{k+1}^{(1)} + T_b)$.

As already mentioned, the diffusion process computation is limited to $t_s \leq T_d$, where at each new computational increment another layer of nodes is added to the tree $\mathcal{T}$. However, the final depth of $\mathcal{T}$ is upper bounded only with the motion planning horizon T_m beyond the motion plan is considered too uncertain to be further explored. (*cf* Sec. 4.2).

Regarding the goal search, *ie* finding a trajectory towards a goal configuration, there are three approaches distinguished here:

1. *waypoint following*: the current set of waypoints $\mathcal{Q}_g$is the next topological node to reach in the environment (see Sec.6.3), such as an intersection, route crossing, lane changing waypoint, etc.;

2. *route following*: the current set of waypoints $\mathcal{Q}_g$is a collection of intermediate configurations, which together describe a route between topological nodes.

3. *global navigation function*: the cost to the next global waypoint is an interpolated navigation function that can be used as a goal cost metric.

In case 1 the cost function of a node to determine the best trajectory is based on a distance metric $\|\tilde{s} - \{q_g, c_q\}\|$ between the goal waypoint with its constraints and the last state $\tilde{s}$ on a trajectory:

$$w_\tau(\tilde{s}) = \alpha_g \cdot \|\tilde{s} - \{q_g, c_q\}\| + \alpha_t \cdot t_\tau(\tilde{s}) \tag{4.42}$$

α_g and α_t are the weighting factors between minimizing the distance to the waypoint and minimizing the cumulative time $t_\tau(\tilde{s})$ to reach it, respectively. The distance metric can be a simple Euclidean distance between the goal waypoint and the node. However, to obtain a metric that takes into account also the non-holonomic constraints of the vehicle, the Continuous Curvature Path length which is comprised of line segments, circular arcs, and clothoids is used as the distance metric [Fraichard and Asama, 2004a]. The advantage of using such metric over the simple Euclidean distance is that the such paths are generated with continuous, upper-bounded curvature and upper-bounded curvature derivative which can be set directly as parameters relating to the maximum steering angle and steering rate, respectively, of the ego-vehicle. Therefore, given a node and waypoint configuration it will appropriately estimate the non-holonomic distance.

In case 2 there is more information available based on the environment structure, in the form of a route. These geometrical configurations can be followed with a type of path following technique, with the possibility of deviating from the route based on the tree structure, if the dynamic obstacles trajectories require such evasive maneuvers. However, in the absence of dynamic obstacles $\mathcal{O}_d$ and proper route definition according to the a-priori knowledge of static obstacles $\mathcal{O}_s$, the vehicle should follow the route as close as possible. Assuming that a path following controller (implementation details on the controller can be found in [Solea and Nunes, 2006], [Macek *et al.*, 2008a]) computes an error function $\mathcal{E}$ which describes the discrepancy between a particular vehicle state s and the route $\mathcal{Q}_g$, then the cost of a node can be computed in the error terms as:

$$w_\tau(\tilde{s}) = \sum_{j=1}^{N_{\tilde{s}}} \|\mathcal{E}(s_j, \mathcal{Q}_g)\| \tag{4.43}$$

where the set of discretized states $\{s_{j=1} = s(k+1) \dots s_j \dots s_{N_{\tilde{s}}} = \tilde{s}\}$ with the associated arcs a forms a trajectory $\tilde{\pi}$. The cost calculation step is performed in the INSERT_WITH_COST function (Tab. 4.4. L42-L45). The best trajectory

$\pi\star = \pi(k)$ which will be applied in the next navigation cycle can be therefore determined from node $\tau_\star$ with minimum overall cost w_τ, iff $\tau_\star \neq \varnothing$.

The case 3 provides the connectivity to the goal configuration/waypoint in form of a navigation function that can describe different costs to goal and are typically computed in off-line fashion. Two interesting cases include performing dynamic programming in 2-D grid on the a-priori static environment, ignoring the non-holonomic constraints of the vehicle [Montemerlo *et al.*, 2008b]. The resultant navigation function, resembling a potential field represents the cost-to-goal as Euclidean distance metrics that also takes into account the local obstacle configuration. An off-line technique described in [Maxim Likhachev, 2008] computes the time optimal cost to a goal configuration for each cell of a grid of discrete vehicle configurations. This so-called lattice-based planning is particularly interesting since it already includes the non-holonomic vehicle constraints in the cost function, as well as the dynamics of the longitudinal and rotational velocity profile and the static obstacle configuration of the environment.

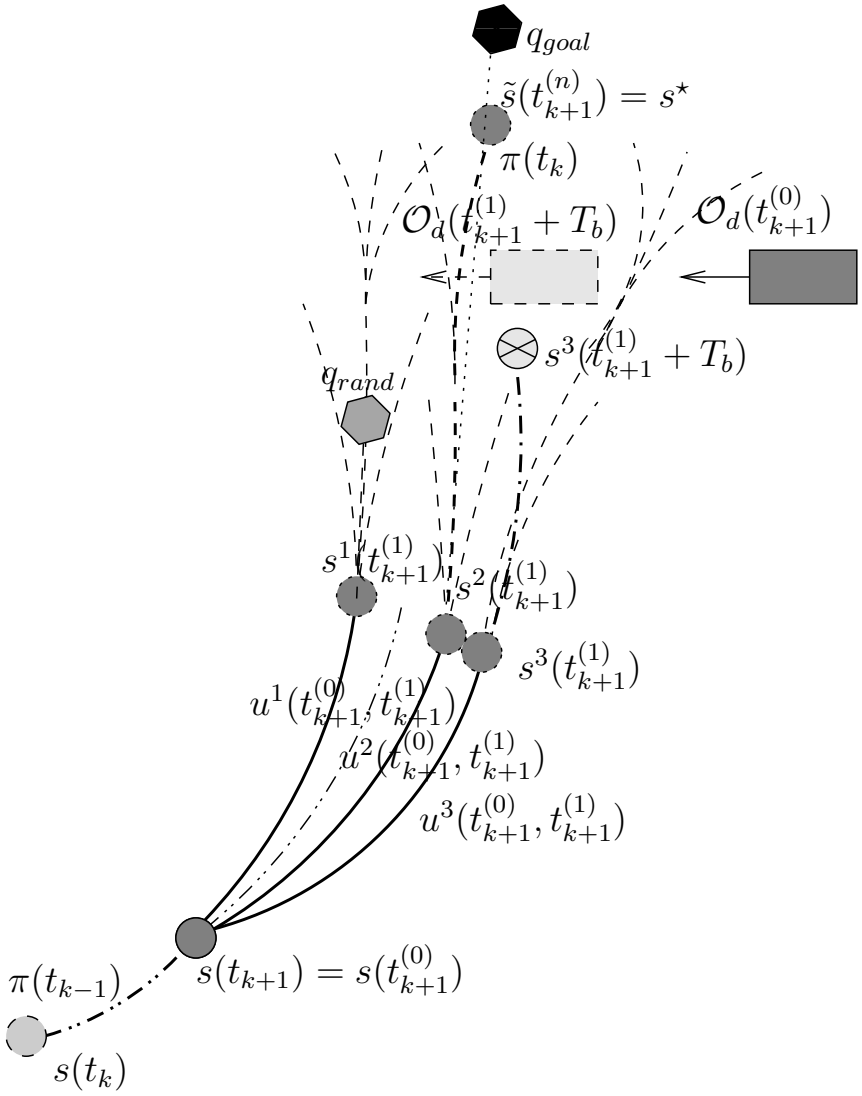


Figure 4.8: Trajectory diffusion with individual node expansion among dynamic obstacles. At the beginning of the replanning cycle at time t_k, the vehicle state is simulated according to the plan π_{k-1} available from previous cycle $k-1$, from state s_k to the initial state s_{k+1}. Only at the end of cycle k is the new plan π_k available for execution. The new trajectory π_k consists of trajectory primitives concatenation up to the winning node $\tilde{s}_{k+1}^{(n)}$, where n denotes the tree depth expansion and $t_{k+1}^{(n)} = t_{k+1} + nT_h$, where T_h is the expansion duration of a single trajectory primitive. $t_{k+1}^{(n)} \geq T_e$, where T_e is the execution time of the low-level trajectory controller.

4.5 Safety Issues

The purpose of this section is to explore the safety issues related to the proposed navigation scheme with trajectory space search.

The diffusion technique presented in Sec. 4.4.3 aims at building a tree embedded in the state×time space of the system $\mathcal{R}$ and to extract from this tree a partial motion π that is used during the next time cycle to drive the system towards its goal.

The concept of Inevitable Collision State (ICS)[1] [Fraichard and Asama, 2004b] and the motion safety criteria introduced in [Fraichard, 2007] show that it does not suffice that each partial motion π be collision-free to ensure the safety of $\mathcal{R}$.

From a theoretical point of view, the safety of $\mathcal{R}$ is guaranteed if and only each π is ICS-free up to the time T_d (that corresponds to the initial state of the partial motion that is to be computed at the next navigation cycle), because then, at the next navigation cycle, the navigation module *always* has a safe evasive maneuver available.

Now, checking whether a given state of π is an ICS or not requires in theory the *full* knowledge of the environment of $\mathcal{R}$ and its future evolution, *ie* the knowledge of the space-time $\mathcal{W} \times [0, \infty)$. In practice however, one has to deal with the sensors' limited field of views and the elusive nature of the future. Knowledge about the environment of $\mathcal{R}$ is thus limited both *spatially* and *temporally*: it is limited to $\mathcal{W}_p \times [0, T_p]$ where $\mathcal{W}_p \subset \mathcal{W}$ denotes the subset of the environment which is perceived and T_p the prediction horizon.

To further ensure safety with respect to the objects that lie outside of $\mathcal{W}_p$, its boundary is treated in a manner similar to [Fraichard and Asama, 2004b] or [Alami *et al.*, 02] as a potentially moving object whose motion direction is unknown but whose velocity is upper-bounded.

$\mathcal{W}_p \times [0, T_p]$ and the objects within, fixed and moving, yields in the state×time space of $\mathcal{R}$ a set of ICS which is only an approximation of the true set of ICS

[1] A state is an ICS iff a collision eventually occurs no matter how $\mathcal{R}$ moves.

generated by $\mathcal{W} \times [0, T_p]$.

This is the very reason why it is impossible to guarantee an absolute level of safety (absolute in the sense that it can be guaranteed that $\mathcal{R}$ will never end up in an ICS and therefore crash eventually).

This intrinsic impossibility compels us to settle for weaker levels of safety. Although weaker, the important thing is that such levels of safety will be guaranteed given the information that $\mathcal{R}$ knows about its environment, *ie* given $\mathcal{W}_p \times [0, T_p]$. We have explored two different levels of safety, they are detailed in the next two sections.

4.5.1 Safety Level #1

The first safety level we have looked to enforce is the one which guarantees that if a collision should ever occur, $\mathcal{R}$ will be at rest. In other words, if a collision is inevitable, it can be guaranteed that $\mathcal{R}$ always have the possibility to brake down and stop before the collision occurs. Such a safety level is a form of *passive* safety in the sense that $\mathcal{R}$ will never actively collide with an object. It is henceforth called *Passive Safety* and denoted by PS.

Under PS, a state s is considered as being safe iff there exists at least one braking maneuver starting at s which is collision-free until the time where $\mathcal{R}$ has stopped. PS yields the following definition for a safe state:

Def. 1 (**Passive Safety**). *a state s is safe under PS (or **p-safe**) iff there exists at least one braking maneuver starting at s and collision-free until T_b, with T_b the time where $\mathcal{R}$ is at rest (the braking time).*

In practice, the function EXTEND.WITH.SAFETY.CHECK (*cf* Tab. 4.3) samples a finite and discrete set of braking maneuvers and checks them for collision against $\mathcal{W}_p \times [0, T_p]$. If one collision-free maneuver exists the state considered is labeled as p-safe and unsafe otherwise.

4.5.2 Safety Level #2

In a way, PS leaves a part of the collision-avoidance burden to other objects. In certain situations however, this may be unsatisfactory: $\mathcal{R}$ may for instance decide to move on a railway track to reach its goal because, under PS, it is safe to do so (indeed $\mathcal{R}$ would have the time to stop before being hit by the train). Unfortunately, the train in spite of its best efforts may not be able to avoid crashing into $\mathcal{R}$ because of its own dynamics.

In an environment where the moving objects are assumed to be *friendly*, *ie* seeking to avoid collisions, and for which a certain knowledge about their dynamic properties is available, it can be desirable to enforce a stronger level of safety. This second safety level guarantees that, should a collision ever occur, $\mathcal{R}$ will be at rest and the colliding object would have had the time to slow down and stop before the collision had it wanted to. This safety level is henceforth called *Passive Friendly Safety* and is denoted PFS. It yields the following definition for a safe state:

Def. 2 (**Passive Friendly Safety**). *a state s is safe under PFS (or **pf-safe**) iff there exists at least one braking maneuver starting at s and collision-free until $T_b + T_{ob}$, with T_b the braking time of $\mathcal{R}$, and T_{ob} the maximum braking time of the moving objects present in the environment.*

The conservative nature of Def. 2 should be noted. It is possible in practice to refine it in order for example to take into account the dynamics of the particular moving objects that would collide with $\mathcal{R}$ when it follows a particular braking maneuver. For the time being, Def. 2 is left as is.

Other safety levels could be proposed. The ultimate one of course is to determine safety with respect to the set of ICS which is defined by $\mathcal{W}_p \times [0, T_p]$. Given the complexity of characterizing this ICS set, Passive Safety and Passive Friendly Safety constitutes interesting alternatives in the sense that they can be computed efficiently and provide an adequate level of safety.

4.5.3 Simulation results

In order to analyze the collision checking of the motion planning scheme based on the described trajectory diffusion of Sec. 4.4.3 and the safety level defined as *Passive Safety* of Sec. 4.5, the Fig. 4.9 represents a snapshot of the vehicle environment whereas the Fig. 4.10 represents the the trajectory diffusion process of the Partial Motion Planner (PMP) with included safety checking with braking maneuvers.

According to the Fig. 4.8 the PMP trajectory diffusion starts at the root of the tree structure which is then grown according to a particular expansion technique. In the case of Fig. 4.10, the expansion method applied is the exhaustive search, i.e. each tree node is expanded according to the full set of control inputs. The steering control input includes in this case $\{steer\ left, keep\ current\ steering, steer\ right\}$, with respect to the maximum steering rate and the duration expansion of a single trajectory arc (primitive). The longitudinal velocity of the vehicle is simulated as a maximum setpoint. The vehicle starts accelerating from the tree root until it achieves the maximum setpoint velocity, which can be seen from different trajectory arc lengths connecting the expansion nodes. Note, that this is just a depiction of a particular case of vehicle starting from a stop, whereas at the next replanning cycle the tree root state would already describe at a certain non-zero longitudinal velocity.

Different levels of the tree expansion (depths) are clearly visible as the delimiting space between each arc is an expansion node. The tree expansion is governed by the goal target as well as the static environmental configuration and the dynamic object traces, expanding only in dynamically available free space. The current best overall trajectory in accordance with the metric defined with respect to goal objective (in this case placed in the lane leading to the right of to the vehicle) is a concatenation of all the collision free trajectory primitives from the tree root (depicted magenta). In this case, the vehicle has already traversed a part of the best trajectory within the current cycle k of duration T_d. In parallel to the

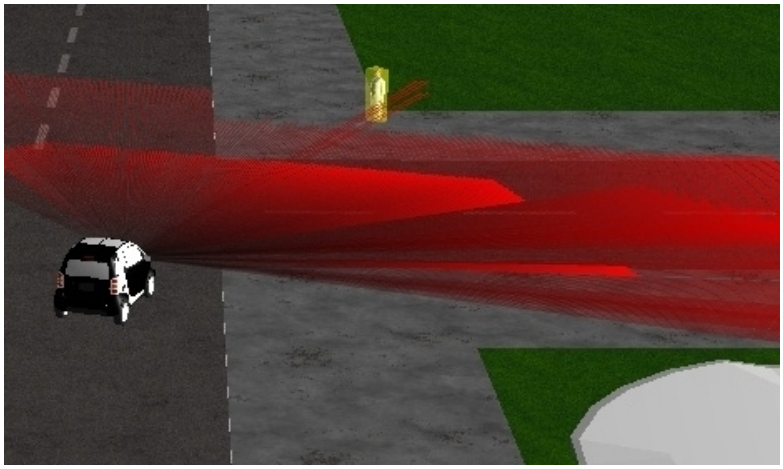


Figure 4.9: Trajectory diffusion and safety checking with braking maneuvers (scene view).

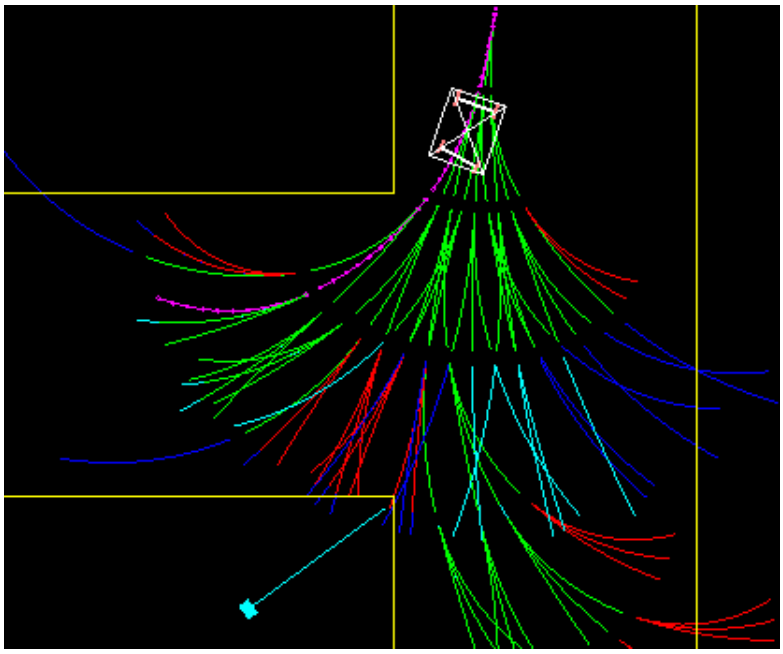


Figure 4.10: Trajectory diffusion and safety checking with braking maneuvers (navigation view).

vehicle execution controller, the motion planner is already exploring the solution for the next cycle t_{k+1}, which will be available at the end of the cycle k as π_k. In fact, according to time index notation in Sec. 4.4.3, the depicted tree in Fig. 4.10 is from the previous exploration cycle $k-1$ and the best trajectory being executed is π_{k-1}.

Each trajectory primitive is checked against collision at a discretized set of trajectory points describing a particular vehicle configuration in time. Time is given in the absolute frame, since the motion of dynamic objects is also described for future time instances. The trajectory primitives that are completely void of collisions (whether static environment or dynamic objects) as marked green in this case. If a trajectory primitive expanding from a node leads to a colliding vehicle configuration it is prohibited (depicted red).

However, as described in Sec. 4.5 on safety issues the collision check only on the trajectory primitive itself is not enough. According to the definition of *Passive Safety*, a particular trajectory primitive from an expansion node is deemed passively safe only if a braking maneuver following from the end of the trajectory would enable the vehicle to come to a full stop. Therefore, for instance some trajectory expansions in Fig 4.10 in the center of the intersection are passively safe, since they are followed with non-colliding breaking maneuvers (depicted cyan), others however are prohibited, since they wouldn't allow for the vehicle to stop before hitting the lane borders (depicted blue). Note, that different braking maneuvers could be considered, in this case, they simulated maximum braking capability of the vehicle while maintaining the current steering angle.

At this stage of development, more involved safety levels such as *Passive Friendly Safety* of Sec. 4.5.2 were not explored, mainly due to the fact that simulating such events would imply that all the dynamic objects considered would have to react actively to changes in the environment, in a sense distributing the motion planning problem among all the dynamic objects involved. However, given a dynamic object motion prediction that would also include their potential dynamic limits, such as braking capabilities could straightforwardly be included in the trajectory diffusion of the ego-vehicle as. Instead of collision checking at the end of each trajectory primitive for T_b braking time of the vehicle, it would also have to include the braking maneuver of the dynamic object for the total conservative duration of $T_b + T_{ob}$ as mentioned earlier.

The proposed motion planning technique in trajectory space as described here was applied successfully to the problem of autonomous vehicle navigation in urban dynamic scenarios, the results of which will be presented in more detail in Chap. 6.

4.6 Conclusion

When considering the autonomous vehicle navigation in presence of dynamic obstacles, the synchronization between the obstacle motion prediction, motion planning and execution is of key importance. This chapter analyzed the key time constraints imposed on the navigation scheme.

Second important element is the exploration of possible feasible motions which was approached here in transformed state space in Sec. 4.3 and direct trajectory space of the vehicle and obstacle motion in Sec. 4.4. Both navigation schemes have in common that the best feasible trajectory that defines the future vehicle motion is build in an incremental fashion using various possible trajectory primitives that define the transitions between two vehicle states. The trajectory primitives must both comply to the kino-dynamic constraints of vehicle motion as well as be collision free. The underlying container structure present in both navigation schemes was a tree of trajectory primitives starting from the starting vehicle state node at each replanning cycle whereas the branching conditions (explorations) varied according to a particular implementation.

The key overall paradigm to the problem of motion planning applied here was the Partial Motion Planning approach which at each cycle receives the information about the future dynamic obstacle motions, extrapolates the state of the ego-system to the next replanning cycle according to the current trajectory solution and then explores a new feasible trajectory until the decision time of the planner is exhausted.

Due to the vehicle and dynamic obstacle inertia it is not sufficient to check for possible obstacle collisions only for a particular state of the ego-vehicle but rather

it has to be assured that the system will be able to avoid collisions also at all future times. The condition on avoiding Inevitable Collision States which imply all future time instance was relaxed in Sec. 4.5 to Passive Safety and Passive Friendly Safety, of which the Passive Safety is guaranteed in the current implementation of the Partial Motion Planner.

The results included generic dynamic obstacle scenario and an urban like scenario in Sec. 4.3, whereas the application results of motion planning developed in Sec. 4.4 will be analyzed in more detail in Chap. 6 withing a larger hierarchical navigation scheme.

Chapter 5

Dynamic scene analysis

5.1 Introduction

In order to develop a motion plan in a dynamic environment the ego-motion estimation, the structure of the static environment and potential moving obstacles must be identified. In Sec. 5.2 a localization module for full 3D vehicle pose estimation is presented. The environment specifically analyzed here is a road structure where the driving lane should be detected and described in a functional form with respect to the ego-vehicle, which is presented in Sec. 5.3 including experimental results on different road structures. Moreover, dynamic objects of the environment such as moving vehicle on the road need be detected and tracked in time in order to predict their future motion. Based on a occupancy map of the environment that is updated based on range sensor measurements in Sec. 5.3, temporal differences in grid's occupancy cells can be detected which are associated with dynamic objects. By segmenting locally adjacent cells with similar temporal changes, parts or contours of objects are obtained in Sec. 5.5. By appropriate association of detected objects to existing or newly created tracks, future motions of the dynamic obstacles can be obtained. Experimental results on a set of dynamic urban situations is presented in Sec. 5.6.

5.2 Localization

The localization scheme in this work is based on both global world frame GPS signal as well as inertial measurements of the ego-vehicle motion. Notably, following signals were used:

- GPS unit: HPPOS (High-Precision Position Differential GPS), GGA (Global Positioning System Fix Data) at 100 Hz
- Inertial Measurement Unit: vehicle attitude, roll ϕ, pitch θ, yaw ψ angles at 100 Hz
- Optical gyro unit: yaw rate, ω_z at 100 Hz
- Ego-vehicle odometry: vehicle base speed v_b at 100 Hz

The vehicle state estimation scheme is based on the Extended Information Filter (EIF) algorithm, which is the dual form of the Extended Kalman Filter (EKF). In brief, the EKF form implies that the probabilistic belief $Bel(x_t)$ about the state of the system is approximated with Gaussian pdfs describing the the first two statistical moments, namely the mean μ_t and covariance Σ_t of the state x_t. The EIF form represents the state estimate in form of the information vector [Thrun, 2000]:

$$\xi_t = \Sigma_t^{-1}\mu_t \tag{5.1}$$

whereas the covariance matrix is replaced by the information or precision matrix:

$$\Omega_t = \Sigma_t^{-1} \tag{5.2}$$

therefore, for instance, for a system with very high initial uncertainty, the information matrix can simply be set to $\Omega_t \approx 0$, which is numerically advantageous over trying to represent covariances of almost infinite value.

However, the most important advantage of the EIF form is that the innovation step of the Kalman filtering, i.e. the fusion of new measurements to the state estimation becomes additive. Therefore, if new information arrives in the system in form of measurements it is simply added to the information vector ξ (after appropriate system Jacobian transforms for full derivation). Since addition is commutative it means that the measurements from different sensors can be added in arbitrary order and arbitrary delays. In fact, the EIF canonical form represents probability in logarithmic form [Thrun, 2000] and bears close resemblance to the $\log Odds$ probabilistic representation that will be detailed in Sec. 5.4.

A complete EIF localization scheme that estimated global 3D vehicle pose $[x, y, z, \phi, \theta, \psi]^T$ was already explored and implemented on the Smart vehicle in [Lamon *et al.*, 2006b]. It relied primarily on the GPS signal (HPPOS and GGA signal) and inertial measurements attitude information, namely the roll, pitch and yaw $[\phi, \theta, \psi]^T$ to estimate its 3D pose assuming an unconstrained 3D body motion. Moreover, the unconstrained 3D body motion implied also that the 3D vehicle velocity vector was estimated from the real-time kinematic GPS data. In the presence of a stable GPS signal, this allows for globally consistent motion estimation (up to the attitude bias).

However, the testing experience in particularly in urban environments showed that the GPS signal can be denied for extensive periods of time due to weak signal strength or become unstable due to the multipath problem between buildings. In order to be able to estimate the 3D vehicle motion based only on inertial measurements an additional 3D vehicle motion model was introduced here based on [Kelly, 2004]. It describes the vehicle constrained motion on a tangential plane defined by the roll ϕ and the pitch θ attitude angles. According to this model, the full 3D kinematic motion equation derives as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v_b \cos\psi \cos\theta \\ v_b \sin\psi \cos\theta \\ -v_b \sin\theta \\ \omega_x + \tan\theta(\omega_y \sin\phi + \omega_z \cos\phi) \\ \omega_y \cos\phi - \omega_z \sin\phi \\ \frac{1}{\cos\theta}(\omega_y \sin\phi + \omega_z \cos\phi) \end{bmatrix}$$

where the z–y–x Euler angle sequence is used, assuming the strapped down gyroscopes of the IMU oriented along the three axes of the body frame reading the three components of the angular velocity $[\omega_x, \omega_y, \omega_z]$. In the practical implementation, the yaw angle estimation from the IMU unit proved insufficiently accurate, therefore a high precision optical gyro unit was introduced for accurate ω_z measurement and consequently ψ estimate. Note also, that the Ackermann nonholonomic constraint of the vehicle motion, namely $\dot{\psi} \approx \frac{\tan\phi_s}{L} v_b$, where ϕ_s is the steering wheel angle and L the axis base, does not appear in the Eq. 5.5.3, since the yaw angle rate $\dot{\psi}$ is measured via the inertial units in the state transition equation.

There is a pertaining problem of negotiating the state estimation belief from the GPS measurement, which is global in nature but can induce a lot of outlier values and the inertial measurement motion estimation on the other side, which is locally smooth but inevitably introduces a vehicle pose estimation drift on a larger estimation scale. If due to the multipath problem the GPS signal introduces an outlier measurement z_t at time t, it can in principle be filtered out using ellipsoidal gating [Blackman and Popoli, 1999]:

$$\tilde{z}_t^T S^{-1} \tilde{z}_t \leq G \tag{5.3}$$

where S is the residual covariance matrix of the EIF/EKF filtering, G is the gate value (a design parameter) and $\tilde{z}_t$ is the difference between the actual measurement

z_t and the expected measurement $\hat{z}_t$ based on the system state update equations and the current state estimate $\hat{x}_t$ before the innovation step. Essentially, if the actual measurement is close enough to the expected measurement value, it is accepted, otherwise rejected according to the G value. However, this technique only tackles the short cycle GPS outlier problem in the on-the-fly vehicle pose estimation but does not solve the problem, for instance of a precise GPS signal that is inaccurate, which can be the case for the urban environment with occlusions and multipath problem.

5.3 Lane detection

An important aspect of dynamic scene analysis in structured environments is to detect road boundaries and describe the lane configuration where the vehicles travel. In this work, the lane detection module is based on a vision system. The vision module presented here is based on several image filters that provide diverse information about the environment. A set of hypotheses about the state of the system is generated by a probabilistic particle filter. Assuming a predefined model of the road the particles are tested according to image filters to infer the best belief vehicle position. Emphasis was placed on extracting relevant information from the scene and efficient testing. In particular, a new testing module based on Canny edge filter and Hough transform increased the accuracy and robustness of estimation. Performance of the vision module was tested under various real-road conditions [Macek *et al.*, 2004].

A vision module detecting road lane boundaries to determine position of the ego vehicle forms the basis of a driver support system. Several approaches have been employed already completing long distance road tests [Lützeler and Dickmanns, 1998], [Jochem *et al.*, 1993], [Broggi *et al.*, 1999]. Researchers at the Universityät der Bunderswehr München [Lützeler and Dickmanns, 1998] modelled the road as a clothoid while using edge detection to search for lane markings

in the image. A multiple camera configuration was used to focus on regions of interest around the vehicle. The approach used by the robotics institute of Carnegie Mellon University [Jochem *et al.*, 1993] used a single camera setup. The lane position was determined using a neural network that had undergone a training phase based on features in the image running parallel to the road. The ARGO vehicle at the Universita di Parma [Broggi *et al.*, 1999] [Bertozzi *et al.*, 2000] used the images from a single camera in which the perspective effect had been removed. The lane marking features detected in these images were then matched to a straight-line road model. The SCARF system [Crisman and Thorpe, 1991] and MOSFET vehicle [Beuvais and Kreucher, 1997] used a color camera information under the assumption of a homogeneously colored road. The former system used Bayesian classification to determine the road-surface whereas the latter approach used color segmentation to identify lane markings and fit a parabola to the detected lines. The Australian National University (ANU) approach [Apostoloff and Zelinsky, 2002] is based on a single camera setup. The vision module relies on different image filter information to assess the state of the vehicle. Possible states of the system are described by a set of hypotheses which are generated by a probabilistic particle filter.

5.3.1 Vision Module Description

The approach used in this paper is based on the ANU approach but the main difference is in testing the hypothesis on the vehicle position against different image filters, in particular the Hough transform testing that provides a more robust evaluation. The presented module is the vision core of a larger navigation module that is being developed.

The single CCD camera RGB image is used for two principal image filter types: RGB image for the color filter and grayscale image for the Canny edge detector and Laplacian of Gaussian (LoG) edge filter. Each filter that is used to test current assumptions about position of the vehicle is called a cue.

Hypotheses about the state of the system are assigned a certain probability measure and are called particles. The particle filter handles probabilistic inference in such a way that testing of each particle against each cue produces the final best belief about the state of the system.

A cue-based probabilistic architecture has several advantages. Extracting different information about the environment enriches the information fusion. The technique allows also including any non-image based useful information, such as a-priori knowledge about the road configuration, number of lanes, etc., to be included as a separate cue. The probabilistic fusion of different cues should give a liable result even in cases where certain cues perform poorly which can also give an indication when some cues should be given less credit or even switched off [Apostoloff and Zelinsky, 2002].

The cues used in this work are all image based with the confinement being that the road width is within a region less than two standard road widths which reduces the possible search space. This prevents the system detecting a whole road as a lane since the aim is to determine position of the vehicle within a single lane. The road lane is assumed to be straight and flat which is valid on low curvature roads where the look-ahead distance is not too great [Xu *et al.*, 2000]. Using this assumption only three parameters are needed to specify the vehicle's ego state. These are the road lane width L, the lateral offset of the vehicle d from the middle of the lane, and its angle to the lane φ. Although the straight-line road model may be adequate in many practical situations it is not suitable in particular for high-curvature roads, roundabouts and intersections where extended road models should be applied.

The layout for the lane detection system is shown in Fig. 5.1. The raw image from the camera is processed as RGB and grayscale image before being used by three image cues: Canny edge filter-Hough transform cue, LoG edge filter cue and Colour segmentation cue. A particle filter handles the hypotheses about the vehicle state and passes these particles to the cues for testing. Each cue tests all

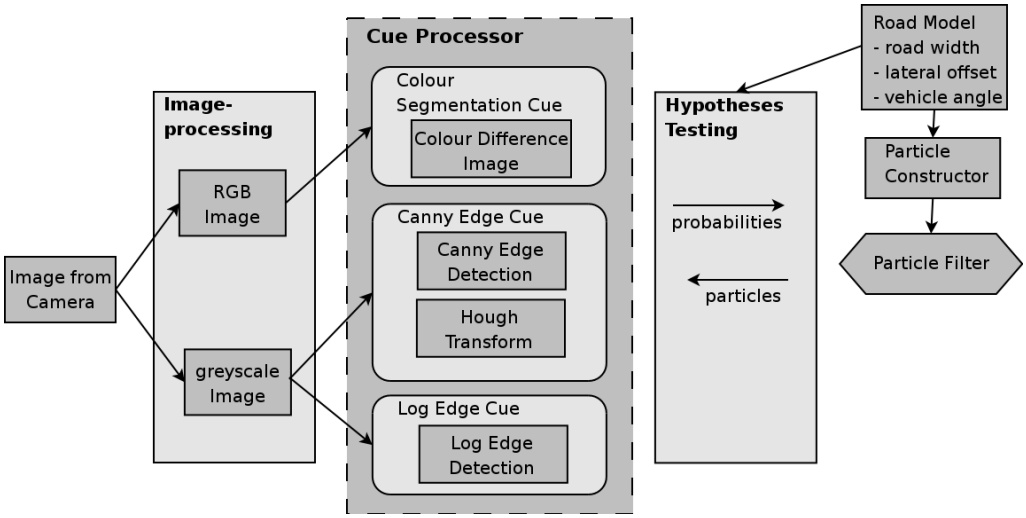


Figure 5.1: The lane detection vision module schema.

of the particles and assigns a probability to each. The final belief is then formed by the particle filter based on total evaluation from each separate cue.

5.3.2 Particle Filtering

Particle filtering is also known as Condensation or Monte Carlo algorithm and is based on Bayesian probabilistic reasoning under Markov assumption (past and future data are independent if the current state of the system is known) [Apostoloff and Zelinsky, 2002], [Thrun *et al.*, 2001], [Isard and Blake, 1998].

If the previous belief about the state of the system is $Bel(x_{t-1})$, the action model is $P(x_t|x_{t-1}, a_{t-1})$ representing the transition from previous state x_{t-1}, the sensor model is $P(o_t|x_t)$ representing the observations at the current instant t and η_t is the normalization factor, then the recursive Bayesian formula for the final belief $Bel(x_t)$ at the current instant t can be described as:

$$Bel(x_t) = \eta_t P(o_t|x_t) \int P(x_t|x_{t-1}, a_{t-1}) Bel(x_{t-1}) dx_{t-1}. \tag{5.4}$$

Representing the continuous probability density function (pdf) by a set of n weighted samples also called particles is computationally more efficient and proves

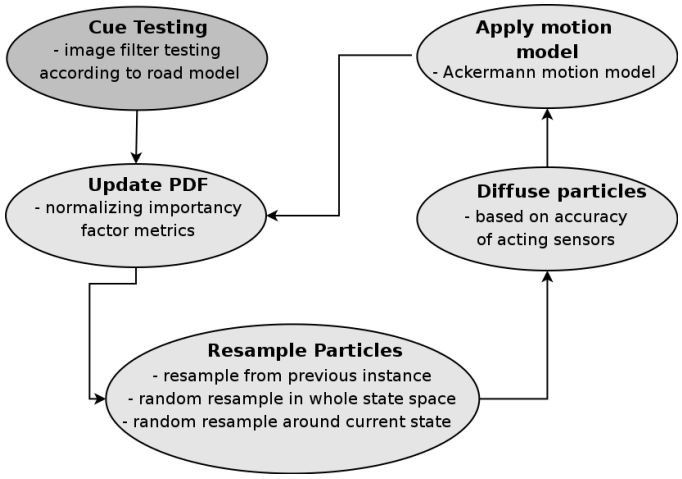


Figure 5.2: The particle filter cycle.

satisfactory also for representation of multi-modal non-Gaussian belief states if the number n is sufficiently large:

$$Bel(x_t) \approx \left\{ x_t^{(i)}, \omega_t^{(i)} \right\}_{i=1,2,\ldots,n}, \tag{5.5}$$

where $x_t^{(i)}$ is a sample state of the random variable x_t, called a pose and $\omega_t^{(i)}$ an importancy weight factor representing probability measure of each particle. In the case of straight-line model of the road, a possible position of the ego-vehicle is a sample in the state space defined as $x_t^{(i)} = \left\{ L_t^{(i)}, d_t^{(i)}, \varphi_t^{(i)} \right\}$ with the probability $\omega_t^{(i)}$ associated to this hypothesis.

The particle filter cycle at each time step comprises of four parts as is shown in Fig. 5.2. Firstly, a group of particles that received best probability estimates from the previous step are resampled from the whole particle set, a fraction of particles are resampled around the best belief particle to refine the search in that region and a certain number of particles are resampled in the whole state space (uniformly random). The later enables resolving the relocalization problem where the particle filter may lost track due to disturbances such as obstacles in the image or brightness intensity change where i.e. standard Kalman filtering may fail. Secondly, particles are diffused according to the error model of the action

sensors (in the automotive case these are typically the translational and rotational velocity of the vehicle). Thirdly, the action model takes into account the motion of the system where for the automotive case this is generally the Ackermann motion model [Kiencke and Nielsen, 2000a].

Finally, when particles are diffused in the state space they each represent a hypothesis that has to be tested against the current vision information from the cues. For an i-th particle $x_t^{(i)}$ the j-th cue gives a probability $P(o_t^{(j)}|x_t^{(i)})$ where the observation $o_t^{(j)}$ depends on the particular image representation of each cue. Total discrete a-posteriori pdf for the sensor model of the whole state space x_t is given as:

$$P(o_t|x_t) = \prod_{j=1}^{m} P(o_t^{(j)}|x_t). \tag{5.6}$$

Combining probabilities from different cues using product operation implies that no cue should return probability 0 since then information would be lost from other cues. Therefore, each cue may return only a probability between $[\alpha, 1]$ with α being the lowest probability measure that may also vary depending on the general performance of each cue (typically set to $\alpha = 0.1$). In general, the combining and normalization step may be performed also by using a weighted sum of contributions of each cue. However, by using product of probability measures to infer the total probability the preferred combinations are those where each single contribution is approximately equally likely. This can be seen when comparing normalized probability cubes of two random variables X_1, X_2 for summation and product case in Fig. 5.3 and Fig. 5.4.

In the summation based inference, the extremal case of the value of one variable being very likely and the other variable being very unlikely may get an equal total probability as in case when they are both equally likely. In the product based inference the extremal cases are more suppressed. Thus, for a liable result all cues must give a sufficiently high confidence. This feature is exploited furthermore in testing of each single cue where the probabilities of detected position for the left

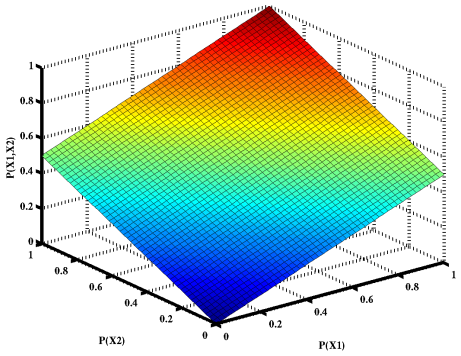


Figure 5.3: Probability cube for summation based inference.

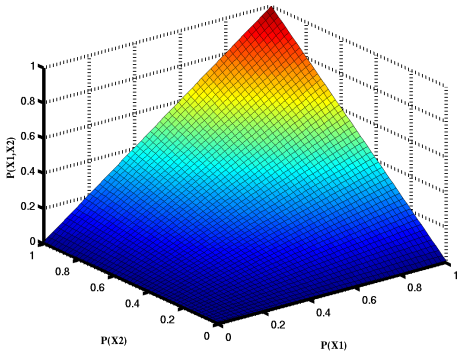


Figure 5.4: Probability cube for product based inference.

and right lane boundary must be high for both boundaries resulting in a more stable and robust inference scheme. The lowest probability measure in cue testing is denoted as p_o (typical value 0.01).

5.3.3 Cue Testing

To test each particle a number of cues can been developed. Each cue measures how well the image information matches what would be expected if the particle correctly described the current state of the vehicle. It uses this to assign a probability as to the likeliness that the particle is correct. This section describes the three image cues that have currently been implemented.

5.3.3.1 Canny Edge - Hough Transform Cue

Canny edge detector processes gray-scale image in multiple stages [Canny, 1986]. The image is first smoothed by a Gaussian convolution mask and then a 2-D first derivative operator is applied to highlight the regions of high contrast - edges in the image. Edges give rise to ridges of high gradient magnitude, however these ridges may be wide and broken, therefore an adaptive threshold nonmaximum suppression is performed by a recursive edge neighbor search to extract accurately

the position of a single pixel based connected edge. Such edges are suitable for line detection algorithms [Haralick and Shapiro, 1992]. A typical highway scene image is shown in Fig. 5.5.

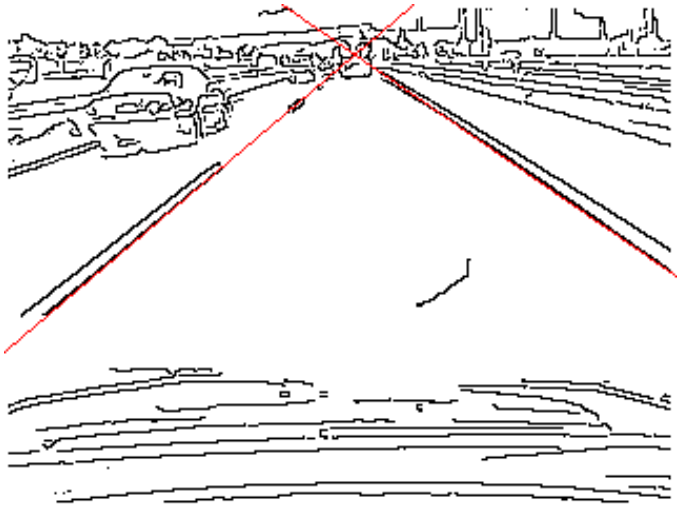


Figure 5.5: A Canny filter edge image with the winning particle (red lines).

For small curvature roads the long straight line segments may represent the road boundaries, lane markings or other objects, smaller size line segments may represent dashed lane markings or other. However, trying to apply a neighborhood based line detection algorithm directly in the image space may not be suitable for two major reasons. Firstly, the length of the line segments in the image is not known in advance. For instance, dashed lane markings that actually represent the same road boundary will result in different line segments in the image that may be difficult to segment in a postprocessing step. Secondly, in a real road scene road boundaries may be occluded by other objects, in particular by other vehicles, which render neighborhood based line algorithms difficult to use. Moreover, trying to overlay a perspective road mask in the image space is not liable since single pixel edges of the Canny filter do not provide enough hits for the mask summation.

In order to detect line segments, the Hough transform [Haralick and Shapiro, 1992] is used in this work since it is invariant to pixel position in an image. This implies that it is not necessary to distinguish between full or dashed lane markings,

marked road edges or natural road boundaries as long as the intensity difference is sufficient for the Canny filter response. Essentially, it transforms edge pixels in image space that lie along the same straight line to a single point in Hough space. The coordinates of this point are the distance ρ of the line to an origin and the angle θ the line makes to the Ox-coordinate axis. Its implementation requires a two dimensional accumulator array where each cell corresponds to a small span of line parameters ρ and θ.

Ordinarily, every possible line going through each edge element is plotted in Hough space as a sine curve. An actual line detected in a particular accumulation cell is then the overlap of all these curves rendering the processing slow. However, by using the angle direction information of the first derivative operator each edge pixel can be transformed directly to a unique accumulator cell by increasing only the total count I of that cell which significantly speeds up the processing time. The Hough transform image of the Canny edge map in Fig. 5.5 is shown in Fig. 5.6 where darker points represent accumulator cells with a higher intensity value I. It is visible that the lane boundaries where the ego-vehicle is placed occupy very confined regions of the accumulator array, i.e. the center of the left lane boundary lying at coordinates $(\theta, \rho) = (59°, 105)$ and the right lane boundary center lying at $(-51°, 78)$ (transformation origin in the upper left corner).

To derive a probability measure of each particle according to the straight-line model, the left and right edges of the hypothesized lane are converted to two Hough point centers $c_l^{(i)} = \left\{\theta_{cl}^{(i)}, \rho_{cl}^{(i)}\right\}$ and $c_r^{(i)} = \left\{\theta_{cr}^{(i)}, \rho_{cr}^{(i)}\right\}$, respectively. Fig. 5.7 shows left (blue color) and right (green color) lane positions for each particle of the particle set. Several particles might have one same edge point in Hough space but differ in other.

Since both lane boundaries may differ significantly in intensity and structure, the particle edges must be tested and normalized separately. For instance, alignment of a particle's hypothesized left edge center $c_l^{(i)}$ to the left lane boundary cluster of N points with intensity values $I^{(k)}$ in Hough space that are found within the

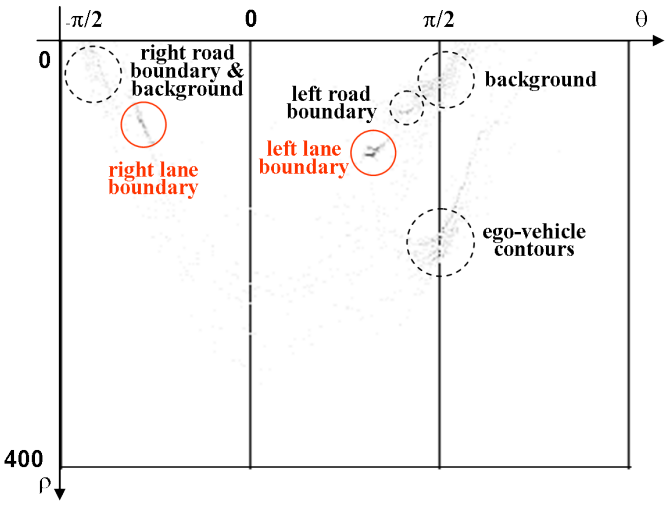


Figure 5.6: Hough transform image.

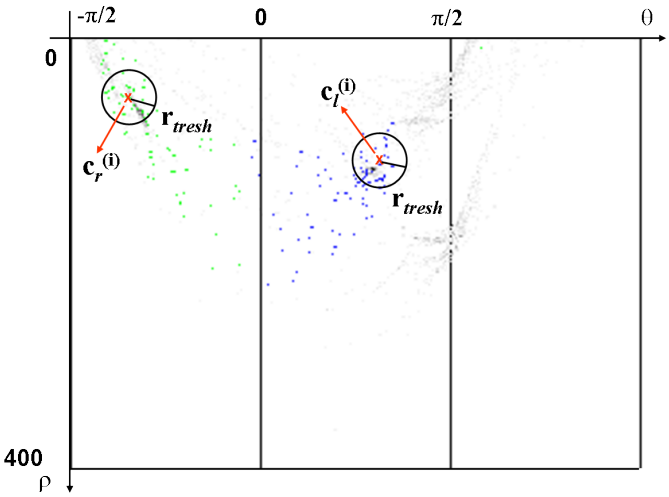


Figure 5.7: Left and right lane positions of the particle set (blue and green color respectively).

circular threshold region with radius $r_{tresh} = \sqrt{(\theta_{tresh})^2 + (\rho_{tresh})^2}$ is best when the particle's edge is placed in the center of the real road cluster. The probability measure describing this is:

$$\delta_{cl}^{(i)} = \sum_{k=1}^{N} \frac{(r_{tresh} - \sqrt{(\theta_{cl}^{(i)} - \theta^{(k)})^2 + (\rho_{cl}^{(i)} - \rho^{(k)})^2})}{r_{tresh}} I^{(k)}. \tag{5.7}$$

The threshold values θ_{tresh} and ρ_{tresh} are determined such that if the road lane

is bounded by a lane marking which generally transforms in two distinguished edges, both edge peaks are included as a single road boundary cluster (i.e. lane marking is taken to occupy 8% of the standard road width). This can be distinctly seen in Fig. 5.6 for the left lane boundary (the cluster radius is enlarged for clarity).

The total probability ${\omega_{Canny}}^{(i)}$ of a particle for the cue is determined by taking both left and right edge boundaries into account:

$$ {\omega_{Canny}}^{(i)} = (\frac{\delta_{cl}^{(i)} - \delta_{clmin}}{\delta_{clmax} - \delta_{clmin}} + p_o)(\frac{\delta_{cr}^{(i)} - \delta_{crmin}}{\delta_{crmax} - \delta_{crmin}} + p_o). \tag{5.8} $$

Using a first derivative operator, in this case the Sobel operator, to determine the gradient direction inherently involves an error [Haralick and Shapiro, 1992]. This reflects upon the Hough transformation where pixels with same gradient direction error margin in the image space have a different spread in the ρ component depending on the choice of transformation origin and the pixel position in the image whereas the θ component spread depends only on the gradient direction error itself. For instance, if the upper left corner of the image is chosen as the transformation origin the pixels most sensitive to this error belong to the right lane edge. Since the edge's cluster has a larger spread (*cf* Fig. 5.6) the measure of alignment to the circular threshold region is less accurate resulting also in less accurate right lane boundary detection.

Therefore, to increase robustness of the estimated lane position two Hough transforms are used for each particle. The probability evaluations become $\omega_L^{(i)}$ and $\omega_R^{(i)}$ where L and R denote the transformation origin at the upper left and right image corner, respectively. These origins are chosen as to minimize the ρ error sensitivity at values of θ at 45°. The nominal angle in real road scene is around 55° for the left lane edge to the L origin and right lane edge to the R origin. Attempting to use a single centrally placed transformation origin is also not robust enough and transforms points to all 4 angle quadrants (as opposed to

3 that are needed here). The final probability measure then becomes:

$$\omega^{(i)}{}_{Canny} = \omega_L^{(i)} \omega_R^{(i)}. \tag{5.9}$$

5.3.3.2 Laplacian of Gaussian Edge Cue

The LoG filter performs a Laplacian *2nd*-order spacial derivative of a Gaussian smoothed grayscale image. A zero-crossing detection of the gradient magnitude image enables extraction of edges which in general may be thicker than single pixel based edges of the Canny edge detector. Thus, this type of edges is suitable for comparison with a perspective model mask that is overlaid directly in image space. The mask is taken to be left and right stripes that are lane marking size wide, determining a generic non-lane region which may also be the road boundary with no lane markings. Fig. 5.8 depicts a LoG edge map which is overlaid by the winning particle's perspective model mask (yellow). The stripes height is limited to exclude the far-sight region close to the vanishing point where other vehicles may represent false lane boundary detection.

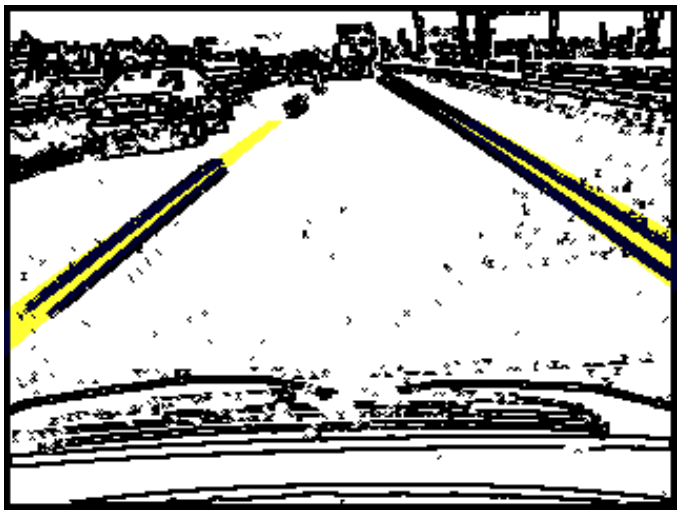


Figure 5.8: A LoG filer edge image with the perspective model mask of the winning particle.

The edge map can be regarded as a binary map, i.e. $I^{(k)} = \{0, 1\}$ where every

edge pixel represents a positive hit. Each particle generates a different perspective model mask where a simple measure of edge pixel count within the total mask pixel count $N_{l,r}$ of the left l and right r stripe represents the probability measure for each side of the lane:

$$\delta_{l,r}^{(i)} = \sum_{k=1}^{N_{l,r}} I_{l,r}^{(k)}. \tag{5.10}$$

The probability measure of the i-th particle is then:

$$\omega_{LoG}{}^{(i)} = (\frac{\delta_l^{(i)} - \delta_{lmin}}{\delta_{lmax} - \delta_{lmin}} + p_o)(\frac{\delta_r^{(i)} - \delta_{rmin}}{\delta_{rmax} - \delta_{rmin}} + p_o). \tag{5.11}$$

The zero-crossing detection phase involves searching for pixel neighborhood where the *2-nd*-order derivative of the image changes sign. By using a larger neighborhood mask and appropriate smoothing noise variance the edges can be further enhanced, i.e. thickened for a more robust comparison to the perspective model mask.

5.3.3.3 Colour Cue

Colour cue performs comparison of the RGB input image to the mean values $\overline{R}$, $\overline{G}$, $\overline{B}$ of each component of the road surface color. If a pixel's RGB values lie within the threshold defined by all three variances σ_R, σ_G , σ_B it is considered to be of the road color. Thus, a binary delta map can be acquired which is used for particle testing. The perspective mask that is overlaid on the delta map in this case includes both left and right lane boundaries which must be of non-road color and the central region which must be of road color (*cf* Fig. 5.9).

Similar to the LoG filter, for each part of the perspective mask a $\delta_{l,r,c}^{(i)}$ measure is calculated representing pixel hit of non-road or road color within the total mask area and the final probability measure of the i-th particle is:

Figure 5.9: A delta color image with the perspective model mask of the winning particle.

$$\omega_{Colour}{}^{(i)} = (\frac{\delta_l^{(i)} - \delta_{lmin}}{\delta_{lmax} - \delta_{lmin}} + p_o)(\frac{\delta_r^{(i)} - \delta_{rmin}}{\delta_{rmax} - \delta_{rmin}} + p_o) (\frac{\delta_c^{(i)} - \delta_{cmin}}{\delta_{cmax} - \delta_{cmin}} + p_o). \quad (5.12)$$

At each instant a new $\overline{R}$, $\overline{G}$, $\overline{B}$ and σ_R, σ_G , σ_B are calculated based on the winning particle information, rendering the cue adaptive to lightning condition changes.

5.3.4 Experimental results

The lane detection module has been tested using a single SONY DFW-VL500 camera mounted in the rear view mirror position inside of the testing vehicle. The module was tested in several typical real-road conditions that are shown in Fig. 5.10 through Fig. 5.21.

The lane detection module proved robust under different road conditions. If a cue performs poorly at a certain instant the contribution of other cues to the probabilistic measure diminishes its negative effect to the overall performance. This

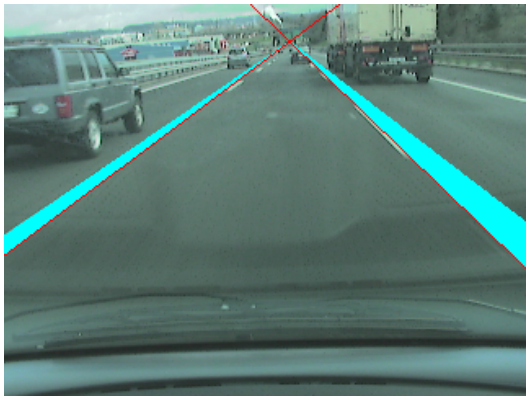


Figure 5.10: Highway - heavy traffic.

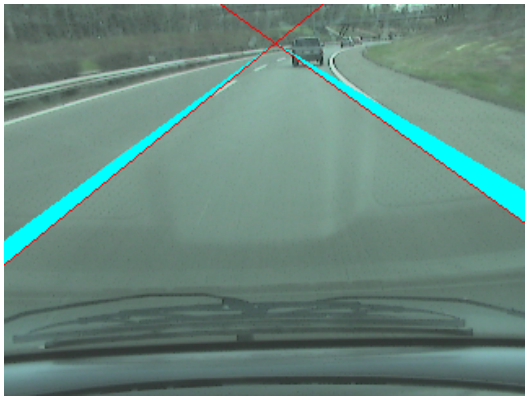


Figure 5.11: Highway - high curvature.

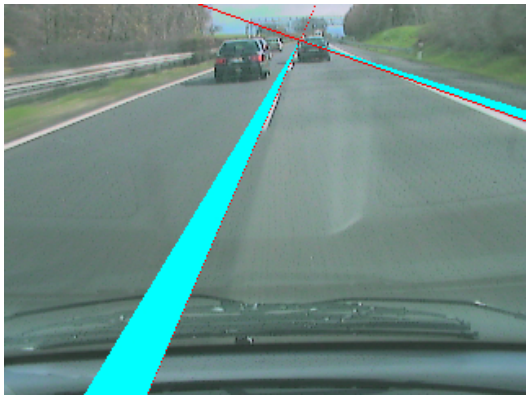


Figure 5.12: Highway - changing lanes I.

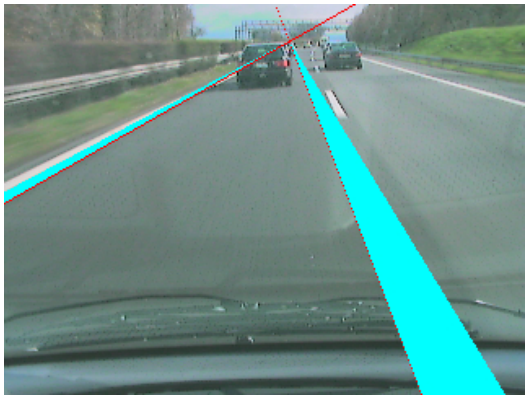


Figure 5.13: Highway - changing lanes II

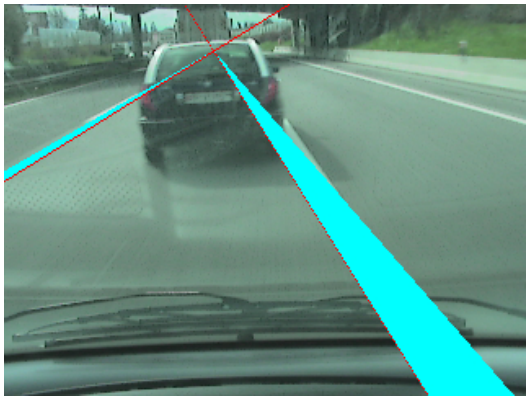


Figure 5.14: Highway - a front car occluding the view.

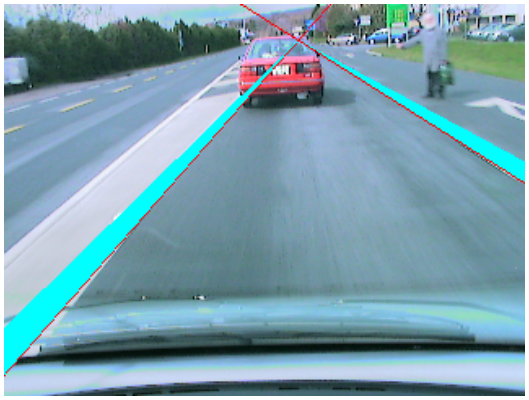


Figure 5.15: Magistral road - inner city.

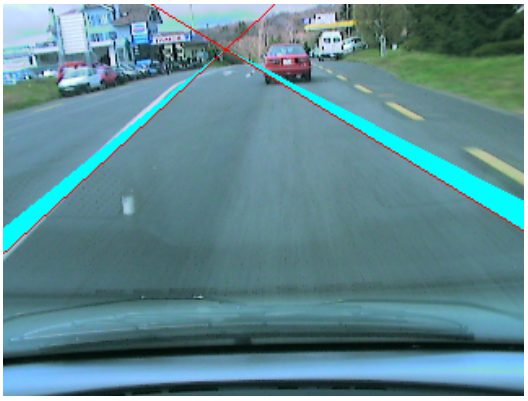


Figure 5.16: Magistral road - ambiguous lane border position.

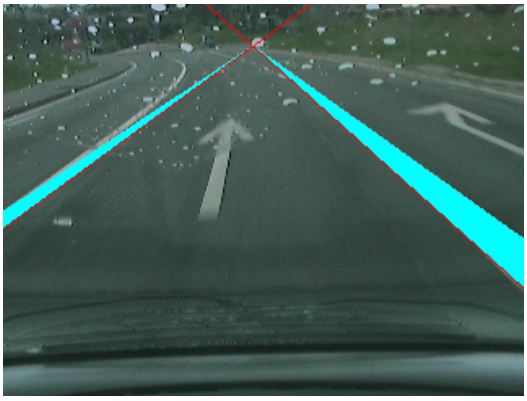


Figure 5.17: Magistral road - ground signs.

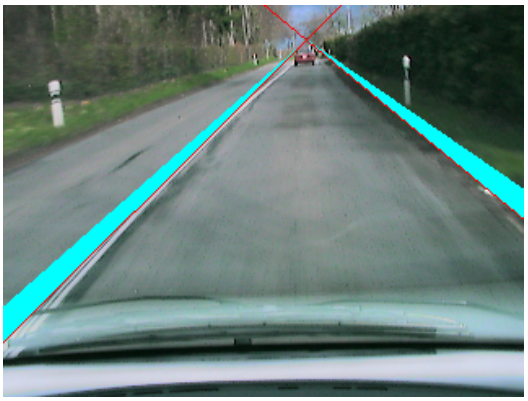


Figure 5.18: Magistral road - country lane.

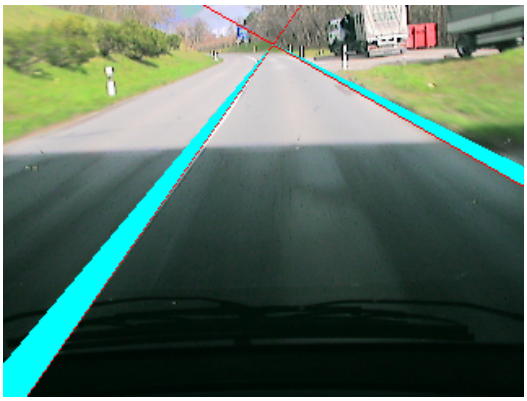


Figure 5.19: Magistral road - leaving a small tunnel.

is particularly the case with the Colour cue which is quite sensitive to brightness change and shadow areas on the road. A multimodal color histogram distribution instead of a single mean for each color component may improve its performance. In general, the Canny edge filter in combination with Hough transform proved to be the most robust cue that responds immediately to a change in scene since it retains information both about edge magnitude and direction, adapts to different levels of noise and is pixel position insensitive. LoG filter is isotropic and does not contain the edge direction information, thus it may become less stable when other obstacles occlude the lane boundary view, in changing lanes situations or on roads

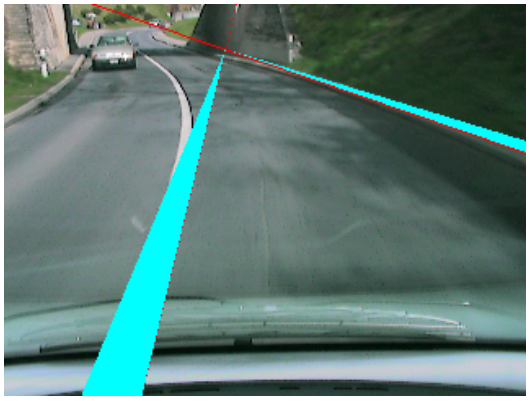


Figure 5.20: Magistral road - high curvature.

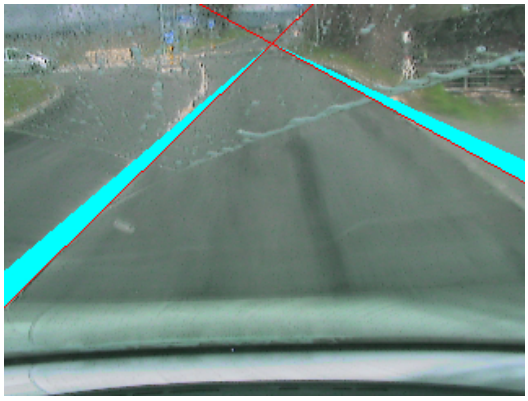


Figure 5.21: Magistral road - dirty windscreen.

with large shadow areas. Moreover, the LoG and Colour cue are tested directly in the image space on pixel masks which increases significantly the processing time in comparison to small edge pixel number of the Canny cue.

The case of dirty front window showed potential to tackle worse visibility conditions particularly due the adaptive Canny filter which is also suitable for abrupt lightning changes such as small tunnels, however, this assumption was not extensively tested. The system detected correctly the near-sight contours of the lane even for higher curvature roads which may be sufficient information for an i.e. lane keeping module. However, using the straight-line model does not allow to place correctly the objects in the far-sight view in the overall environment representation.

In summary, in this section a vision module was presented for lane detection in vehicles that forms the basis of a driver assistance navigation module. Edge detection and color image filters were used to extract information about the road environment of the ego vehicle. The information was processed within a probabilistic framework using particle filtering where the belief about the state of the system was described by a discrete set of particles each representing a possible solution within the state space. Particles were assigned a probabilistic measure according to evaluation against image filters called cues and the particle with the

best total probability measure from all cues described the most likely position of the vehicle.

The single camera experimental setup mounted in a vehicle was tested in various real-road conditions, such as highway traffic scenes and lane changing, magistral road in inner town with different ground signs and magistral road in outer city areas with different lightning conditions, road shadows and windscreen visibility. The vision system performed robustly in most cases except in situations where road edges were too obscured by dark areas in outer city areas and in ambiguous positions on the border between two lanes where flickering between two lanes may have occurred since no vehicle dynamics was included due to lack of vehicle motion sensors.

In comparison with the LoG edge cue and Colour segmentation cue that are based on particle testing in image space, the Canny edge filter with double Hough transform particle testing performed significantly better in terms of robustness and computation speed. This cue presents a good basis for development of a higher curvature road model from the straight-line road model that is currently being used.

5.4 Map building

The map of interest in this work used for autonomous navigation is a local environment map that is rebuild on-line according to the ego-vehicle motion, typically in a range of 100 m-150 m in front of the ego-vehicle. The information about the environment is gained in form of range point data, that is gathered via a laser sensor, such as a SICK, ALASCA XT or as an output of a stereo-rig triangulated feature points of the scene. The data may arrive asynchronously with respect to the map update cycle, here simply noted as a discrete time index t. The individual sensor point data is gathered in respective buffers as vehicle pose registered point-cloud data. As an illustration, the case of Smart vehicle implementation features

following sensory cycles:

- Localization cycle: 100 Hz
- SICK laser cycle: 75 Hz
- ALASCA XT laser cycle: 25 Hz
- Stereo-rig cycle: 7.5 Hz
- Local map recompute cycle: 5 Hz

In order to build a 3D consistent world representation, a voxel 3D grid map $\mathcal{M}$ is used with different horizontal x-y and z-plane cell sizes, therefore each voxel map cell represents a rectangular box volume in space. The anchor of the map is redefined automatically in 3D global GPS space, also taking into account the vehicle orientation as soon as the ego-vehicle exits the current local map bounds.

The pointcloud measurements at each map recompute cycle can be denoted according to standard notation as:

$$z_t = \{z_t^1, \ldots, z_t^j, \ldots, z_t^K\} \tag{5.13}$$

K corresponding to the total number of points in all the registered pointclouds at t and $z_t^j = \{\tilde{x}_t^j, \tilde{y}_t^j, \tilde{z}_t^j\}$, coordinates of individual points in space. In order to fuse the new measurements as occupancy evidence in the voxel map, the Bayesian theorem is applied. The derivation follows the steps defined in [Vu *et al.*, 2007] and [Thrun, 2000], for a 2D gridmap occupancy representation, which applies the same in 3D case as well.

Given all the observations in use for fusion of the current local map $z_{1:t} = \{z_1, \ldots, z_t\}$ and the known vehicle poses $x_{1:t} = \{x_1, \ldots, x_t\}$, the posterior probability of occupancy $P(m|z_{1:t}, x_{1:t})$ for each voxel cell m can be determined as:

$$P(m|z_{1:t}, x_{1:t}) = \frac{P(z_t|z_{1:t-1}, x_{1:t}, m)P(m|z_{1:t-1}, x_{1:t})}{P(z_t|z_{1:t-1}, x_{1:t})} \tag{5.14}$$

where the expression $P(z_t|z_{1:t-1}, x_{1:t}, m)$ is the sensor model that can further be simplified. Namely, given the assumption that the current measurement z_t is independent of previous measurements $z_{1:t-1}$ and poses $x_{1:t-1}$, the sensor model probability $P(z_t|x_t, m)$ can be factored as:

$$P(z_t|x_t, m) = \frac{P(z_t, x_t, m)}{P(x_t, m)} = \frac{P(m|z_t, x_t)P(z_t|x_t)P(x_t)}{P(x_t)P(m)} \tag{5.15}$$

where $P(x_t, m) = P(x_t)P(m)$ is independent, since the current pose x_t is assumed known, depending only on previously applied control action. This assumption is valid for the map building problem considered here, as opposed to a full SLAM (Simultaneous Localization and Mapping) problem where current pose x_t is also estimated with respect to the map $\mathcal{M}$ being built.

According to Eq. 5.15, the Eq. 5.14 can now be expressed as:

$$P(m|z_{1:t}, x_{1:t}) = \frac{P(m|z_t, x_t)P(z_t|x_t)P(m|z_{1:t-1}, x_{1:t})}{P(m)P(z_t|z_{1:t-1}, x_{1:t})} \tag{5.16}$$

which represents the probability that a given voxel cell is occupied. According to Eq. 5.16, similar reasoning can be applied to express the probability that a cell is free:

$$P(\bar{m}|z_{1:t}, x_{1:t}) = \frac{P(\bar{m}|z_t, x_t)P(z_t|x_t)P(\bar{m}|z_{1:t-1}, x_{1:t})}{P(\bar{m})P(z_t|z_{1:t-1}, x_{1:t})} \tag{5.17}$$

It is interesting to express the quotient of the two probabilities as:

$$\frac{P(m|z_{1:t}, x_{1:t})}{P(\bar{m}|z_{1:t}, x_{1:t})} = \frac{P(m|z_t, x_t)}{P(\bar{m}|z_t, x_t)} \frac{P(\bar{m})}{P(m)} \frac{P(m|z_{1:t-1}, x_{1:t-1})}{P(\bar{m}|z_{1:t-1}, x_{1:t-1})} \tag{5.18}$$

also taking into account that $P(m|z_{1:t-1}, x_{1:t}) = P(m|z_{1:t-1}, x_{1:t-1})$, since the map occupancy probability, given only previous time steps measurements $z_{1:t-1}$ is independent of pose x_t.

The quotient of probabilities of a probabilistic event ω and its complement $\bar{\omega}$ is also known as the *Odds* representation:

$$Odds(\omega) = \frac{P(\omega)}{P(\bar{\omega})} = \frac{P(\omega)}{1 - P(\omega)} \tag{5.19}$$

Therefore the Eq. 5.18 can be expressed as:

$$Odds(m|z_{1:t}, x_{1:t}) = Odds(m|z_t, x_t)Odds(m)^{-1}Odds(m|z_{1:t-1}, x_{1:t-1}) \tag{5.20}$$

which in its logarithmic form yields an iterative map occupancy update scheme:

$$\log Odds(m|z_{1:t}, x_{1:t}) = \log Odds(m|z_t, x_t) - \log Odds(m) + \log Odds(m|z_{1:t-1}, x_{1:t-1}) \tag{5.21}$$

The $\log Odds(m|z_{1:t-1}, x_{1:t-1})$ refers to the $\log Odds$ map value of the previous time step $t-1$, the $\log Odds(m)$ is related to the a-priori map probability $P(m)$, typically set to 0.5 as being "unknown". The probability $P(m|z_t, x_t)$ represents the inverse sensor model.

As mentioned earlier, all sensor measurements in the case of this work are fused in the voxel map as pointcloud pose registered data $z_t^j = \{\tilde{x}_t^j, \tilde{y}_t^j, \tilde{z}_t^j\}$ representing an individual point coordinate with respect to the pointcloud origin x_t^j. Therefore, the inverse sensor model represents the probability of map occupancy at all the map voxel cells affected by ray tracing from a measurement point to the pointcloud origin. The occupancy probability is modeled by a normally distributed probability density, namely:

$$P(m(d)|z_t^j, x_t^j) = \mathcal{N}(d, \sigma^j) \tag{5.22}$$

where d is the distance of a voxel cell from the measurement point z_t^j and σ^j the estimated radial point variance. This inverse sensor model corresponds well to

a laser based beam measurement. More involved inverse sensor models can in practice render the inference on many thousands of pointcloud points impractical.

In order to consistently fuse incoming measurement data in a 3D environment, a 3D environment representation is needed, in this case in a form of a 3D voxel map, which can be used for various analysis of the environment. In the case of this work, a more compact occupancy information is also derived in form of a 2D gridmap. Among different inference methods to "flatten" the voxelmap columns onto the 2D gridmap, also called flatmap $\mathcal{F}$ here, the conservative max operator is chosen:

$$P_{\mathcal{F},i,j}(m) = \max_{k} P_{\mathcal{M},i,j,k}(m) \tag{5.23}$$

where the $P_{\mathcal{M},i,j,k}(m)$ represents the occupancy probability of a voxel map cell with indices (i, j, k) and $P_{\mathcal{F},i,j}(m)$ the occupancy probability of the corresponding flatmap cell with indices (i, j).

Note, that the voxelmap orientation (map anchor) is computed with respect to the Earth's geoid (approximately gravity oriented), such that the map anchor is re-initialized based on global vehicle $[x, y, z, \psi]^T$ coordinates (including the yaw angle), whereas the roll ϕ and pitch angles θ are set to zero. This implies that that the vehicle travels within the voxelmap on the tangential plane defined by vehicle's roll and pitch angles, whereas the flatmap $\mathcal{F}$ projection is defined with respect to the gravity vector. This allows for consistent 2D approximation of the local static 3D space since (buildings vertical to gravity). The effect of the pitch and roll angle on the projection of dynamic objects on a slope is considered less critical.

5.5 Detection and tracking of dynamic objects

5.5.1 Detection of dynamic regions of space

In Sec. 5.4 building of a local environment map in form of a 3D voxelmap was presented. This section will be concerned with extracting dynamic vs. static parts of the environment based on the temporal differences in the map occupancy. This can be considered as a first step towards detection of dynamic objects in a static background scene.

Based on the occupancy probability $P_{i,j,k}(m)$ of each voxel cell and consequently the occupancy probability of the flatmap $P_{\mathcal{F},i,j}(m)$, there is a label $L^{occ}(P(m))$ defined that can describe the occupancy semantically from the label set of L^{occ} as $\{UNKNOWN, FREE, BLURRED, OCCUPIED\}$:

Occupancy probability $P(m)$	**Label** L^{occ}
initial $P(m) = 0.5$	UNKNOWN
$P(m) \leq p_{free}$	FREE
$p_{free} < P(m) < p_{occ}$	BLURRED
$p_{occ} \leq P(m)$	OCCUPIED

Table 5.1: Semantic labelling of the map occupancy.

where p_{free} and p_{occ} are design parameters determining free or occupied cells, respectively (typical values could be $p_{free} = 0.2$ and $p_{occ} = 0.8$).

In order to determine whether there were any temporal changes in the projected flatmap $\mathcal{F}$, the occupancy labels for each cell can be analyzed at times $[t, t-1, \ldots]$ from the corresponding occupancy labels L stored $[L_t^{occ}, L_{t-1}^{occ}, \ldots]$ history. In principle, the whole local map history could be taken into account for determining current temporal changes, however at this time we will consider only the "first-order" temporal changes, namely by analyzing only the $[L_t^{occ}, L_{t-1}^{occ}]$ pair for each map cell, the current versus previous map update cycle. Temporal change are described by the dynamicity label in the semantic set $L^d \in \{STATIC, TRANSIENT, DYNAMIC\_UP, DYNAMIC\_DOWN\}$. The dy-

namicity label L^d of each map cell is thus determined according to the following inference Tab. 5.2:

Occupancy label L^{occ}_{t-1}	**Occupancy label** L^{occ}_{t}	**Dynamicity label** L^{d}_{t}
UNKNOWN	FREE	STATIC
UNKNOWN	BLURRED	TRANSIENT
UNKNOWN	OCCUPIED	TRANSIENT
FREE	FREE	STATIC
FREE	BLURRED	TRANSIENT
FREE	OCCUPIED	DYNAMIC_UP
BLURRED	FREE	TRANSIENT
BLURRED	BLURRED	TRANSIENT
BLURRED	OCCUPIED	TRANSIENT
OCCUPIED	FREE	DYNAMIC_DOWN
OCCUPIED	BLURRED	TRANSIENT
OCCUPIED	OCCUPIED	STATIC

Table 5.2: Semantic labelling of the map dynamicity (temporal changes).

In the current implementation, the only well defined dynamic cases are $DYNAMIC\_UP$ and $DYNAMIC\_DOWN$ that mark the occupancy change from $FREE$ to $OCCUPIED$ and vice versa. It is important to remark that the dynamic objects considered here are dynamic in the absolute sense, that is with respect to the local map and not relative to the ego-vehicle motion. For instance, a vehicle moving in front of an ego-vehicle at a relative speed 0 with respect to the ego-vehicle is still considered dynamic with respect to the local map.

According to Tab. 5.2 the $TRANSIENT$ cases can be considered potentially both static or dynamic. To resolve the ambiguity, the future work would include exploring past cell dynamicity labels as well, namely $[L^d_t, L^d_{t-1}, \ldots]$, following the intuition that dynamic areas in the past are more likely to be dynamic at the current cycle as well (same intuition applying to the static areas also). Furthermore, in a complete loop from past object tracks to current measurement observation many ambiguity areas could be resolved as well.

The step following the map cell labelling is the point measurement labelling that were used to update the map in the current cycle. All the measurements

points (hits) found within a map cell are labelled in correspondence to the map cell label itself. The exception are the $DYNAMIC\_DOWN$ labelled map cells which require an additional search in th vicinity of the current map cell. All the measurement points found in the vicinity are labelled $DYNAMIC\_DOWN$ if their corresponding cell occupancy is labelled $OCCUPIED$ or $BLURRED$. This step is necessary to account for the fact that $DYNAMIC\_DOWN$ marks the presence of moving objects in the past that already left the analyzed map cell.

Similar approach to temporal map occupancy exploration using labelling was implemented in [Burlet *et al.*, 2007]. However, in [Burlet *et al.*, 2007] dynamicity labelling was done only on current measurement data. Accounting for past dynamic events was done via a histogram binning, whereas the implementation in this work allows for analysis of the full dynamic history of a map cell with map cell labels associated with precise timestamping. Most importantly, the $DYNAMIC\_DOWN$ case was not modelled explicitly in [Burlet *et al.*, 2007].

5.5.2 Object extraction

Object extraction following the dynamicity labelling is done on the measurement points and not on the map cell level. This is to achieve higher precision object geometry that is independent on map resolution limits as well as to account for cases such as $DYNAMIC\_DOWN$ that would on the contrary present "ghost objects" from the past motion.

The measured points with labels of interest $STATIC$, $TRANSIENT$, $DYNAMIC\_UP$, $DYNAMIC\_DOWN$ are now processed separately for each individual label to obtain object contours based on their spacial relation. For this purpose a KD-Tree (Kernel-Density Tree) structure is used to estimate the initial spacial data density. The resultant spacial decomposition is used as the seed value for the K-means clustering algorithm in order to obtain the final clusters (note that the number of clusters was not known a-priori) [Redmond and Heneghan, 2007]. In order to obtain the object contours a bounding box was used for each

final cluster of points. A PCA (Principal Component Analysis) step was run on the points in order to compute the first two principal axis, giving the cluster the size and orientation. To speed up the object extraction phase, the whole clustering procedure was run on the x-y coordinates of the measurement points only, thereby implicitly assuming the objects are not hollow.

5.5.3 Dynamic object tracking

In the current state of development, the object tracking on real world measurement data from urban scenarios is not yet tested. In order to close the loop from raw measurement data to object tracking the following additional steps are yet to be developed:

- *Object size and orientation estimation*: as mentioned in Sec. 5.5.2 the object contours and orientation are based on PCA analysis and a bounding box approach. This procedure is a good initial seed to estimation of the object position, size and orientation. However, since only a partial view of the objects (in particular oncoming vehicles) is available at any given moment, the bounding box approach may be insufficient. Additional feature analysis should be performed including line or corner feature segmentation within each cluster, among others.

- *Data association and track management*: Associating the currently extracted object to existing tracks, track update, initialization and pruning is a vast topic on its own and will not be analyzed further here. Suffice to say here that one of very promising approaches is the IMM (Interacting Multiple Model) filtering [Blackman and Popoli, 1999].

Nevertheless, in order to render the dynamic object motion estimation in the simulated environment close to real world estimation, the simulated object motion prediction of Chap. 4 and Chap. 6 was computed via a Kalman Filtering target tracking module. The simulated data provided by the simulator engine were the

position, bounding box size and identification number of an object, such as a vehicle or a pedestrian (no orientation was given a priori). Clearly, given the identification number information, the data association problem and track management reduced only to initialization of a new track for a new id and discarding the old tracks for ids not present anymore.

In order to estimate the state of an individual tracked object the Constant-Velocity (CV) target motion model is used, also known as Piecewise Constant White Acceleration Model, which assumes that the acceleration between sampling intervals with time constant T is unknown, thus white noise modeled. The state of a tracked object is defined as:

$$x^o = [x, vx, y, vy]^T \tag{5.24}$$

thereby defined by it's 2D position $[x, y]$ and velocities $[vx, vy]$. Formulating the state estimation equation as:

$$x^o_{k+1} = \Phi x^o_k + \Gamma \nu_k \tag{5.25}$$

where ν_k is the process noise with zero-mean acceleration and noise gain [Singer, 1970]:

$$\Gamma = \begin{bmatrix} \Gamma_x \\ \Gamma_y \end{bmatrix} \ , \ \Gamma_x = \Gamma_y = \begin{bmatrix} \frac{1}{2}T^2 \\ T \end{bmatrix}$$

which implies the process noise:

$$Q = \mathbb{E}[\Gamma \nu_k \nu_k \Gamma^T] = \begin{bmatrix} Q_x & \mathbf{0} \\ \mathbf{0} & Q_y \end{bmatrix}$$

as being:

$$Q_x = Q_y = \begin{bmatrix} \frac{1}{4}T^4 & \frac{1}{2}T^3 \\ \frac{1}{2}T^3 & T^2 \end{bmatrix} \sigma_v^2$$

The recommendation for the choice of σ_v with respect to the maximum target acceleration a_{max} is [Blackman and Popoli, 1999]:

$$0.5a_{max} \leq \sigma_v \leq a_{max} \tag{5.26}$$

The system transition matrix is a simple integrator:

$$\Phi = \begin{bmatrix} \Phi_x & \mathbf{0} \\ \mathbf{0} & \Phi_y \end{bmatrix}, \; \Phi_x = \Phi_y = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}$$

The current heading Ψ of the tracked object is estimated via the velocity components $\Psi = \arctan \frac{v_y}{v_x}$. Since the motion planning requires the state estimate prediction of the track for the duration of the object prediction horizon, at each filter update the heading is estimated anew and the object trajectory in the future is extrapolated linearly according to the current position, velocity and heading. Simulated results on object tracking used in Chap. 6 proved that given a sufficiently high filter update cycle (in our case up to 50 Hz), the heading of the object can be estimated accurately.

As an estimation enhancement, if also the orientation of the object is available as the input information, it is interesting to estimate the current turning rate $\dot{\psi} = \omega$ of a tracked object, which can be useful for estimation of turning maneuvers around the corners at intersections and predicting for instance which lane a given object might take. However, this aspect was not investigated here.

5.6 Experimental results on dynamic scene analysis

The experimental results in this section represent the fusion between the localization module of Sec. 5.2, the local map building module of Sec. 5.4 and the dynamic object detection and extraction of Sec. 5.5. The results are in Fig. 5.22 to Fig. 5.31 are divided into three views: top - the left (driver side) camera view, middle - sensory measurements (point clouds) and extracted objects at the current map update and clustering cycle, bottom - the flatmap occupancy values for the whole history of the local map building with vehicle position traces. The colors in the measurement and object extraction figures represent the following dynamic labels: $STATIC$ - blue, $TRANSIENT$ - green, $DYNAMIC\_UP$ - red, $DYNAMIC\_DOWN$ - magenta.

Fig. 5.22 and 5.23 depict detection of a moving vehicle in otherwise static environment in two distinct snapshots. Fig. 5.24 and 5.25 depict detection of a moving tram in the opposite lane. Since the tram is a fairly long vehicle, only the front part of its silhouette is detected as being dynamic, whereas the longitudinal parts whose occupancy evidence was longer respect to the map cycle update, are labelled static. Also, the static part of the environment on the right side of the moving ego-vehicle (pavement) can induce spurious dynamic readings in Fig. 5.24 which is corrected as the more evidence is gathered Fig. 5.25. In comparison, the left side wall with a clearly distinct vertical structure in the upper part of the measurement figure is detected accurately as being static.

Fig. 5.26 and 5.27 depict detection of moving vehicles at an intersection. The same effect of dynamic detection can be visible, namely, the parts of the vehicle that with the biggest motion change with respect to the ego-vehicles view are detected dynamic (typically the front and the rear of the vehicles), whereas the sides can be labelled static as well. Fig. 5.28 depicts detection of a pedestrian at a zebra crossing that is on the move and an approaching vehicle in the opposite lane. The tram is in this case stopping down at a tram station.

Fig. 5.29 to 5.31 depict detection of vehicles negotiating a lane intersection

and a pedestrian moving into the field of view of the ego-vehicle. The horizontal occupancy cell size in this case is reduced (0.5 as opposed to the nominal 0.2) in order to show that as long as the sensory measurements are labelled properly in the sense of their dynamicity, the object clustering/extraction process is independent with respect to the occupancy map resolution, which is the consequence of how the voxel/flatmap and sensory measurement history are build (according to Sec. 5.4). On the other side, an occupancy map too coarse would not reflect the spatial object placements accurately enough.

As a general observation, it can be seen that whereas the frontal and rear parts of the moving objects are fairly accurately detected as being dynamic, on the level of bigger objects, there are also parts that appear longer in the occupancy map, that are detected as being transient or static. Moreover, the static part of the environment where the buildings have a clear vertical structure, the objects are detected as being static, whereas the horizontal surfaces such as pavements or vertical objects with complex structure (buildings with interchanging columns, trees) can present spurious dynamic object detections. This implies that a post-processing step would be required on the level of labelled clusters, namely connect locally adjacent clusters into bigger objects where a voting procedure would be applied, whether the whole object is dynamic or not. The applied clustering procedure represents a good seed for such object voting as well as refining the objects orientation based on, for instance, line segmentation and corners detection.

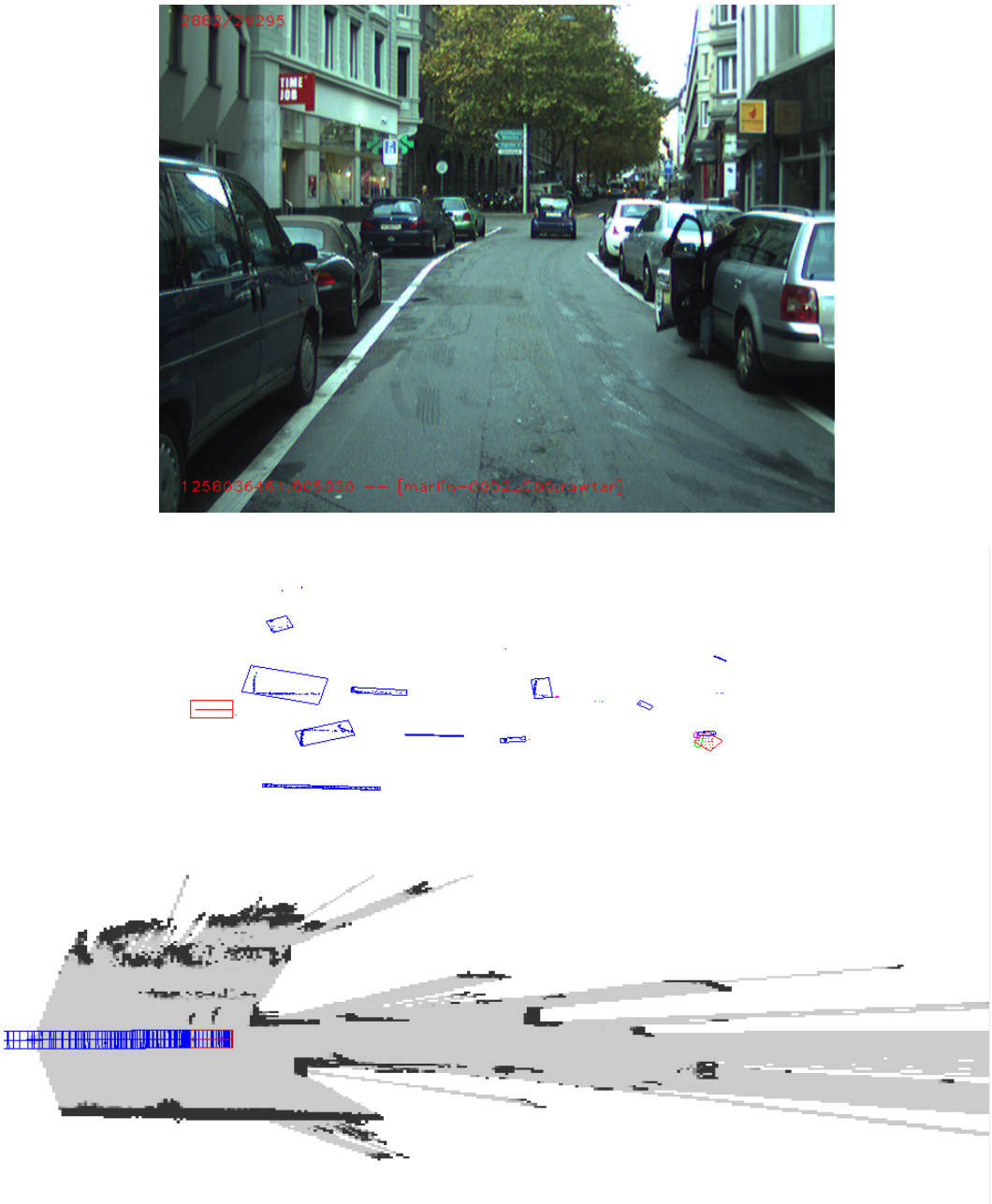


Figure 5.22: Detection of a vehicle moving away I (top: scene view, middle: extracted objects, bottom: occupancy map).

Figure 5.23: Detection of a vehicle moving away II (top: scene view, middle: extracted objects, bottom: occupancy map).

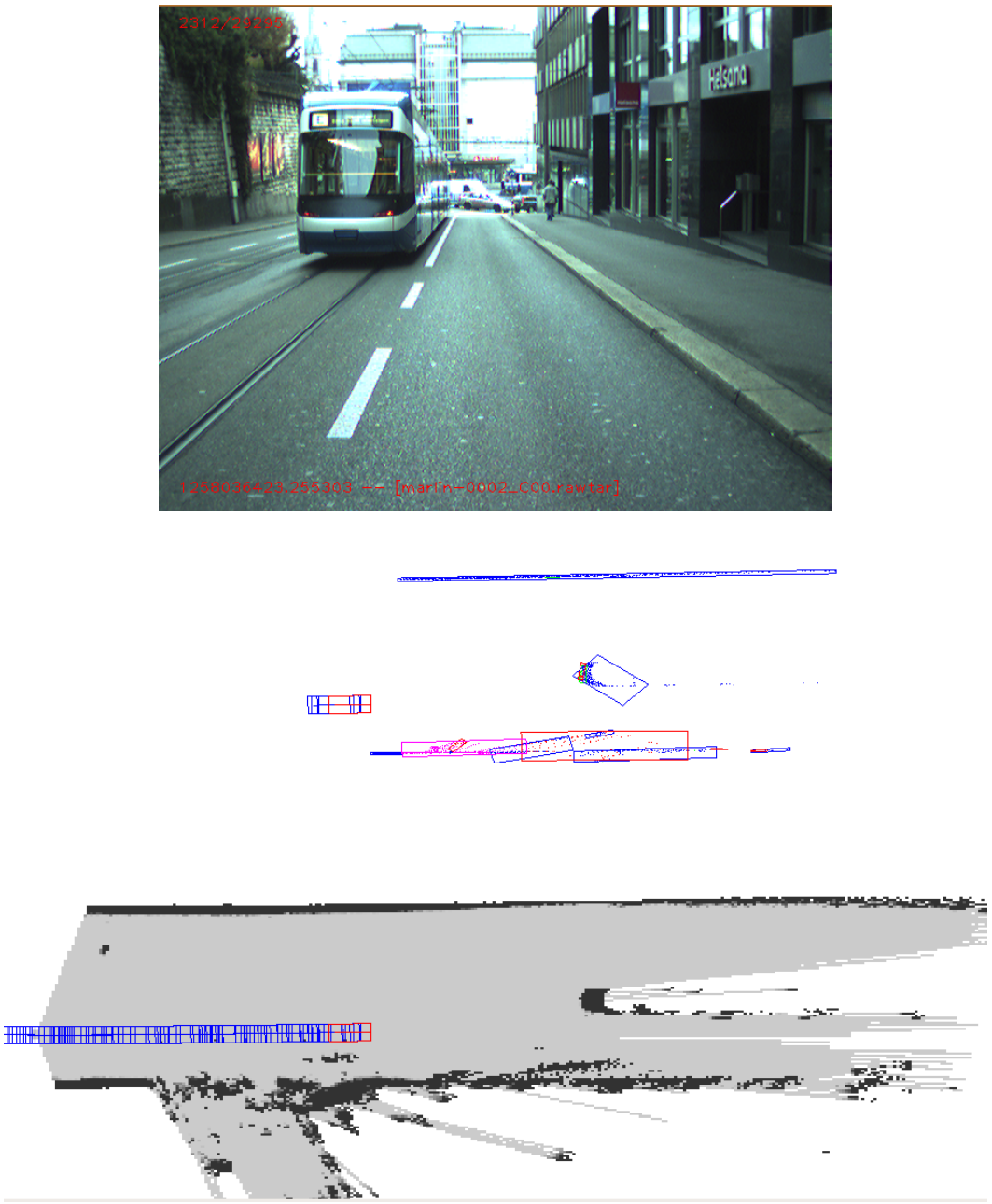


Figure 5.24: Detection of a moving tram and pavement structure I (top: scene view, middle: extracted objects, bottom: occupancy map).

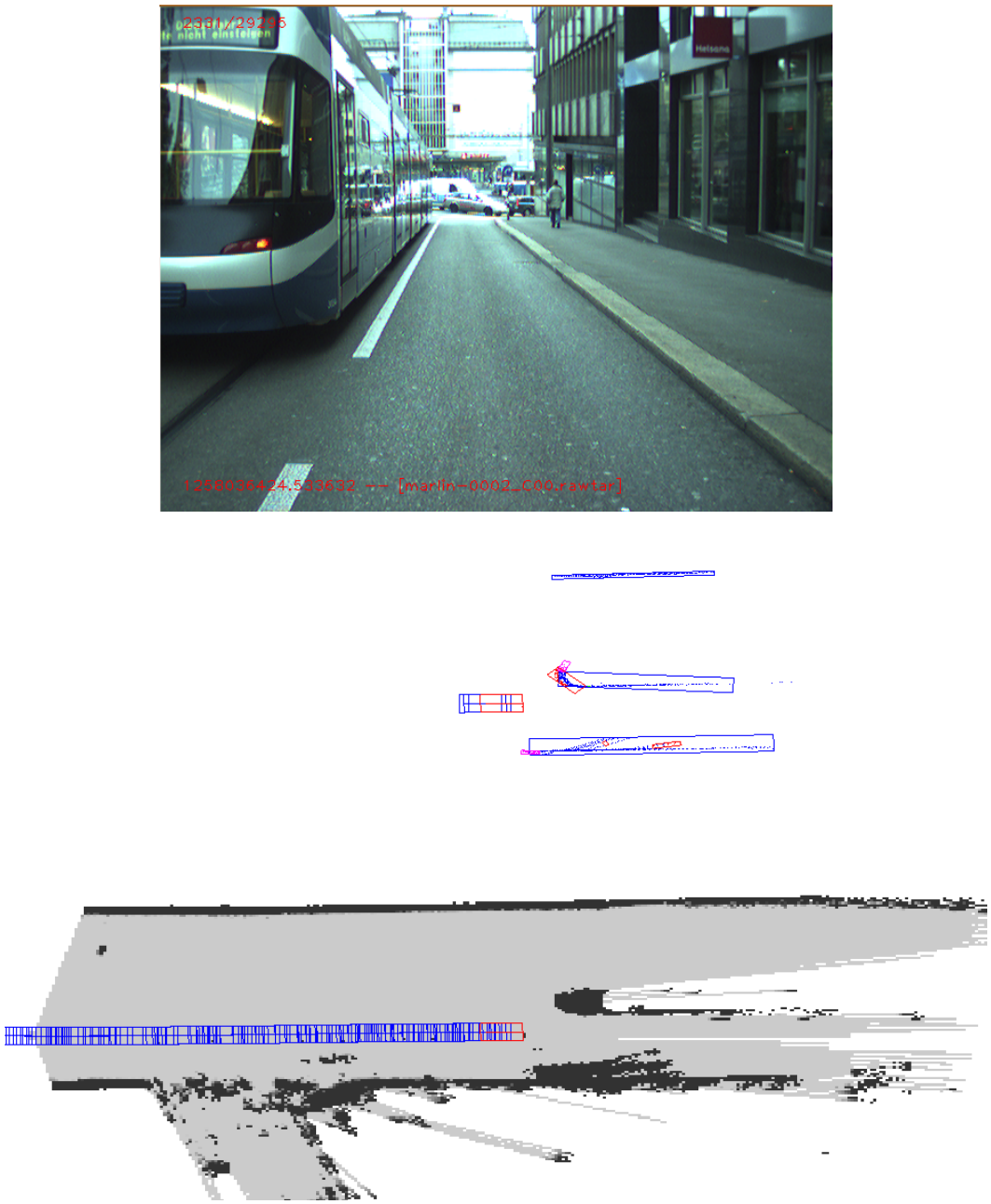


Figure 5.25: Detection of a moving tram and pavement structure II (top: scene view, middle: extracted objects, bottom: occupancy map).

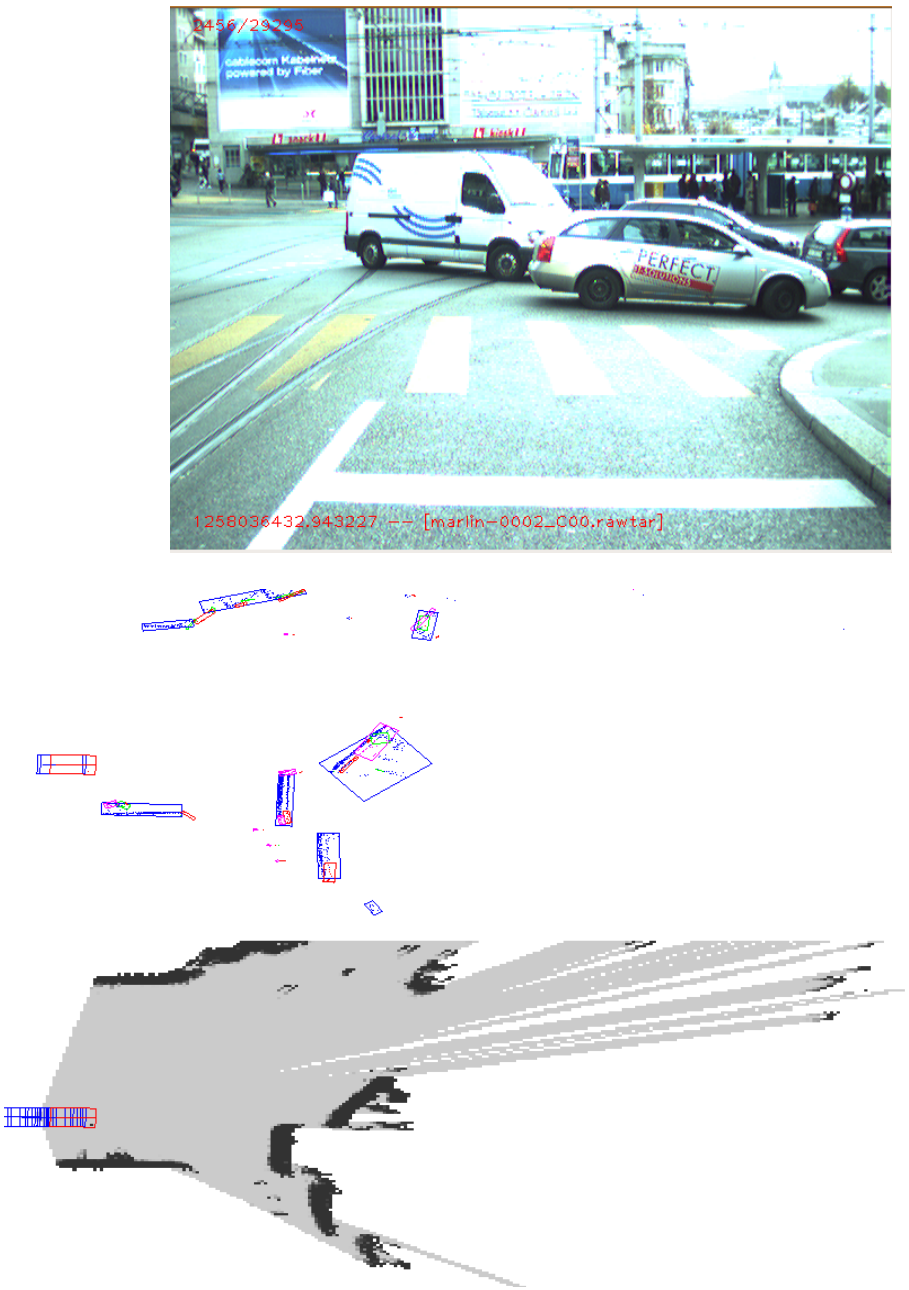


Figure 5.26: Detection of moving vehicles at an intersection I (top: scene view, middle: extracted objects, bottom: occupancy map).

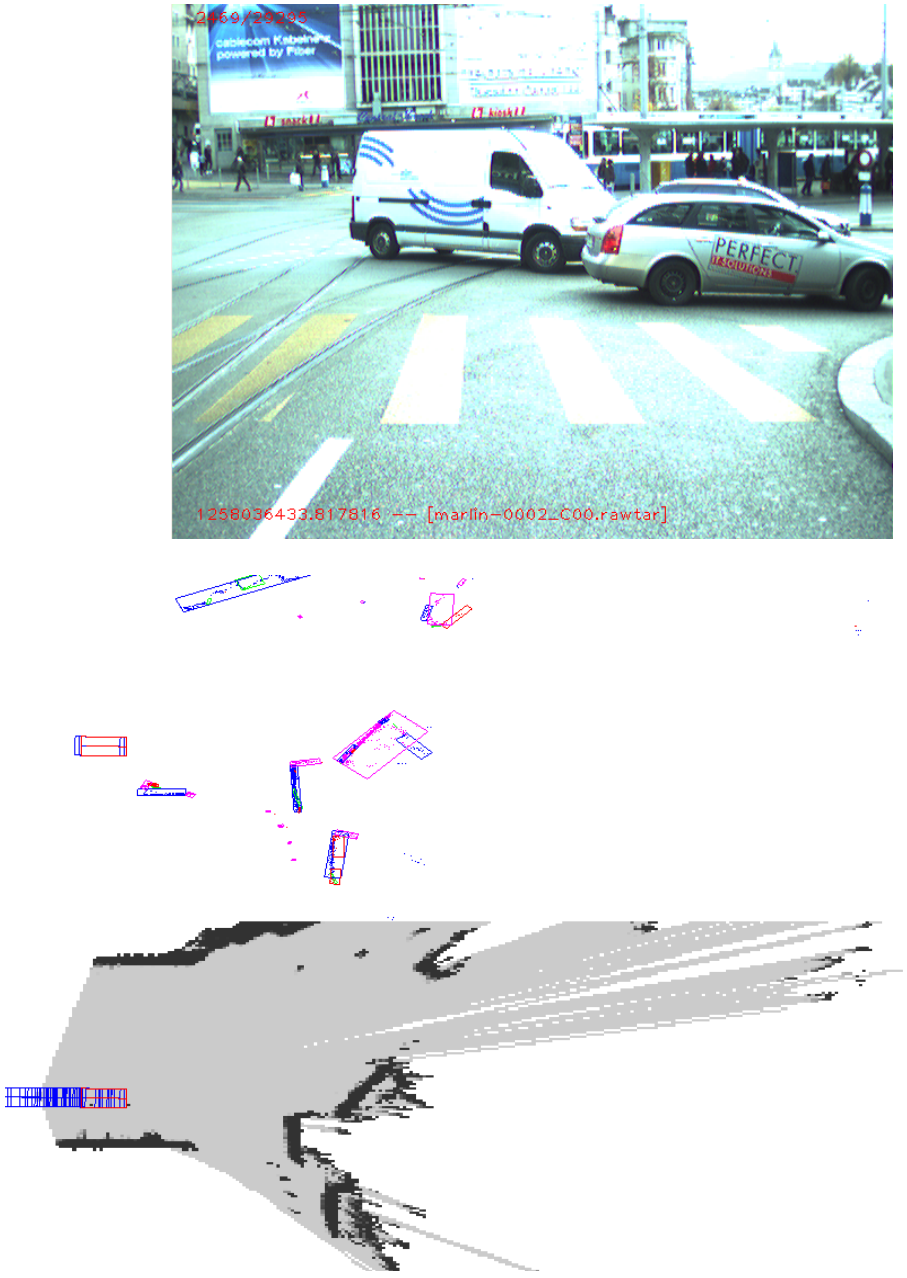


Figure 5.27: Detection of moving vehicles at an intersection II (top: scene view, middle: extracted objects, bottom: occupancy map).

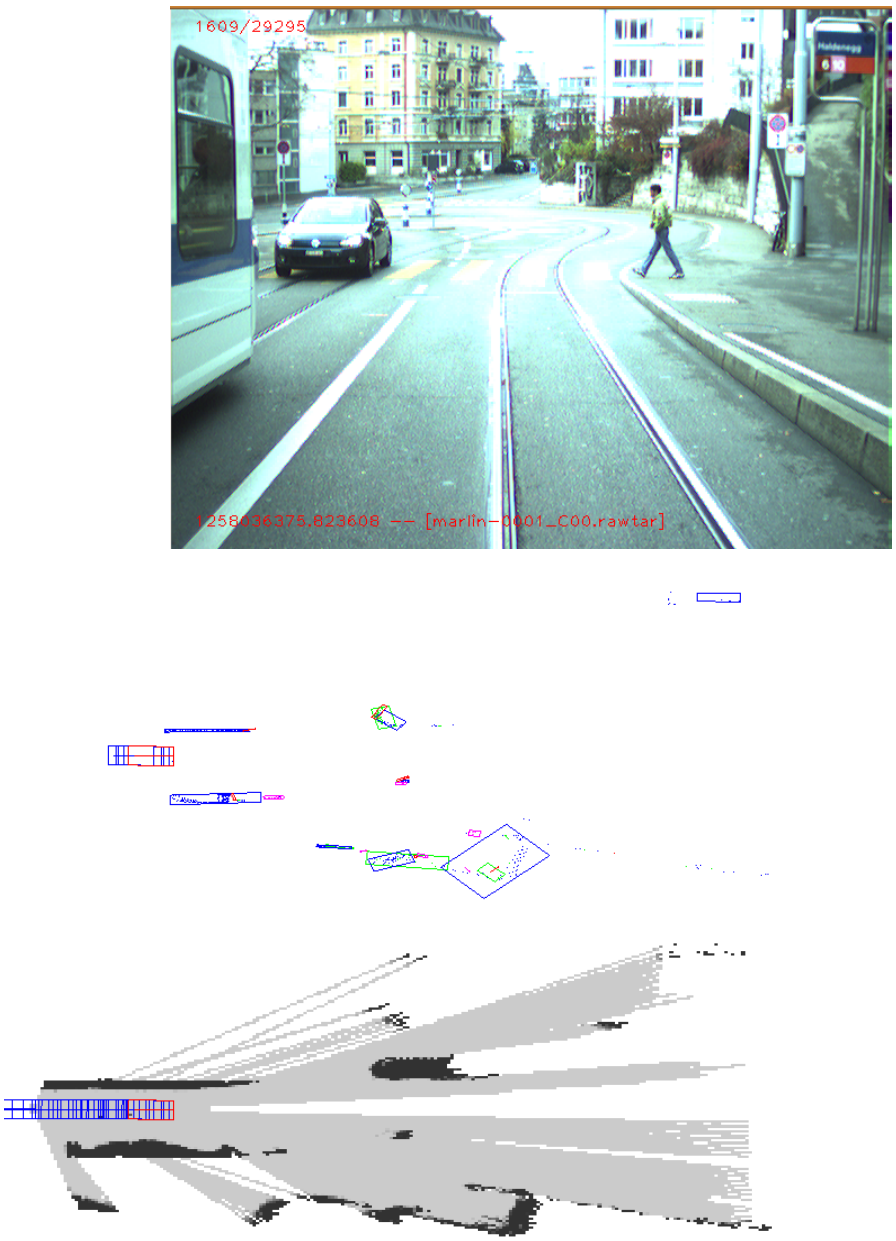


Figure 5.28: Detection of a pedestrian, tram and an approaching vehicle (top: scene view, middle: extracted objects, bottom: occupancy map).

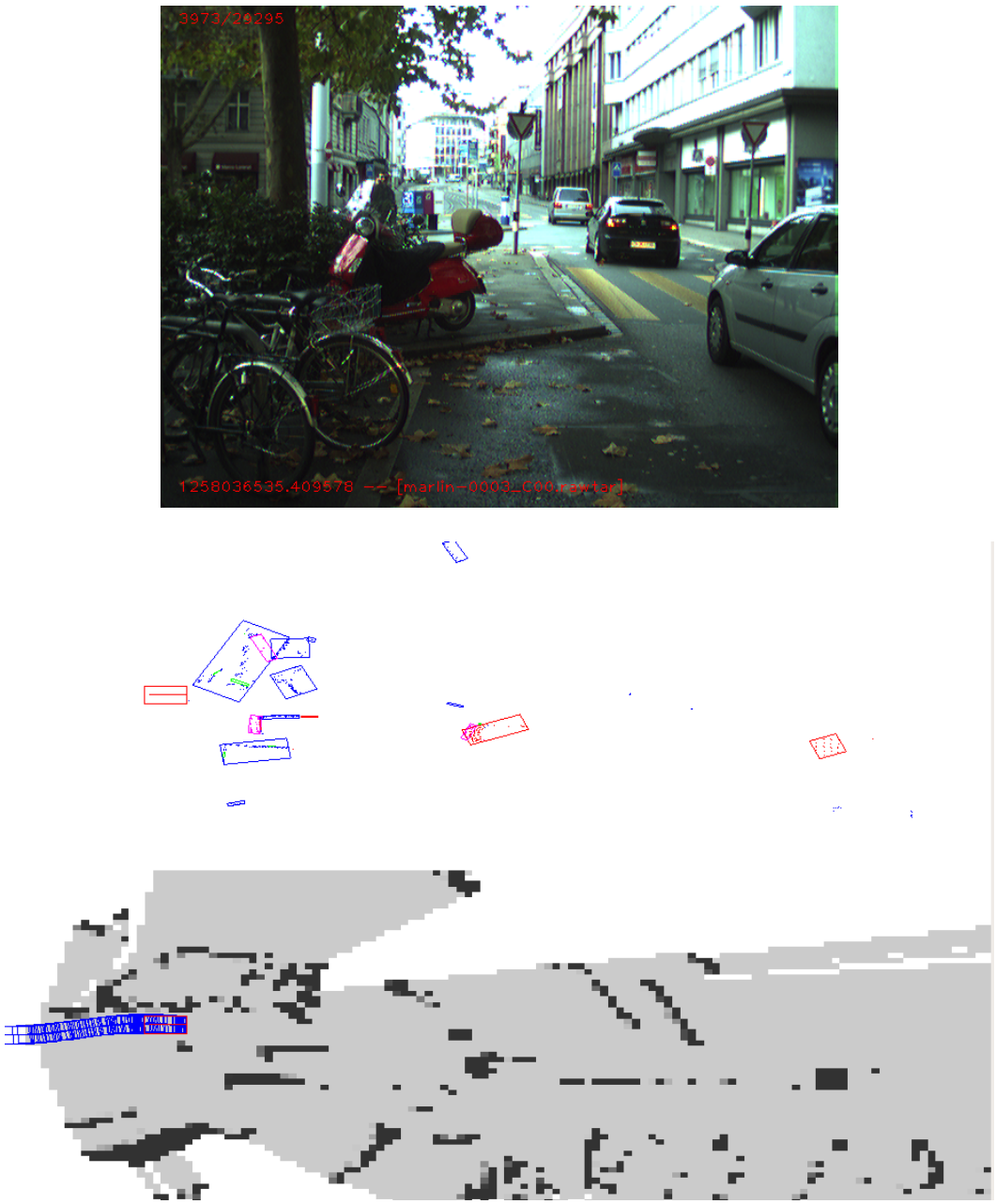


Figure 5.29: Detection of moving vehicles and a pedestrian (low resolution occupancy map) I (top: scene view, middle: extracted objects, bottom: occupancy map).

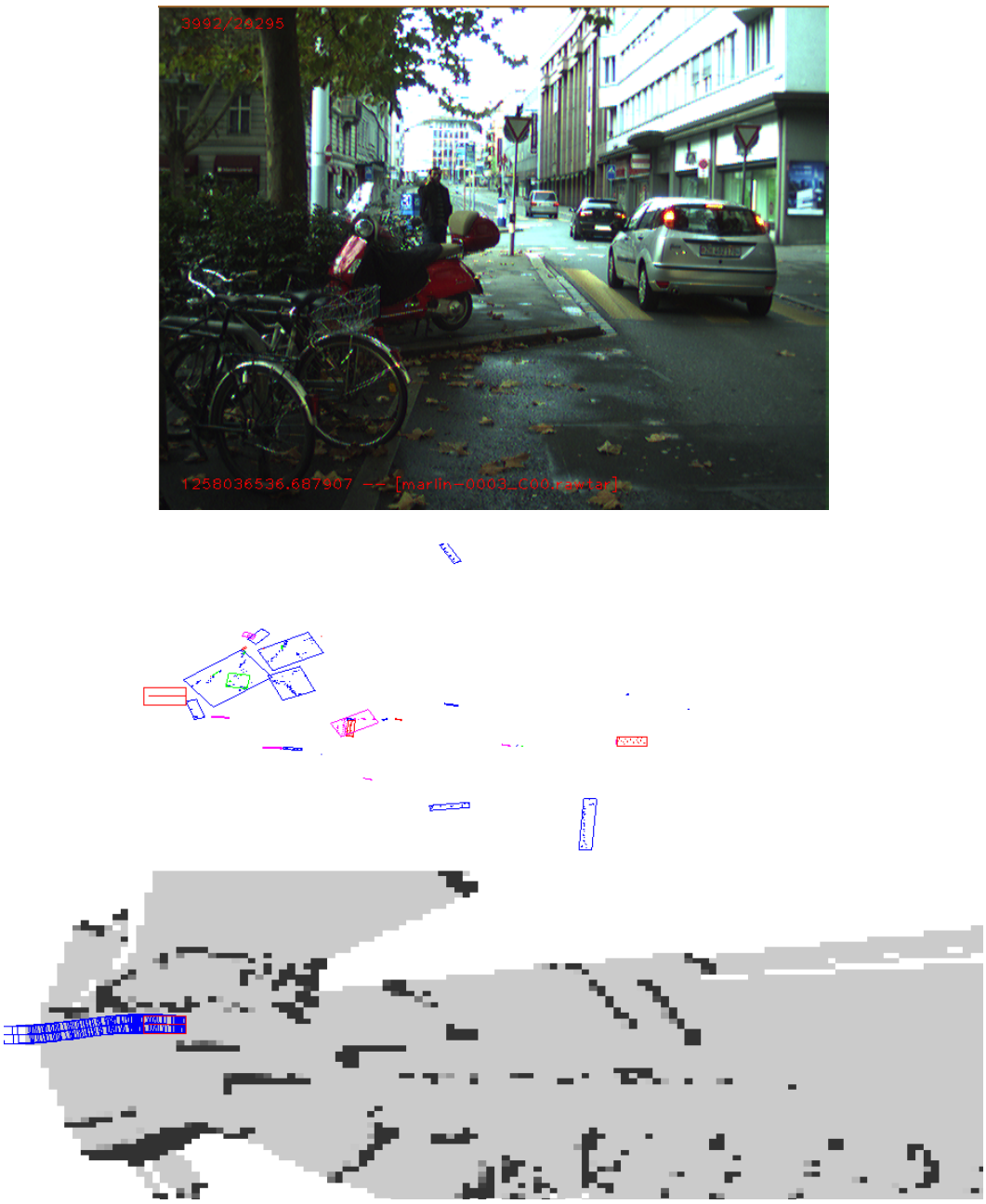


Figure 5.30: Detection of moving vehicles and a pedestrian (low resolution occupancy map) II (top: scene view, middle: extracted objects, bottom: occupancy map).

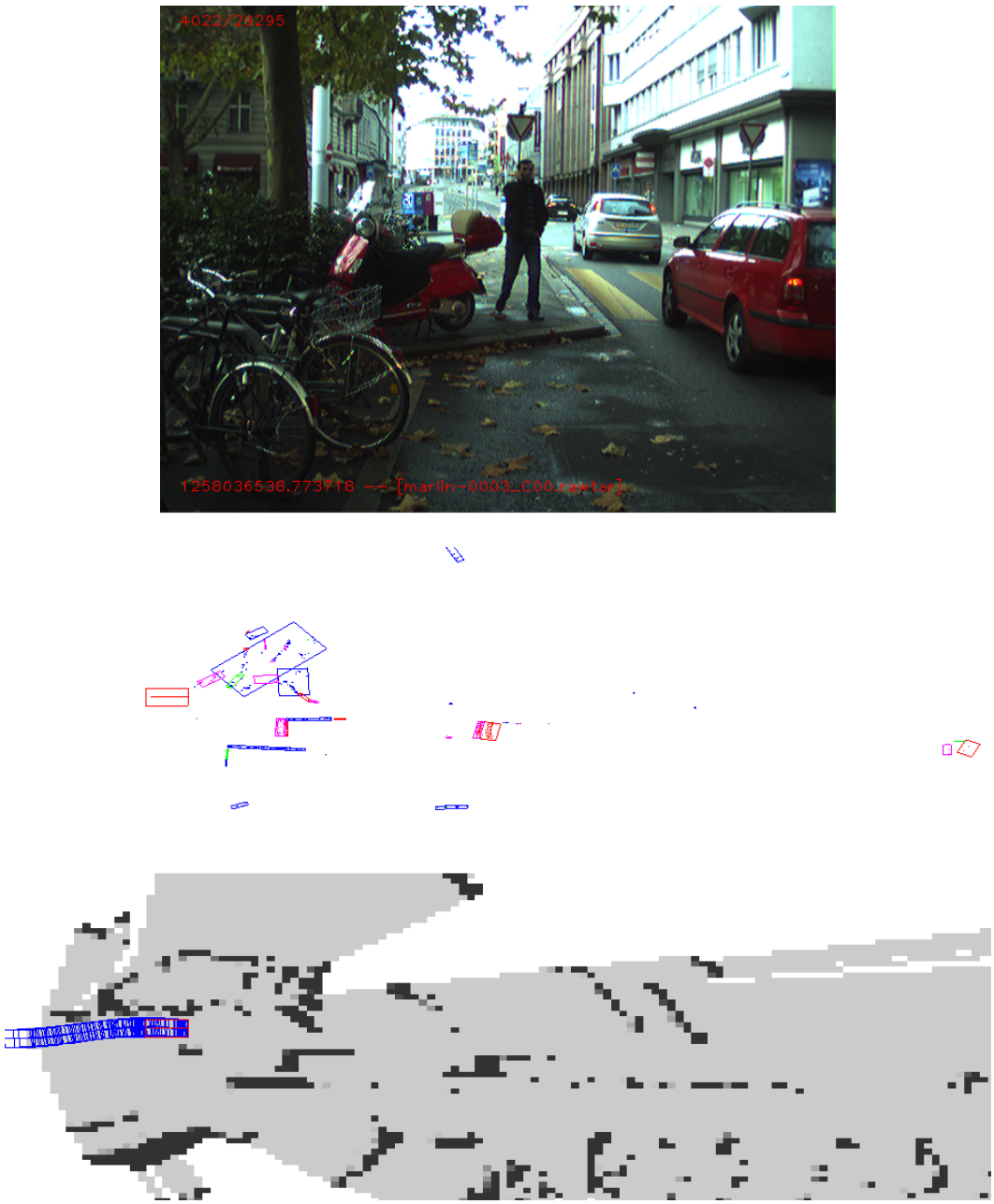


Figure 5.31: Detection of moving vehicles and a pedestrian (low resolution occupancy map) III (top: scene view, middle: extracted objects, bottom: occupancy map).

5.7 Conclusion

In Chap. 5 key elements of the dynamic scene analsis for autonomous vehicle navigation were presented starting with the ego-motion estimation/localization based on GPS, as well as ego-vehicle based inertial data. The initial road structure segmentation, namely the lane detection, was presented based on a vision module. Local vehicle navigation map based on voxelmap representation and a compact 2D occupancy map inferred from the voxelmap was presented. Based on the occupancy grid evidence, the incoming sensory measurements were labelled according to their dynamicity aspect, follow by a clustering procedure in order to obtain a set of dynamic versus static or transient object candidates. The results were experimentally tested in a urban dynamic scene. Although not used on the current clustered object candidates, an object tracking method was also analzed which was used to estimate the motion of dynamic objects, such as vehicles and pedestrians in a simulated dynamic scene.

At the current stage of the development, the lane detection module of Sec. 5.3 is yet to be integrated with the scene map building and object extraction module. A future integration is an interesting prospect. The vision lane detection module could be enhanced by the laser measurements in order to better estimate the road curvature, especially in the presence of pavements. On the other hand, the object detection and extraction module could profit of the lane information in order to filter out spurious dynamic readings labelling on the building structures of the urban scenario as well as to be able to better predict the dynamic object motion based on the lane information, with a strong prior probability that vehicles and trams follow certain routes/lanes whereas the pedestrian motion is more “brownian”, but two distinctive behaviors can still be defined - crossing a lane/road or walking parallel to the road structure on the road side.

Chapter 6

Autonomous navigation in dynamic urban scenarios

6.1 Introduction

Typical urban-like traffic environment consists of a road network structure consisting of lanes and intersections where dynamic obstacles such as vehicles and pedestrians are present. Alternatively, parking lot zones do not posses a clear structure but are bounded only with the perimeter limits. Typical objects involved static or slowly moving vehicles and pedestrians.

The aim of this chapter is to integrate the motion planning technique developed in Sec. 4.4 into a hierarchical navigation structure that can be applied for the urban traffic case. The hierarchy of the navigation structure itself is presented in Sec. 6.2. The overall mission of the vehicle to follow is defined on the road topology as a set of waypoints and represents hierarchically the highest planning level of the navigation scheme. Sec. 6.3 describes how the road network is represented that can be used for global route planning. Sec. 6.4 and Sec. 6.5 analyze simulation results for the lane and intersection case as well as parking lot, respectively.

6.2 Navigation architecture

Fig. 6.1 presents an overview of the navigation architecture for the dynamic urban environment. It is structured in layers where the higher level components interact with all the lower level layers in the hierarchy. The only exception is the world model, which is can be directly accessed by all the levels of hierarchy from static road structure topology to current dynamic obstacles information based on the requirements.

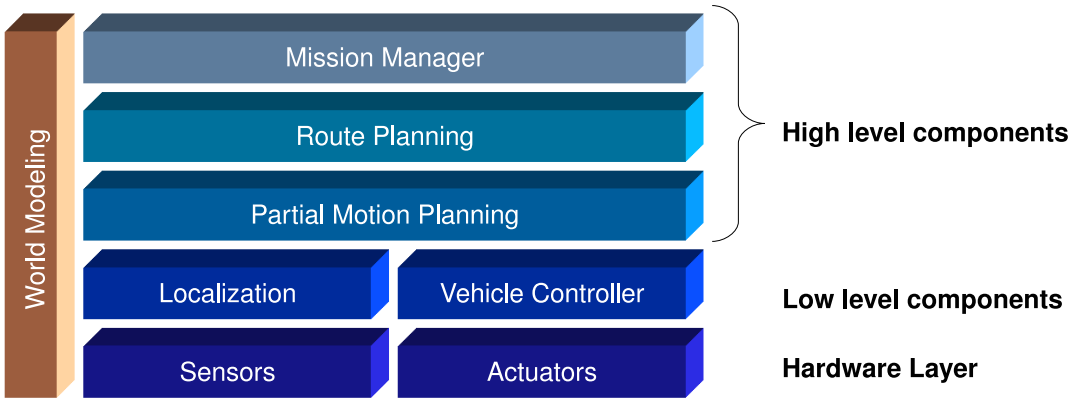


Figure 6.1: Overall components scheme of the navigation architecture for dynamic urban environments.

1. *Mission Manager*: it is responsible of translating high-level tasks (*eg* pick-up a person at an address) into initial and a goal configurations that the vehicle should reach in order to accomplish those tasks.

2. *Global Route Planning* (*cf* Sec. 6.3): it is concerned with finding a global route for the vehicle between the initial and goal configurations given by the mission manager, using information about the static characteristics of the environment (e.g. lane geometry, speed limits). It is assumed that the knowledge about these characteristics is available *a priori* so that it is possible to perform at least part of the computations off-line.

3. *Partial Motion Planning* (*cf* Sec.4.4.2): it is responsible of trajectory planning level of the overall mission up to the next waypoint defined by Global Route Planning. It integrates knowledge about the dynamic elements of the environment and interacts tightly with the world model.

4. *World model* (*cf* Chap. 5): the world model gathers all the available information about the environment, including the vehicle's localization, the environment structure and the current and predicted states of the other objects that are present in the environment. As mentioned above, it is different from other modules because it interacts with all the levels of the hierarchy.

Note that the *Vehicle Controller* module in the sense of the hierarchy presented here includes not only the low-level steering and gas pedal controllers but also the intermediate trajectory controller that is provided by the *Partial Motion Planning* for the duration of execution cycle T_e according to Sec. 4.2 on time constraints, where the decision cycle of the planner T_d equals $T_d = T_e$. Moreover, the trajectory controller executes the reference trajectory motion in an open-loop fashion, assuming that the dynamic limits such as maximum linear acceleration and steering rate are taking into account in the trajectory generation. A closed loop reference trajectory tracking is also possible, but has to be modelled accordingly with the low-level control loop included in the trajectory diffusion phase (*cf* Sec. 4.4.3).

6.3 World model and global route planning

6.3.1 World model

Fig. 6.2 depicts an urban traffic scene that is used to validate the proposed hierarchical navigation scheme. It contains essential elements of road structure based on lanes and intersections as well as a parking area (the zone in the lower right corner of the scene). The simulated dynamic objects are vehicles and pedestrians that

follow certain pre-recorded routes or can move in the scene in a randomized fashion. The simulator engine also includes different laser and camera sensor models as well as physical body dynamics and contact modeling as options.

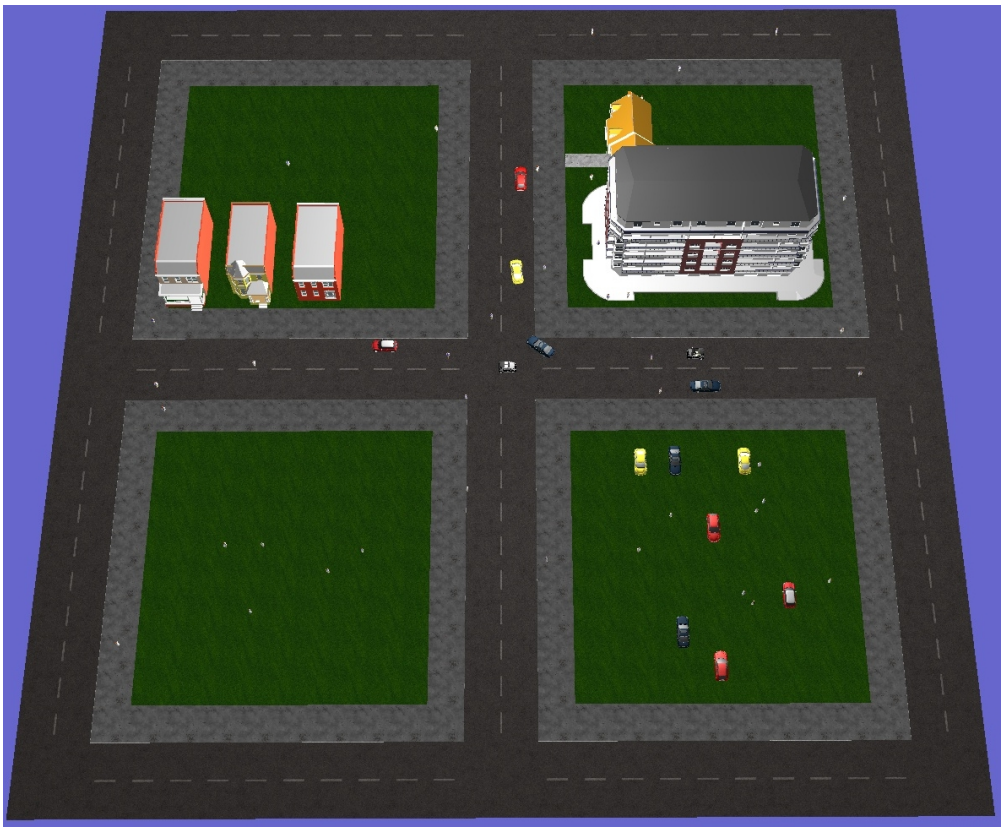


Figure 6.2: A simulated urban traffic scene.

The prior knowledge about the road structure is described via a Route Network Definition File (RNDF) [rnd, 2007] (*cf* Appendix A for the listing of RNDF describing the scene in Fig. 6.2). The RNDF description represents essentially a topological division of the environment into road segments that contain lanes in both driving directions and zones that contain perimeters (arbitrarily structured areas, such as parking lots). Type of lanes which can be *broken_white*, *solid_white*, *solid_yellow*, *double_yellow* are also contained in the RNDF. The collision checking module of the Partial Motion Planner treats the *broken_white* as the absence of static obstacles, the other cases are implemented as polyline obstacles.

Fig. 6.3 depicts a possible lane division to the scene of Fig. 6.2 where all the different types of lanes are present. The current state and future motion prediction of all the potentially dynamic obstacles are estimated based on object's current position and the obstacle tracking module presented in Sec. 5.5.3 as can be seen from object traces in Fig. 6.3.

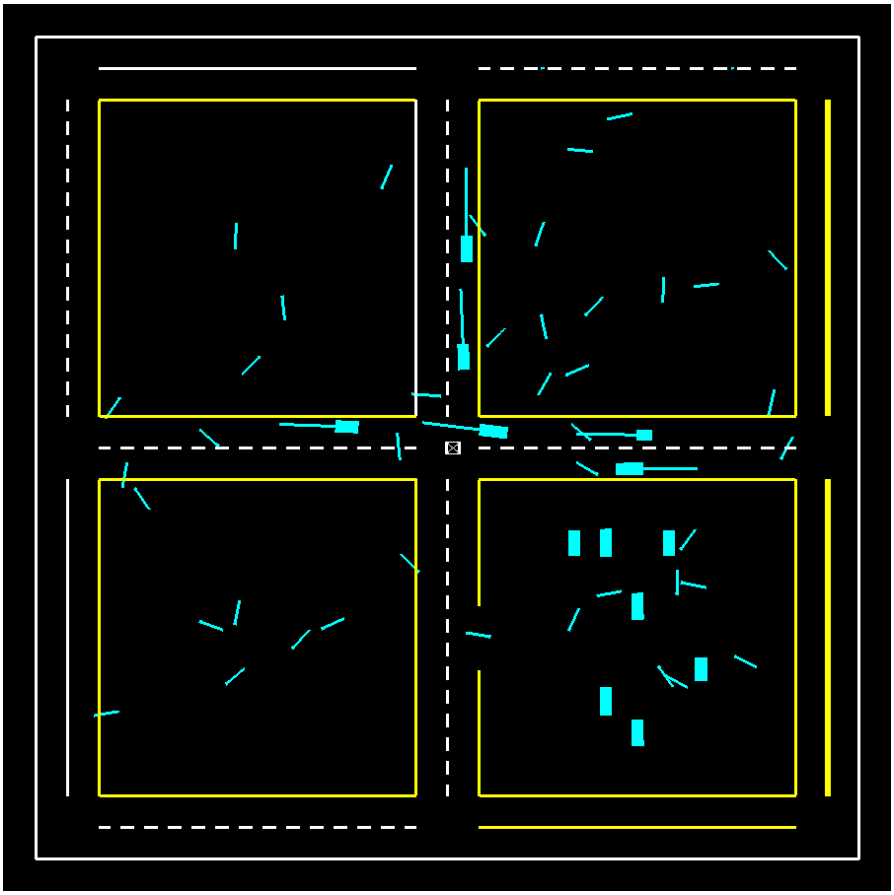


Figure 6.3: Lane structure of the simulation environment based on RNDF topological description and the lane definition file containing the metric information about the lanes. The lane types include *broken_white*, *solid_white*, *solid_yellow*, *double_yellow*. On the collision checking level each lane is described as a static polyline obstacle. The present dynamic objects such as vehicles and pedestrians and their corresponding future predicted motion trajectories are depicted as well (cyan).

In addition to the topological lane description and lane width defined in the RNDF, also the geometric information is needed to define the static part of the environment obstacles. In a real experimental vehicle this information may be

available via a lane detection module, such as the one described in the Sec. 5.3, a GPS navigation system unit or by using an additional lane definition file based on global GPS coordinates, an example of which is depicted in Fig. 6.4. Intersections are not explicitly described in the RNDF form, since they are implicitly defined by the adjacent road segments.

Figure 6.4: A section of the lane definition map based on global GPS coordinates. The L labelled points define the outer lane boundaries, whereas the M labelled points define the middle of the road. Other GPS points are the waypoints defined by the RNDF description.

Within each lane or perimeter of the RNDF description, there is a set of waypoints that contain position information. Additionally, a waypoint can be defined as stop or exit point, which provides the connectivity information about allowed transitions between lanes at intersection points. A subset of waypoints, named checkpoints, are waypoints with an id number that can be used to define the a global level navigation mission scenario. Fig. 6.5 represents full waypoints connectivity map based on the RNDF description of the traffic scene in Fig. 6.2.

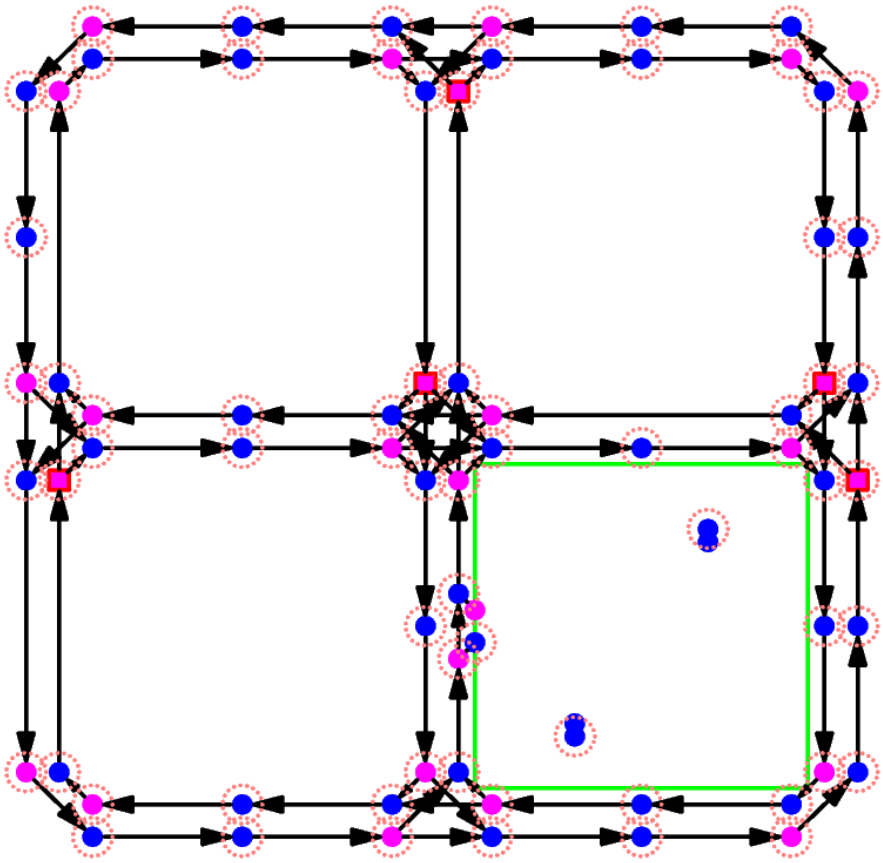


Figure 6.5: Route Network Definition File (RNDF) describes topological connectivity in a road structure. Basic waypoints (blue), checkpoints (circle dashed), exit points (magenta) and stop points (red squared) all describe metric distances as well as transitions in lanes and intersection. Zones contain perimeters which are described by polygons (green). A perimeter can among other contain a parking lot with parking spots defined by checkpoints.

6.3.2 Global Route Planning

The global mission task is defined by the *Mission Manager* is as a list of checkpoints, i.e. a list of waypoints from the RNDF that have to be traversed in order to complete a mission (typical scenario in DARPA vehicle challenges, for instance). The list can be only one goal checkpoint or a list of many. In either case, the *Global Route Planning* module uses the RNDF connectivity map presented in Fig. 6.5 in order augment the global route plan with the set of available intermediate waypoints. The route planning is straightforward by applying the A* graph search

algorithm using the length of the graph's edges as a cost function in order to find the shortest route between two checkpoints (an alternative cost function might be the fastest route between the checkpoints). Each checkpoint defined by the mission task contains specific speed limitations also, or the general traffic speed limits in the urban area apply. The stop rule for a particular set of waypoints of the RNDF file cannot be overruled. From the RNDF and the lane definition file also the orientation of each waypoint within the lane structure can be derived.

The output of the *Global Route Planning* module consists therefore of a list of waypoint configurations $\mathcal{Q}_g = \{q_g^1, \cdots, q_g^{N_g}\}$ that the vehicle should reach. Each configuration can be described as a position and orientation:

$$q_g^i = \{x_g^i, y_g^i, \theta_g^i\} \tag{6.1}$$

Associated with every configuration, there is a constraint vector describing the bounds within which the vehicle's motion is considered as acceptable with respect to the task at hand and the traffic rules:

$$c^i = \{r^i_{\max}, \Delta\theta^i_{\max}, v^i_{\min}, v^i_{des}, v^i_{\max}\} \tag{6.2}$$

where $r^i_{\max}$ stands for the maximum distance between the object's actual position and the desired one; $\Delta\theta^i_{\max}$ is the maximum error tolerated for the heading; and v^i_{min}, v^i_{des} and $v^i_{\max}$ are the minimum, desired and maximum velocities associated to the given configuration.

The waypoint configurations described by the Eq. 6.1 and the related constraints defined by Eq. 6.2 can be fed directly to the *Partial Motion Planning* level that computes the current feasible trajectory for the vehicle based on ego-motion trajectory exploration and collision checking with the static and dynamic objects in the environment.

6.4 In lane and intersection navigation

Simulation results in this section are presented for different structured urban environment scenarios, i.e. in lane driving and intersection handling.

The dynamic obstacles presented in different scene views are pedestrians and static and moving vehicles, whose shapes are described as polygons, more specifically rectangular shapes. Their estimated positions and future trajectories are depicted in navigation views which also includes the static part of the environment that is described via polylines (the lane boundaries). Collision checking between the ego-vehicle and environmental obstacles has to be performed for the whole given time horizon. The navigation scheme presented in the results is based on waypoint following (*cf* Sec. 4.4.3, case 1) and exhaustive trajectory exploration phase. The currently active goal waypoint is drawn in the navigation views as a circular region based on the given constraints of Eq. 6.1 and 6.2. The trajectory diffusion starting from the ego-vehicle contains tree nodes that are free of obstacles (green) and prohibited nodes (marked red), whereas the currently best trajectory towards the waypoint is marked in magenta. Note that the given navigation snapshots are based on a certain time instant $t_k$and that the prohibited regions depend on all the predicted future motions of the obstacles.

Fig. 6.6 and 6.7 depict a situation where the vehicle is driving in a lane behind another vehicle. The space currently occupied by the leading vehicle is going to be freed within a certain time in the future, therefore there exists a collision free trajectory for the ego-vehicle to follow. In Fig. 6.8 and Fig. 6.9 the vehicle avoids collision with an pedestrian in the lane. Due to the control input discretization and confined maneuver space, there is just enough space to avoid the object otherwise the vehicle would have to stop according to the braking maneuver capability. It is also discutable if in practice passing this close to an object such as pedestrian would be desired. To avoid this behavior, an enlarged comfort buffer zone around the pedestrian's actual size could be introduced in the collision checking stage.

Fig. 6.10 and 6.11 represent negotiating an intersection crossing in the presence

of several moving objects. The orientation of the next waypoint (blue circle in the upper part of the Fig. 6.11) is deliberately set off the nominal lane orientation in order to show that the partial motion planning scheme is to a certain extend robust with respect to the waypoint configuration if the imposed constraints on $\Delta\theta_{\max}$ are not too tight. In fact, in the case of in-lane driving the structure of the environment itself forces the motion trajectory to be stay oriented with respect to the lane boundaries.

Fig. 6.12 and 6.13 represent another intersection negotiation scenario.where the vehicle successfully avoids dynamic and static obstacles. However, this particular case also shows the limitations of the current hierarchical navigation scheme. The *Partial Motion Planning* level due to the execution/planning and obstacle prediction time constraints has to provide a feasible trajectory at least of duration $T_d = T_e = T_s$, according to Sec. 4.2. The currently active waypoint was "permanently" occupied by pedestrians. Therefore, the next best trajectory that is at computed at least to the T_d decision horizon and has the best end node metric with respect to the Continuous Curvature Path length (*cf* 4.4.3) is actually leading to another lane. This detour is obviously undesirable but in principle cannot be resolved on the *Partial Motion Planning* level.

This implies that there is a need to introduce a *Behavioral Planning* level as an intermediate level between the *Global Route Planning* which provides the list of waypoints to follow and the *Partial Motion Planning* trajectory level. This module would assure that the search space for the *Partial Motion Planning* level would not only be driven by the currently active waypoint but also by additional constraints on the available free space. In the context of the presented scenario, the lane entries straight and left with respect to the vehicle would be blocked with polyline boundaries that would leave the maneuver space of the vehicle only to the intersection area and the right turn lane. This aspect is left as future work.

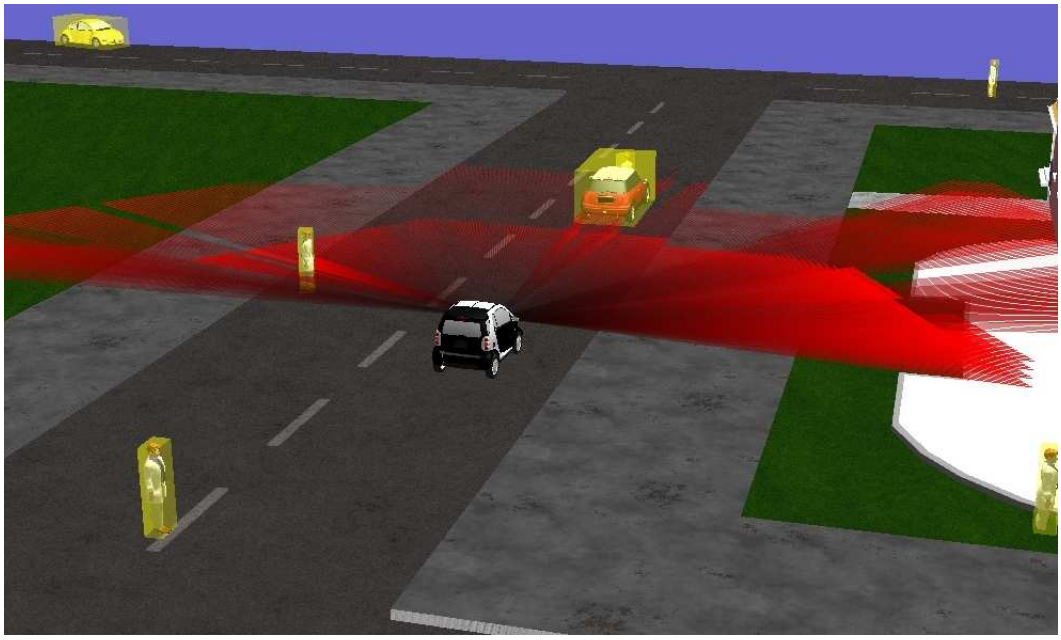


Figure 6.6: In lane (scene view).

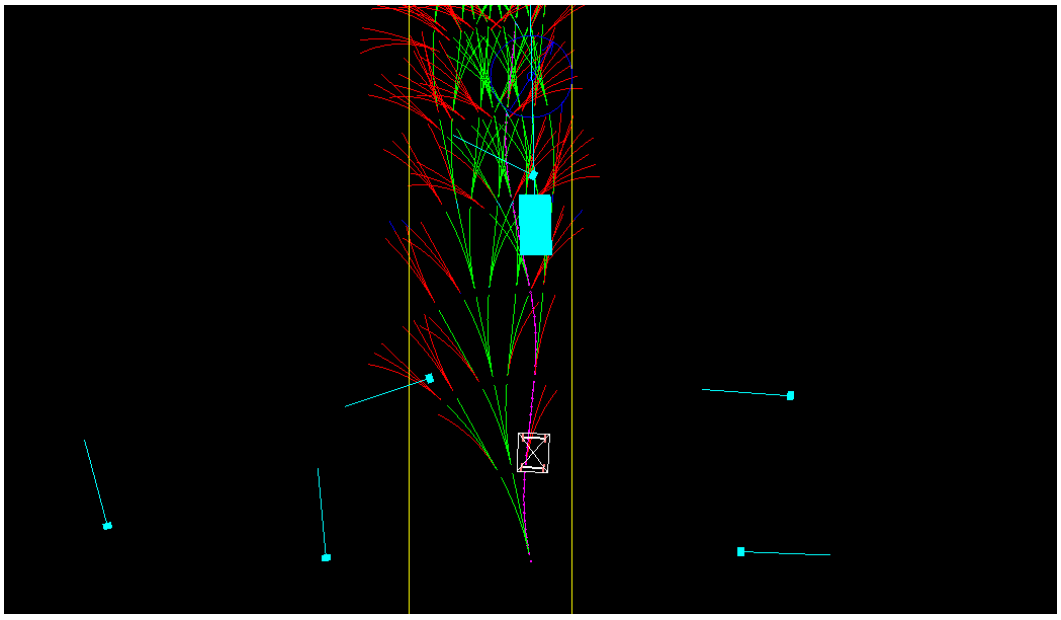


Figure 6.7: In lane (navigation view).

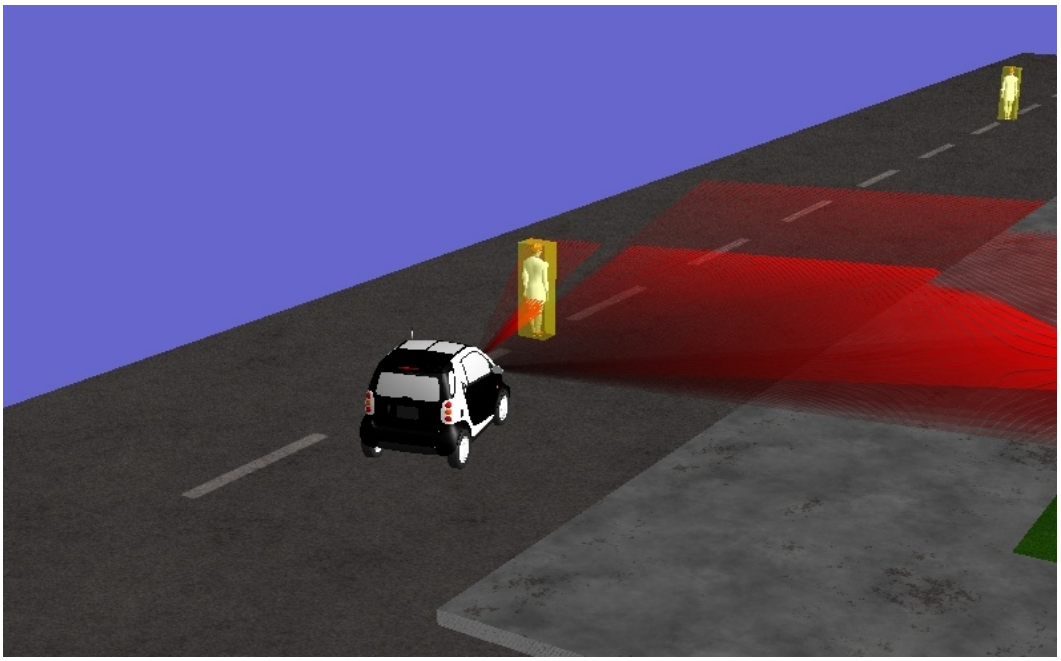


Figure 6.8: In lane pedestrian negotiation (scene view).

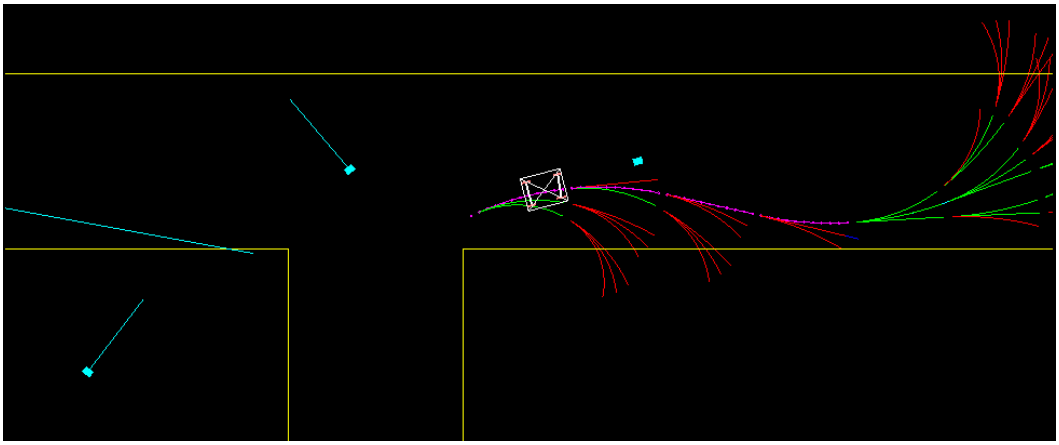


Figure 6.9: In lane pedestrian negotiation (navigation view).

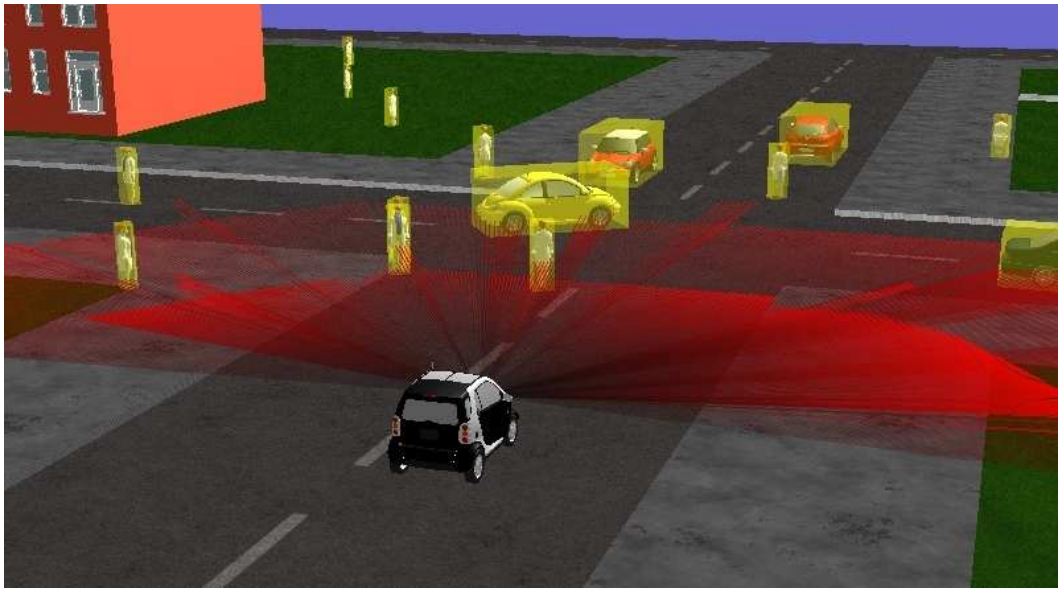


Figure 6.10: Intersection handling I (scene view).

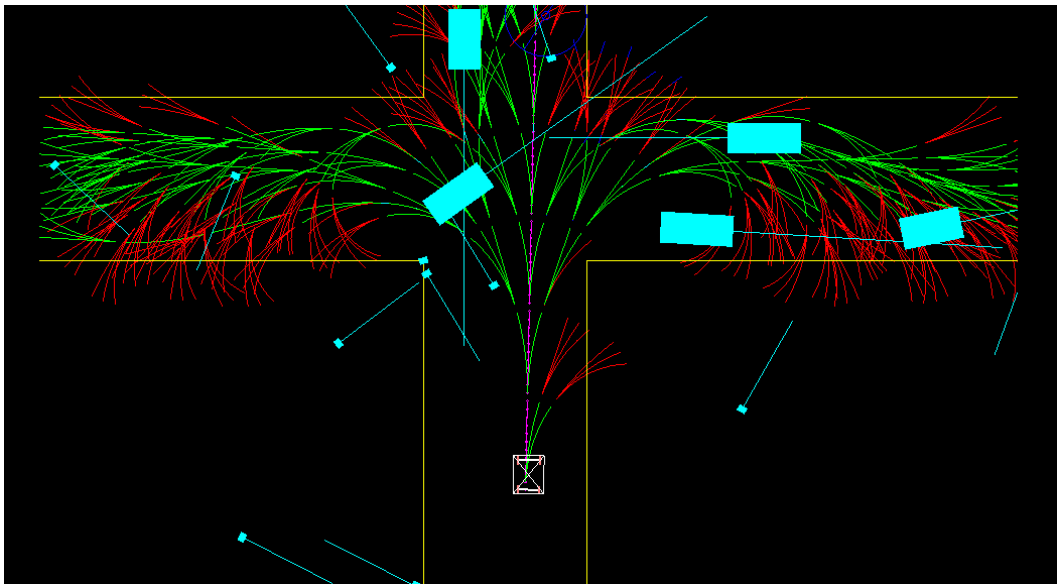


Figure 6.11: Intersection handling I (navigation view).

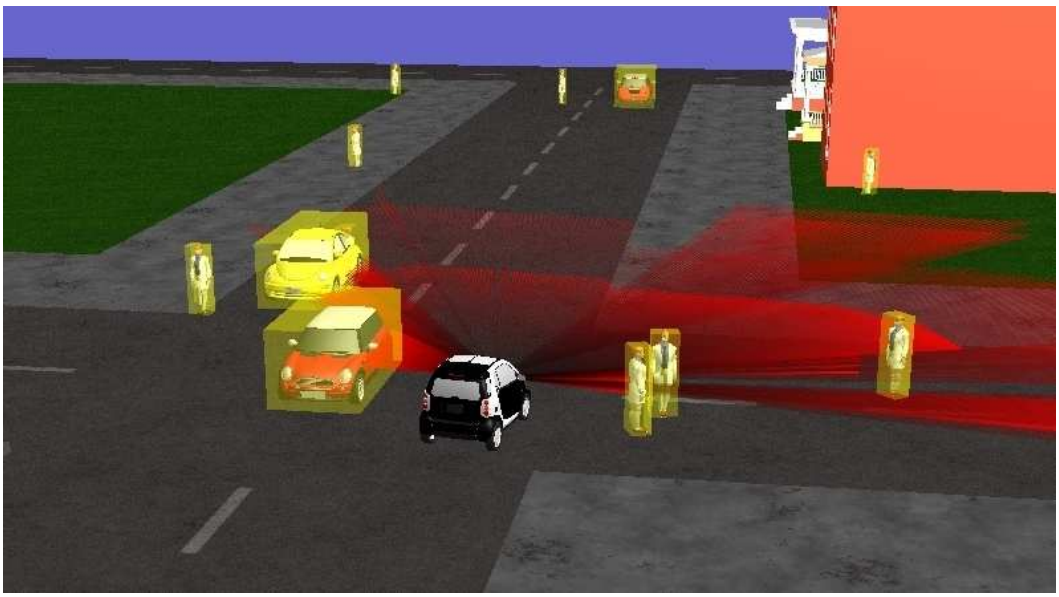


Figure 6.12: Intersection handling II (scene view).

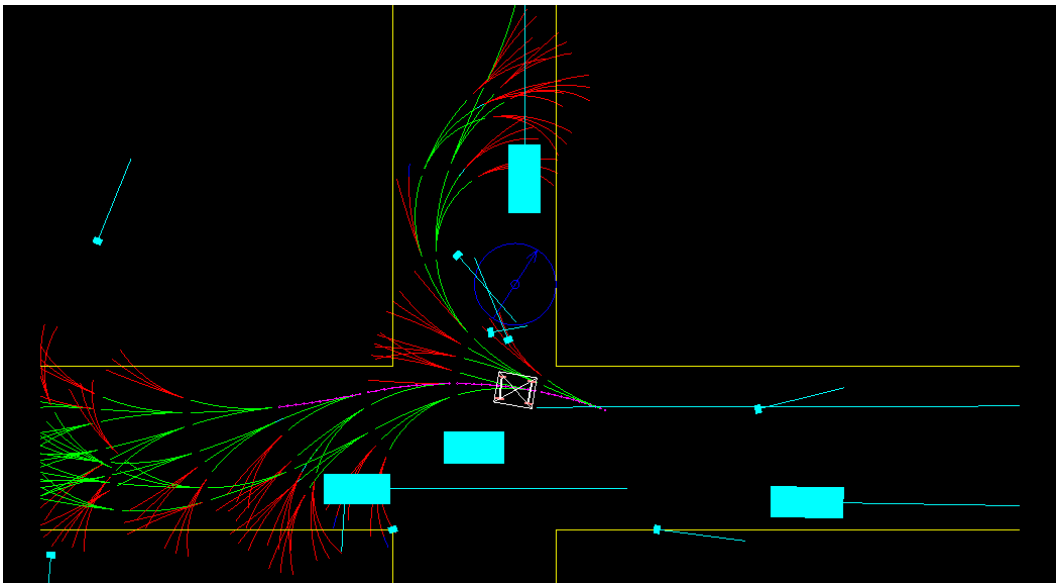


Figure 6.13: Intersection handling II (navigation view).

6.5 Unstructured parking zone navigation

The unstructured parking zone navigation features the similar *Partial Motion Planning* paradigm as the in-lane and intersection navigation, with the difference a combined greedy goal search and randomized control input are used on the level of tree node expansion methods (*cf* 4.4.3), except for the root node, in order to explore the vast free space available. It can be seen from the results that the ego-vehicle is able to reach the active waypoint while negotiating moving obstacles and preventing head-on collisions, as can be seen in Fig. 6.14 to Fig. 6.17, proving the generic applicability of the *Partial Motion Planning* scheme.

However, in order to explore the vast free space available more efficiently, a comparison would be needed between the randomized exploration scheme and a metric based on introducing a global navigation function metric (*cf* Sec. 4.4.3), such as that of the dynamic programming goal cost function in [Montemerlo *et al.*, 2008b].

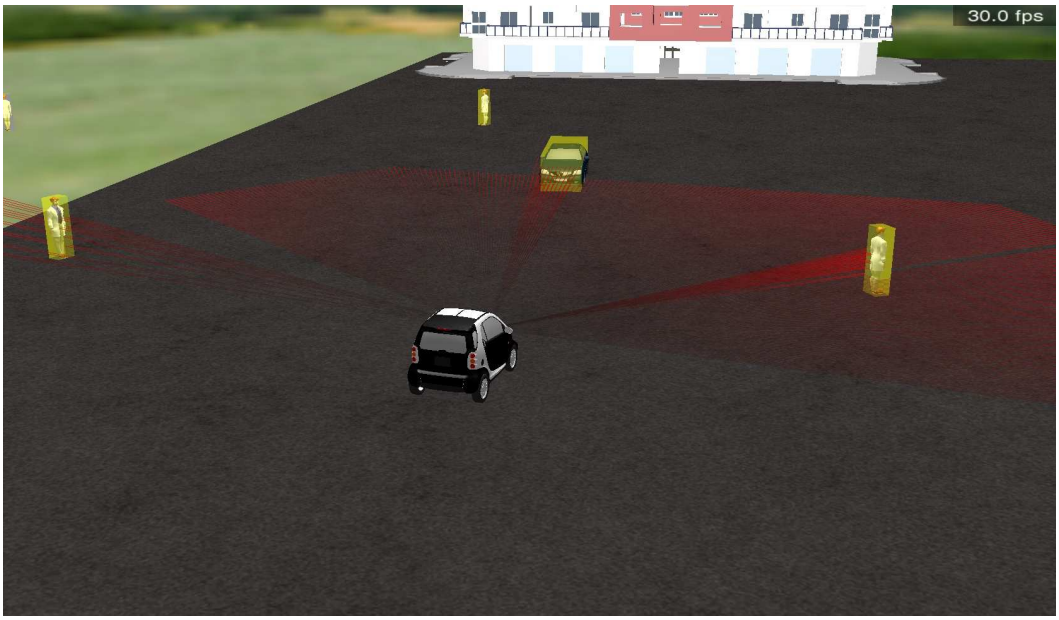


Figure 6.14: Parking lot I (scene view).

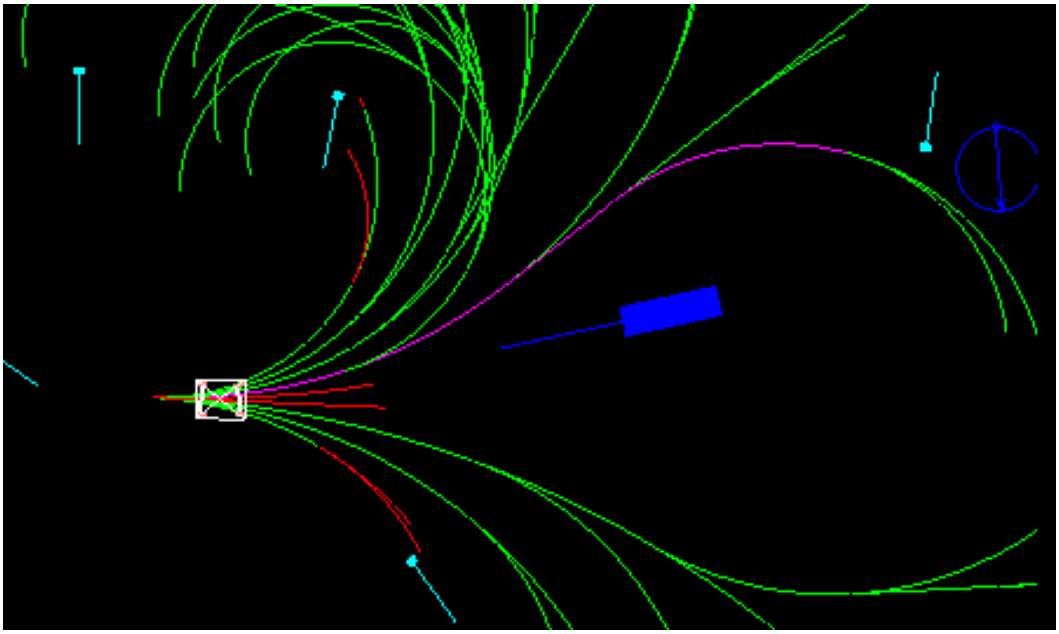


Figure 6.15: Parking lot I (navigation view).

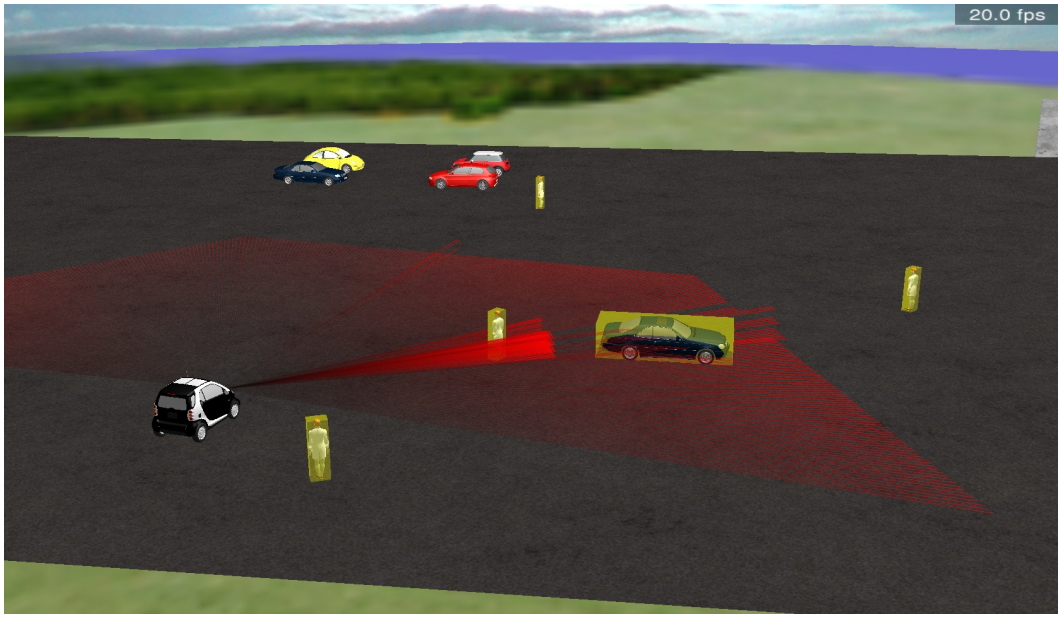


Figure 6.16: Parking lot II (scene view).

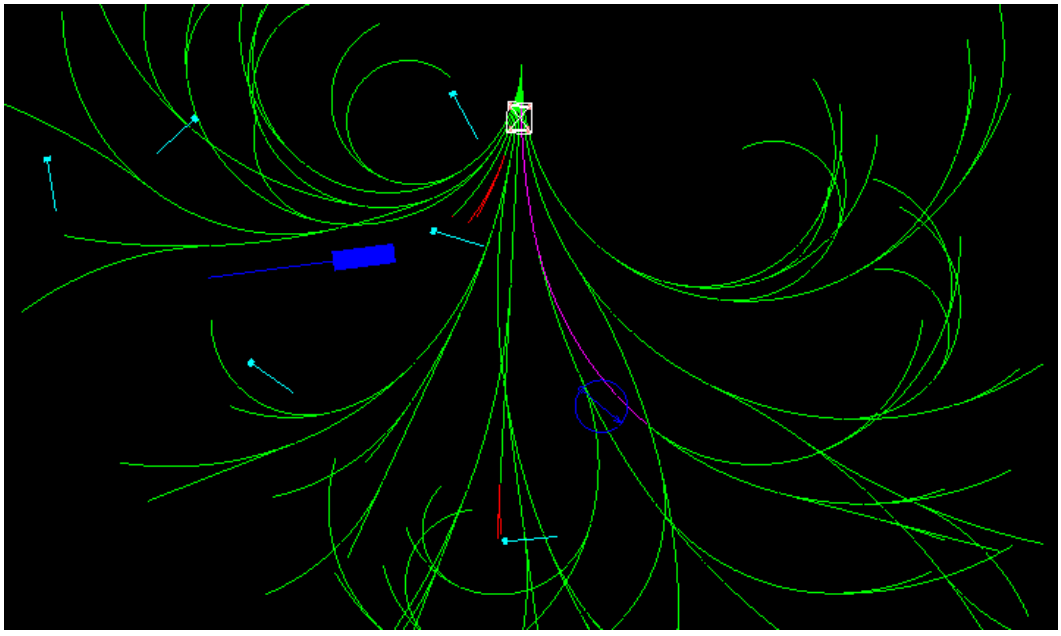


Figure 6.17: Parking lot II (navigation view).

6.6 Conclusion

In this chapter a hierarchical approach to autonomous navigation in dynamic urban scenarios is discussed. It includes at the global navigation level a *Mission Manager* defining the global navigation task that is used by the *Global Route Planner* in order to compute a set of waypoints to be visited by the vehicle. The current trajectory computation and collision avoidance is done at the level of *Partial Motion Planner*. The *World Model* includes the relevant aspects of the environment, such as the road network structure description and dynamic scene analysis. Simulated autonomous navigation results in lane, intersection and parking zone scenes where the ego-vehicle successfully negotiates the dynamic obstacles while navigating towards the next active waypoint. A limitation of the current hierarchical navigation structure was also identified, namely the lack of *Behavioral Planner* that would act as a state machine between the *Global Route Planner* and the *Partial Motion Planner* in order to define the next active waypoint but also to impose further restrictions on the available free space trajectory search. Furthermore, the parking zone areas that include vast free space areas could be explored more efficiently by including a global navigation function metric.

Chapter 7

Conclusion and outlook

7.1 Conclusion

In recent years the automotive industry has been spending increasing efforts in development of Advanced Driver Assistance Systems (ADAS) and their introduction to the modern day vehicles, which should significantly increase the safety, comfort and efficiency of vehicle traffic in the present and future. Several categories of ADAS have been implemented in practice regarding different traffic scenarios, ranging from notification and warning to the driver to automatic control in mitigation of potential critical traffic situations. However, the full range of vehicle maneuverability, sensorial and perceptional capabilities are yet to be exploited.

The issue of autonomous navigation of mobile platforms, which in its essence include environment modeling, motion planning and control, has been a key research area in the robotic community from its very beginnings. Additionally to the more traditional mobile platforms such as indoor exhibition/service mobile robots or offroad exploration rovers, there is also an increase in research effort in autonomous navigation of passenger vehicles, which can be regarded as a complementary approach to the automotive industry's ADAS evolution.

In this sense, this thesis was concerned in autonomous vehicle navigation of vehicles for static as well as dynamic urban scenarios. The main motivation is the

fact that the fully autonomous vehicle agents could in future completely replace the driver engagement in the traffic. This would allow for a much more efficient traffic analysis and reduction of perceptional uncertainty. Moreover, it would allow for precise and optimized traffic planning and control, increasing the timing and energy efficiency, but most importantly bringing the overall traffic safety level to a significantly higher level, potentially life-saving.

The thesis explored several key elements needed to achieve full autonomous driving, namely vehicle modeling, autonomous navigation in unstructured or static environments, autonomous navigation in the presence of dynamic obstacles and dynamic scene analysis. The vehicle modeling, the hierarchical autonomous navigation architecture for static environments and the dynamic scene analysis were tested on an experimental vehicle platform. The motion planning techniques proposed in the presence of dynamic obstacles were investigated in simulation with the particular interest to the case of urban traffic navigation, taking into account the particularities of this environment with respect to dynamic object types such as other participating vehicles and pedestrians, as well as the road network structure.

7.2 Outlook

The future work based on the results in this thesis would primarily include further improvements for the case of autonomous navigation in dynamic urban scenarios. From the point of view of navigation and motion planning, the hierarchical navigation scheme would have to be refined to include the intermediate planning level that would control the transitions on the level of road segments. Increased levels of safety could be implemented by taking into account the maneuvering capabilities of the dynamic objects. From the point of view of dynamic scene analysis, the current dynamic regions detection module would have to be enhanced in order to robustly extract the traffic participant objects by using additional segmentation

techniques as well as the information about the lanes of the road structure. The motion prediction of the dynamic objects could therefore be more complex and take into account their deliberation.

Ultimately, the goal would be to verify the proposed navigation architecture for the dynamic urban scenarios, from perception to planning in different real life experimental scenarios.

Appendices

Appendix A

Route Nework Definition File (RNDF)

Route Nework Definition File (RNDF) is given for the simulated urban environment of Chap. 6 (*cf* Fig. 6.2, 6.3 and 6.5).

```
RNDF_name          urban_scene_rndf_3D
num_segments       12
num_zones          1
format_version     1.0
format_angle       cartesian
creation_date      19June09

segment 1
num_lanes          2
segment_name       segment_1

lane     1.1
num_waypoints      3
lane_width         5
```

```
left_boundary   broken_white
right_boundary solid_white
1.1.1    -62.5    52.5     0.0
1.1.2    -62.5    30.0     0.0
1.1.3    -62.5    7.5      0.0
checkpoint       1.1.1    1
checkpoint       1.1.2    2
checkpoint       1.1.3    3
exit             1.1.3    4.2.1
exit             1.1.3    12.1.1
end_lane

lane 1.2
num_waypoints 3
lane_width    5
left_boundary   broken_white
right_boundary solid_yellow
1.2.1   -57.5    7.5      0.0
1.2.2   -57.5    52.5     0.0
checkpoint       1.2.1    4
checkpoint       1.2.2    5
exit             1.2.2    2.2.1
end_lane
end_segment

segment 2
num_lanes        2
segment_name     segment_2
```

```
lane      2.1
num_waypoints    3
lane_width       5
left_boundary   solid_white
right_boundary solid_white
2.1.1    -7.5     62.5     0.0
2.1.2    -30.0    62.5     0.0
2.1.3    -52.5    62.5     0.0
checkpoint       2.1.1    6
checkpoint       2.1.2    7
checkpoint       2.1.3    8
exit             2.1.3    1.1.1
end_lane

lane 2.2
num_waypoints 3
lane_width     5
left_boundary   solid_white
right_boundary solid_yellow
2.2.1   -52.5    57.5      0.0
2.2.2   -30.0    57.5      0.0
2.2.3   -7.5     57.5      0.0
checkpoint       2.2.1    9
checkpoint       2.2.2    10
checkpoint       2.2.3    11
exit             2.2.3    3.1.1
exit             2.2.3    5.2.1
end_lane
end_segment
```

```
segment 3
num_lanes         2
segment_name      segment_3

lane      3.1
num_waypoints     2
lane_width        5
left_boundary     broken_white
right_boundary    solid_white
3.1.1     -2.5    52.5      0.0
3.1.2     -2.5    7.5       0.0
checkpoint        3.1.1     12
checkpoint        3.1.2     13
exit              3.1.2     4.1.1
exit              3.1.2     10.1.1
exit              3.1.2     7.2.1
stop              3.1.2
end_lane

lane 3.2
num_waypoints 2
lane_width    5
left_boundary     broken_white
right_boundary    solid_yellow
3.2.1   2.5       7.5       0.0
3.2.2   2.5       52.5      0.0
checkpoint        3.2.1     14
checkpoint        3.2.2     15
```

```
exit              3.2.2    5.2.1
exit              3.2.2    2.1.1
stop              3.2.2
end_lane
end_segment

segment 4
num_lanes         2
segment_name      segment_4

lane     4.1
num_waypoints    3
lane_width       5
left_boundary    broken_white
right_boundary   solid_yellow
4.1.1    -7.5     2.5      0.0
4.1.2    -30.0    2.5      0.0
4.1.3    -52.5    2.5      0.0
checkpoint        4.1.1    16
checkpoint        4.1.2    17
checkpoint        4.1.3    18
exit              4.1.3    1.2.1
exit              4.1.3    12.1.1
end_lane

lane 4.2
num_waypoints 3
lane_width    5
left_boundary   broken_white
```

```
right_boundary  solid_yellow
4.2.1    -52.5    -2.5      0.0
4.2.2    -30.0    -2.5      0.0
4.2.3    -7.5     -2.5      0.0
checkpoint       4.2.1     19
checkpoint       4.2.2     20
checkpoint       4.2.3     21
exit             4.2.3     10.1.1
exit             4.2.3     7.2.1
exit             4.2.3     3.2.1
end_lane
end_segment

segment 5
num_lanes        2
segment_name     segment_5

lane     5.1
num_waypoints   3
lane_width       5
left_boundary   broken_white
right_boundary  solid_white
5.1.1    52.5     62.5      0.0
5.1.2    30.0     62.5      0.0
5.1.3    7.5      62.5      0.0
checkpoint       5.1.1     22
checkpoint       5.1.2     23
checkpoint       5.1.3     24
exit             5.1.3     2.1.1
```

```
exit              5.1.3    3.1.1
end_lane

lane 5.2
num_waypoints 3
lane_width    5
left_boundary   broken_white
right_boundary solid_yellow
5.2.1   7.5     57.5     0.0
5.2.2   30.0    57.5     0.0
5.2.3   52.5    57.5     0.0
checkpoint       5.2.1    25
checkpoint       5.2.2    26
checkpoint       5.2.3    27
exit             5.2.3    6.1.1
end_lane
end_segment

segment 6
num_lanes        2
segment_name     segment_6

lane     6.1
num_waypoints    3
lane_width       5
left_boundary   double_yellow
right_boundary solid_yellow
6.1.1    57.5    52.5     0.0
6.1.2    57.5    30.0     0.0
```

```
6.1.3    57.5     7.5      0.0
checkpoint       6.1.1    28
checkpoint       6.1.2    29
checkpoint       6.1.3    30
exit             6.1.3    7.1.1
exit             6.1.3    8.1.1
stop             6.1.3
end_lane

lane 6.2
num_waypoints 3
lane_width    5
left_boundary   double_yellow
right_boundary solid_white
6.2.1   62.5   7.5        0.0
6.2.2   62.5   30.0       0.0
6.2.3   62.5   52.5       0.0
checkpoint      6.2.1    31
checkpoint      6.2.2    32
checkpoint      6.2.3    33
exit            6.2.3    5.1.1
end_lane
end_segment

segment 7
num_lanes       2
segment_name    segment_7

lane    7.1
```

```
num_waypoints    3
lane_width       5
left_boundary   broken_white
right_boundary solid_yellow
7.1.1    52.5     2.5       0.0
7.1.2    7.5      2.5       0.0
checkpoint        7.1.1    34
checkpoint        7.1.2    35
exit              7.1.2    3.2.1
exit              7.1.2    4.1.1
exit              7.1.2    10.1.1
end_lane

lane 7.2
num_waypoints 3
lane_width     5
left_boundary   broken_white
right_boundary solid_yellow
7.2.1    7.5     -2.5       0.0
7.2.2    30.0    -2.5       0.0
7.2.3    52.5    -2.5       0.0
checkpoint        7.2.1    36
checkpoint        7.2.2    37
checkpoint        7.2.3    38
exit              7.2.3    8.1.1
exit              7.2.3    6.2.1
end_lane
end_segment
```

```
segment 8
num_lanes         2
segment_name      segment_8

lane      8.1
num_waypoints     3
lane_width        5
left_boundary   double_yellow
right_boundary  solid_yellow
8.1.1     57.5      -7.5      0.0
8.1.2     57.5      -30.0     0.0
8.1.3     57.5      -52.5     0.0
checkpoint        8.1.1     40
checkpoint        8.1.2     41
checkpoint        8.1.3     42
exit              8.1.3     9.1.1
end_lane

lane 8.2
num_waypoints 3
lane_width     5
left_boundary   double_yellow
right_boundary  solid_white
8.2.1   62.5    -52.5       0.0
8.2.2   62.5    -30.0       0.0
8.2.3   62.5    -7.5        0.0
checkpoint        8.2.1     43
checkpoint        8.2.2     44
checkpoint        8.2.3     45
```

```
exit              8.2.3    6.2.1
exit              8.2.3    7.1.1
stop              8.2.3
end_lane
end_segment


segment 9
num_lanes         2
segment_name      segment_9

lane     9.1
num_waypoints    3
lane_width       5
left_boundary   solid_yellow
right_boundary  solid_yellow
9.1.1    52.5     -57.5    0.0
9.1.2    30.0     -57.5    0.0
9.1.3    7.5      -57.5    0.0
checkpoint        9.1.1    46
checkpoint        9.1.2    47
checkpoint        9.1.3    48
exit              9.1.3    10.2.1
exit              9.1.3    11.1.1
end_lane

lane 9.2
num_waypoints 3
lane_width     5
```

```
left_boundary   solid_yellow
right_boundary  solid_white
9.2.1   7.5     -62.5       0.0
9.2.2   30.0    -62.5       0.0
9.2.3   52.5    -62.5       0.0
checkpoint        9.2.1     49
checkpoint        9.2.2     50
checkpoint        9.2.3     51
exit              9.2.3     8.2.1
end_lane
end_segment

segment 10
num_lanes         2
segment_name      segment_10

lane      10.1
num_waypoints     3
lane_width        5
left_boundary   broken_white
right_boundary  solid_yellow
10.1.1  -2.5      -7.5      0.0
10.1.2  -2.5      -30.0     0.0
10.1.3  -2.5      -52.5     0.0
checkpoint        10.1.1    52
checkpoint        10.1.2    53
checkpoint        10.1.3    54
exit              10.1.3    11.1.1
exit              10.1.3    9.2.1
```

```
end_lane

lane 10.2
num_waypoints 4
lane_width    5
left_boundary  broken_white
right_boundary solid_yellow
10.2.1  2.5     -52.5   0.0
10.2.2  2.5     -35.0   0.0
10.2.3  2.5     -25.0   0.0
10.2.4  2.5     -7.5    0.0
checkpoint      10.2.1  55
checkpoint      10.2.2  56
checkpoint      10.2.3  57
checkpoint      10.2.4  58
exit            10.2.4  7.2.1
exit            10.2.4  3.2.1
exit            10.2.4  4.1.1
exit            10.2.2  100.3.1
end_lane
end_segment

segment 11
num_lanes       2
segment_name    segment_11

lane    11.1
num_waypoints   3
lane_width      5
```

```
left_boundary   broken_white
right_boundary solid_yellow
11.1.1   -7.5     -57.5    0.0
11.1.2   -30.0    -57.5    0.0
11.1.3   -52.5    -57.5    0.0
checkpoint        11.1.1   59
checkpoint        11.1.2   60
checkpoint        11.1.3   61
exit              11.1.3   12.2.1
end_lane

lane 11.2
num_waypoints 3
lane_width     5
left_boundary   broken_white
right_boundary solid_white
11.2.1   -52.5    -62.5    0.0
11.2.2   -30.0    -62.5    0.0
11.2.3   -7.5     -62.5    0.0
checkpoint        11.2.1   62
checkpoint        11.2.2   63
checkpoint        11.2.3   64
exit              11.2.3   9.2.1
exit              11.2.3   10.2.1
end_lane
end_segment

segment 12
num_lanes         2
```

```
segment_name     segment_12

lane     12.1
num_waypoints    2
lane_width       5
left_boundary   solid_white
right_boundary solid_white
12.1.1   -62.5    -7.5     0.0
12.1.2   -62.5    -52.5    0.0
checkpoint       12.1.1   65
checkpoint       12.1.2   66
exit             12.1.2   11.2.1
end_lane

lane 12.2
num_waypoints 2
lane_width     5
left_boundary   solid_white
right_boundary solid_yellow
12.2.1   -57.5    -52.5    0.0
12.2.2   -57.5    -7.5     0.0
checkpoint       12.2.1   67
checkpoint       12.2.2   68
exit             12.2.2   4.2.1
exit             12.2.2   1.2.1
stop             12.2.2
end_lane
end_segment
```

```
zone                100
num_spots           3
zone_name           city_parking

perimeter           100.0
num_perimeterpoints      6
100.0.1             5.0       -27.5     0.0
100.0.2             5.0       -5.0      0.0
100.0.3             55.0      -5.0      0.0
100.0.4             55.0      -55.0     0.0
100.0.5             5.0       -55.0     0.0
100.0.6             5.0       -32.5     0.0
exit                100.0.1   10.2.3
end_perimeter

spot                100.1
spot_width          5
100.1.1             40        -17       0.0
100.1.2             40        -15       0.0
checkpoint          100.1.2    69
end_spot

spot                100.2
spot_width          5
100.2.1             20        -45       0.0
100.2.2             20        -47       0.0
checkpoint          100.2.2    70
end_spot
```

```
spot            100.3
100.3.1         5.0     -32.5   0.0
checkpoint      100.3.1  71
end_spot
end_zone
end_file
```

Bibliography

[Alami *et al.*, 02] R. Alami, T. Simeon, and K. Madhava Krishna. On the influence of sensor capacities and environment dynamics onto collision-free motion plans. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Lausanne (CH), October 02.

[Apostoloff and Zelinsky, 2002] N. Apostoloff and A. Zelinsky. Vision in and out of vehicles: Integrated driver and road scene monitoring. In *Proceedings of the International Symposium of Experimental Robotics*, Italy, 2002.

[Bacha *et al.*, 2008] Andrew Bacha, Cheryl Bauman, Ruel Faruque, Michael Fleming, Chris Terwelp, Charles Reinholtz, Dennis Hong, Al Wicks, Thomas Alberi, David Anderson, Stephen Cacciola, Patrick Currier, Aaron Dalton, Jesse Farmer, Jesse Hurdus, Shawn Kimmel, Peter King, Andrew Taylor, David Van Covern, and Mike Webster. Odin: Team victortango's entry in the darpa urban challenge. *J. Field Robot.*, 25(8):467–492, 2008.

[Bartels *et al.*, 1987] R.H. Bartels, J.C. Beatty, and B.A. Barsky. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling.* Morgan Kaufmann Publishers, Inc., Los Altos, CA 94022, 1987.

[Bertozzi and Broggi, 1999] Massimo Bertozzi and Alberto Broggi. Tools for code optimization and system evaluation of the image processing system paprica-3. *Journal of Systems Architecture*, 45(6-7):519–542, 1999.

[Bertozzi *et al.*, 2000] Massimo Bertozzi, Alberto Broggi, and Alessandra Fascioli. Vision-based intelligent vehicles: State of the art and perspectives. *Robotics and Autonomous Systems*, 32(1):1–16, 2000.

[Beuvais and Kreucher, 1997] L. Michael Beuvais and C. Kreucher. Building world model for mobile platforms using heterogeneous sensors fusion and temporal analysis. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, page 101, Boston, MA, November 1997.

[Blackman and Popoli, 1999] S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems.* Artech House, 1999.

[Bohren *et al.*, 2008] Jonathan Bohren, Tully Foote, Jim Keller, Alex Kushleyev, Daniel Lee, Alex Stewart, Paul Vernaza, Jason Derenick, John Spletzer, and Brian Satterfield. Little ben: The ben franklin racing team's entry in the 2007 darpa urban challenge. *J. Field Robot.*, 25(9):598–614, 2008.

[Broggi *et al.*, 1999] A. Broggi, M. Bertozzi, A. Fascioli, and G. Conte. Automatic vehicle guidance: The experience of the argo vehicle. In *World Scientific*. Singapore, 1999.

[Broggi *et al.*, 2001] Alberto Broggi, Massimo Bertozzi, Gianni Conte, and Alessandra Fascioli. Argo prototype vehicle. In Ljubo Vlacic, Michel Parent, and Fumio Harashima, editors, *Intelligent Vehicle Technologies*, pages 445–493. Butterworth-Heinemann, Oxford, 2001.

[Buehler *et al.*, 2008] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. Editorial. *J. Field Robot.*, 25(8):423–424, 2008.

[Buehler, 2006] Martin Buehler. Summary of dgc 2005 results. *Journal of Field Robotics*, 23(8):465–466, 2006.

[Burckhardt, 1993] M. Burckhardt. *Fahrwerktechnik: Radschlupf-Regelsysteme*. Vogel Fachbuch, Wuerzburg, 1993.

[Burlet *et al.*, 2007] J. Burlet, T.D. Vu, and O. Aycard. Grid-based localization and online mapping with moving object detection and tracking. Rapport de recherche 167687, INRIA, 2007.

[Canny, 1986] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[Canudas de Wit *et al.*, 1995] Carlos Canudas de Wit, H. Olsson, K. J. Astroem, and Lischinsky. A new model for control of systems with friction. *IEEE Transactions on Automatic Control*, 40(3), 1995.

[c:I, 1997] *ISO, Mechanical vibration and shock - Evaluation of human exposure to whole body vibrations - Part 1: General requirements, ISO 2631-1*. 1997.

[Claeys *et al.*, 2001] Xavier Claeys, Jingang Yi, Luis Alvarez, Roberto Horowitz, and Carlos Canudas de Wit. A dynamictire/road friction model for 3d vehicle control and simulation. In *IEEE Intelliegent Transportation Systems*, pages 483–488, Oakland(CA), USA, 2001.

[Continental-ADAS, 2010] `http://www.conti-online.com/`, March 2010.

[Coué *et al.*, 2006] C. Coué, C. Pradalier, C. Laugier, T. Fraichard, and P. Bessière. Bayesian occupancy filtering for multitarget tracking: an automotive application. *International Journal of Robotics Research (IJRR)*, 25(1):19–30, 2006.

[Crisman and Thorpe, 1991] J.D. Crisman and C.E. Thorpe. Unscarf-a color vision system for the detection of unstructured roads. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 2496–2501 vol.3, apr 1991.

[de Boor, 2001] C. de Boor. *A practical guide to splines*. Applied mathematical sciences 27. Springer, 2001.

[Dickmanns and Christians, 1991] E.D. Dickmanns and Th. Christians. Relative 3d-state estimation for autonomous visual guidance of road vehicles. *Robotics and Autonomous Systems*, 7(2-3):113–123, 1991. Special Issue Intelligent Autonomous Systems.

[Dickmanns, 1991] E.D. Dickmanns. 4-d dynamic vision for intelligent motion control. *Engineering Applications of Artificial Intelligence*, 4(4):301–307, 1991.

[Dijkstra, 1959] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[eImpact, 2008] eimpact - socio-economic impact assessment of stand-alone and co-operative intelligent vehicle safety systems (ivss) in europe. Final Report and Integration of Results and Perspectives for market introduction of IVSS, eImpact Partners, November 2008. `http://www.tno.nl`.

[Ferguson and Stentz, 2005] Dave Ferguson and Anthony Stentz. Field D*: An interpolation-based path planner and replanner. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2005.

[Ferguson and Stentz, 2007] D. Ferguson and A. Stentz. Anytime, dynamic planning in high-dimensional search spaces. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[Fletcher *et al.*, 2008] Luke Fletcher, Seth Teller, Edwin Olson, David Moore, Yoshiaki Kuwata, Jonathan How, John Leonard, Isaac Miller, Mark Campbell, Dan Huttenlocher, Aaron Nathan, and Frank-Robert Kline. The mit–cornell collision and why it happened. *J. Field Robot.*, 25(10):775–807, 2008.

[Fleury *et al.*, 1997] S. Fleury, M. Herrb, and R. Chatila. Genom: A tool for the specification and the implementation of operating modules in a distributed robot architecture. In *International Conference on Intelligent Robots and Systems*, volume 2, pages 842–848, Grenoble (France), September 1997. IEEE.

[Fox *et al.*, 1996] D. Fox, W. Burgard, and S. Thrun. Controlling synchro-drive robots with the dynamic window approach to collision avoidance. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1996.

[Fraichard and Asama, 2004a] Th. Fraichard and H. Asama. Inevitable collision states - a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.

[Fraichard and Asama, 2004b] Th. Fraichard and H. Asama. Inevitable collision states. a step towards safer robots? *Advanced Robotics*, 18(10), 2004.

[Fraichard, 1992] C. Fraichard, T.; Laugier. Kinodynamic planning in a structured and time-varying 2-d workspace. In *International Conference on Robotics and Automation*. IEEE, May 1992.

[Fraichard, 2007] Th. Fraichard. A short paper about motion safety. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Roma (IT), April 2007.

[Franke *et al.*, 1998] U. Franke, D. Gavrila, S. Gorzig, F. Lindner, F. Puetzold, and C. Wohler. Autonomous driving goes downtown. *Intelligent Systems and their Applications, IEEE*, 13(6):40–48, nov/dec 1998.

[Franke *et al.*, 2001] Uwe Franke, Dariu Gavrila, Axel Gern, Steffen Gorzig, Reinhard Janssen, Frank Paetzold, and Christian Wohler. From door to door – principles and applications of computer vision for driver assistant systems. In Ljubo Vlacic, Michel Parent, and Fumio Harashima, editors, *Intelligent Vehicle Technologies*, pages 131–188. Butterworth-Heinemann, Oxford, 2001.

[Gregor *et al.*, 2002] R. Gregor, M. Lutzeler, M. Pellkofer, K.-H. Siedersberger, and E.D. Dickmanns. Ems-vision: a perceptual system for autonomous vehicles. *Intelligent Transportation Systems, IEEE Transactions on*, 3(1):48–59, mar 2002.

[Haralick and Shapiro, 1992] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision Vol.1*. Addison-Wesley Publishing Company, Inc., 1992.

[Hart *et al.*, 1968] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[Iagnemma and Buehler, 2006] Karl Iagnemma and Martin Buehler. Editorial for journal of field robotics-special issue on the darpa grand challenge. *Journal of Field Robotics*, 23(8):461–462, 2006.

[Isard and Blake, 1998] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998.

[Jochem *et al.*, 1993] T.M. Jochem, D.A. Pomerleau, and C.E. Thorpe. Maniac: A next generation neurally based autonomous road follower. In *Proc. of the Third International Conference on Intelligent Autonomous Systems*, pages 341–346, Pittsburg, PA, February 1993.

[Junjie *et al.*, 2004] He Junjie, D. A. Crolla, Martin C. Levesley, and Warren J. Manning. Integrated active steering and variable torque distribution control for improving vehicle handling and stability. *Vehicle Dynamics and Simulation 2004*, pages 107–116, 2004.

[Kelly, 2002] A. Kelly. Reactive nonholonomic trajectory generation via parametric optimal control. *ijrr*, 2002.

[Kelly, 2004] A. Kelly. Linearized error propagation in odometry. *The International Journal of Robotics Research*, 23(2):179–218, February 2004.

[Kiencke and Nielsen, 2000a] U. Kiencke and L. Nielsen. *Automotive control systems*. Springer, Inc., 2000.

[Kiencke and Nielsen, 2000b] Uwe Kiencke and Lars Nielsen. *Automotive Control Systems*. Springer, Berlin; Heidelberg; New York; Hong Kong; London; Milan; Paris; Singapore; Tokyo, 2000.

[Kiencke and Nielsen, 2005] Uwe Kiencke and Lars Nielsen. *Automotive Control Systems*. Springer, Berlin; Heidelberg; New York; Hong Kong; London; Milan; Paris; Singapore; Tokyo, 2 edition, 2005.

[Koike *et al.*, 2003] C. Koike, C. Pradalier, P. Bessière, and E. Mazer. Obstacle avoidance and proscriptive bayesian programming. In *Proc. of the Workshop on Reasoning with Uncertainty in Robotics*, Acapulco (MX), July 2003.

[Kondak and Hommel, 2001] K. Kondak and G. Hommel. Computation of time optimal movements for autonomous parking of non-holonomic mobile platforms. In *International Conference on Robotics and Automation*, Seoul (Korea), May 2001. IEEE.

[Kuffner and LaValle, 2000] J. Kuffner and S. LaValle. RRT-Connect: An efficient approach to single-query path planning. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA, 2000.

[Kurjanowicz, 2006] Ron Kurjanowicz. Foreword for journal of field robotics-special issue on the darpa grand challenge. *Journal of Field Robotics*, 23(8):463–464, 2006.

[Lamon *et al.*, 2006a] P. Lamon, S. Kolski, and R. Siegwart. The SmartTer - a vehicle for fully autonomous navigation and mapping in outdoor environments. In *Proc. of the CLAWAR*, Brussels, Belgium, 2006.

[Lamon *et al.*, 2006b] P. Lamon, S. Kolski, and R.. Siegwart. The smartter - a vehicle for fully autonomous navigation and mapping in outdoor environments. In *Proc. of the International Conference on Climbing and Walking Robots*, 2006.

[Larsen *et al.*, 2000] E. Larsen, S. Gottschalk, M.C. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[Latombe, 1991] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, Mass., 1991.

[Laumond, 1998] J.P. Laumond. *Robot motion planning and control*. Lecture notes in control and information sciences 229. Springer, 1998.

[LaValle and Kuffner Jr., 2001] S. M. LaValle and J. J. Kuffner Jr. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.

[LaValle and Kuffner, 2001] S. M. LaValle and J. J. Kuffner. *Algorithmic and Computational Robotics: New Directions*, chapter Rapidly-exploring random trees: Progress and prospects, pages 293–308. Wellesley,MA, 2001.

[Lützeler and Dickmanns, 1998] A.M. Lützeler and E.D. Dickmanns. Road recognition with marveye. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 341–346, Stuttgart, Germany, October 1998.

[Macek and Siegwart, 2006] K. Macek and R. Siegwart. Motion planning in the presence of moving obstacles using RRT search and B-splines. In *Proc. of the 8th International IFAC Symposium on Robot Control (SYROCO)*, 2006.

[Macek *et al.*, 2004] K. Macek, B. Williams, S. Kolski, and R. Siegwart. A lane detection vision module for driver assistance. In *Proc. of the IEEE/APS Conference on Mechatronics and Robotics (MECHROB)*, 2004.

[Macek *et al.*, 2005] K. Macek, I. Petrovic, and R. Siegwart. A control method for stable and smooth path following of mobile robots. In *Proc. of the European Conference on Mobile Robots (ECMR)*, 2005.

[Macek *et al.*, 2006] K. Macek, M. Becker, and R. Siegwart. Motion planning for car-like vehicles in dynamic urban scenarios. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[Macek *et al.*, 2007] K. Macek, K. Thoma, R. Glatzel, and R. Siegwart. Dynamics modeling and parameter identification for autonomous vehicle navigation. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[Macek *et al.*, 2008a] K. Macek, R. Philippsen, and R. Siegwart. Path following for autonomous vehicle navigation with inherent safety and dynamics margin. In *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*, 2008.

[Macek *et al.*, 2008b] K. Macek, D. Vasquez, T. Fraichard, and R. Siegwart. Save vehicle navigation in dynamic urban scenarios. In *Proc. of the Intelligent Transportation Systems Conference (ITSC)*, 2008.

[Maurer and Dickmanns, 1997] M. Maurer and E.D. Dickmanns. A system architecture for autonomous visual road vehicle guidance. In *Intelligent Transportation System, 1997. ITSC '97., IEEE Conference on*, pages 578–583, nov 1997.

[Maurer *et al.*, 1996] M. Maurer, R. Behringer, S. Furst, F. Thomanek, and E.D. Dickmanns. A compact vision system for road vehicle guidance. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 3, pages 313–317 vol.3, aug 1996.

[Maxim Likhachev, 2008] Dave Ferguson Maxim Likhachev. Planning long dynamically-feasible maneuvers for autonomous vehicles. In *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland, June 2008.

[Mekhnacha *et al.*, 2006] K. Mekhnacha, J. M. Ahuactzin, P. Bessiŕe, E. Mazer, and L. Smail. A unifying framework for exact and approximate bayesian inference. Technical Report RR-5797, NRIA - Rhone-Alpes Research Report - E-MOTION team, Montbonnot, France, 2006.

[Miller *et al.*, 2008] Isaac Miller, Mark Campbell, Dan Huttenlocher, Frank-Robert Kline, Aaron Nathan, Sergei Lupashin, Jason Catlin, Brian Schimpf, Pete Moran, Noah Zych, Ephrahim Garcia, Mike Kurdziel, and Hikaru Fujishima. Team cornell's skynet: Robust perception and planning in an urban environment. *J. Field Robot.*, 25(8):493–527, 2008.

[Mitschke, 1990] Manfred Mitschke. *Dynamik der Kraftfahrzeuge, Band C: Fahrverhalten*. Springer, Berlin; Heidelberg; New York; London; Paris; Tokyo; Hong Kong; Singapore, 2 edition, 1990.

[Mitschke, 1995] Manfred Mitschke. *Dynamik der Kraftfahrzeuge, Band A: Antrieb und Bremsung*. Springer, Berlin; Heidelberg; New York; London; Paris; Tokyo; Hong Kong, 3 edition, 1995.

[Montemerlo *et al.*, 2008a] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun. Junior: The stanford entry in the urban challenge. *J. Field Robot.*, 25(9):569–597, 2008.

[Montemerlo *et al.*, 2008b] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden,

Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun. Junior: The stanford entry in the urban challenge. *J. Field Robot.*, 25(9):569–597, 2008.

[Nagel *et al.*, 1995] H. H. Nagel, W. Enkelmann, and G. Struck. Fhg-co-driver: From map-guided automatic driving by machine vision to a cooperative driver support. *Mathematical and Computer Modelling*, 22(4-7):185–212, 1995.

[Neusser *et al.*, 1993] S. Neusser, J. Nijhuis, L. Spaanenburg, B. Hoefflinger, U. Franke, and H. Fritz. Neurocontrol for lateral vehicle guidance. *Micro, IEEE*, 13(1):57–66, feb 1993.

[NHTSA, 2002] The economic impact of motor vehicle crashes, 2000. NHTSA Technical Report, May 2002. `http://www.ntis.gov`.

[NHTSA, 2009] 2008 traffic safety annual assessment – highlights. Traffic Safety Facts, National Center for Statistics and Analysis, June 2009. `http://www-nrd.nhtsa.dot.gov/Pubs/811172.PDF`.

[NHTSA, 2010] Early estimate of motor vehicle traffic fatalities for the first three quarters (january – september) of 2009. Traffic Safety Facts, National Center for Statistics and Analysis, January 2010. `http://www-nrd.nhtsa.dot.gov/Pubs/811255.PDF`.

[OECD, 2003] Road safety - impact of new technologies. Technical Report ITRD Number: E117683, ISBN-92-64-10322-8 OECD, 2003.

[Paetzold and Franke, 2000] F. Paetzold and U. Franke. Road recognition in urban environment. *Image and Vision Computing*, 18(5):377–387, 2000.

[P.E. Gill and Saunders, 2005] W. Murray P.E. Gill and M. A. Saunders. *SNOPT*: An *SQP* algorithm for large-scale constrained optimization. In *SIAM Review*, volume 47, pages 99–131. 2005.

[Petti and Fraichard, 2005] S. Petti and Th. Fraichard. Partial motion planning framework for reactive planning within dynamic environments. In *Proc. of the IFAC/AAAI Int. Conf. on Informatics in Control, Automation and Robotics*, Barcelona (SP), September 2005.

[Philippsen and Siegwart, 2005] Roland Philippsen and Roland Siegwart. An interpolated dynamic navigation function. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

[Philippsen *et al.*, 2007] R. Philippsen, Kolski.S., K. Macek, and R. Siegwart. Path planning, replanning and execution for autonomous driving in urban and

offroad environments. ICRA 2007 Workshop on Planning, Perception and Navigation of Intelligent Vehicles (PPNIV), 2007.

[Philippsen, 2006] Roland Philippsen. A light formulation of the E* interpolated path replanner. Technical report, Autonomous Systems Lab, Ecole Polytechnique Federale de Lausanne, 2006.

[Pradalier *et al.*, 2005] C. Pradalier, J. Hermosillo, C. Koike, C. Braillon, P. Bessière, and C. Laugier. The cycab: a car-like robot navigating autonomously and safely among pedestrians. *Robotics and Autonomous Systems*, 50(1):51–68, 2005.

[Qu *et al.*, 2004] Z. Qu, J. Wang, and C.E. Plaisted. A new analytical solution to mobile robot trajectory generation in the presence of moving obstacles. *IEEE Transactions on Robotics*, 20:978–993, 2004.

[Redmond and Heneghan, 2007] Stephen J. Redmond and Conor Heneghan. A method for initialising the k-means clustering algorithm using kd-trees. *Pattern Recogn. Lett.*, 28(8):965–973, 2007.

[Reimpel and Betzler, 2000] Joernsen Reimpel and Juergen W. Betzler. *Fahrwerktechnik: Grundlagen.* Vogel Verlag, Wuerzburg, 4 edition, 2000.

[rnd, 2007] Route network definition file (RNDF) and mission data file (MDF) formats, March 2007. http://www.darpa.mil/grandchallenge/docs.

[Seder and Petrovic, 2007] M. Seder and I. Petrovic. Dynamic window based approach to mobile robot motion control in the presence of moving obstacles. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[SEiSS, 2005] Exploratory study on the potential socio-economic impact of the introduction of intelligent safety systems in road vehicles. SEiSS Final Report, VDI/VDE Innovation and Institute for Transport Economics at the University of Cologne, January 2005.

[Selby *et al.*, 2001] M. Selby, W. J. Manning, M. D. Brown, and D. A. Crolla. A coordination approach for dyc and active front steering. *Vehicle Dynamics and Simulation 2001*, pages 49–55, 2001.

[Singer, 1970] R.A. Singer. Estimating optimal tracking filer performance for manned maneuvering targets. *IEEE Trans.on Aerospace and Electronic Systems*, AES-5:473–483, July 1970.

[Solea and Nunes, 2006] R. Solea and U. Nunes. Trajectory planning with velocity planner for fully-automated passenger vehicles. In *Proc. of the Intelligent Transportation Systems Conference (ITSC)*, 2006.

[Srinivasan, 2005] B. Srinivasan. *Optimal Control Course: Lecture Notes*. Ecole Polytechnique Fédérale de Lausanne (EPFL), CH, July 2005.

[Stentz, 1994] Anthony Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1994.

[Thorpe *et al.*, 1988] C. Thorpe, M.H. Hebert, T. Kanade, and S.A. Shafer. Vision and navigation for the carnegie-mellon navlab. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 10(3):362–373, may 1988.

[Thorpe *et al.*, 1991a] C. Thorpe, M. Herbert, T. Kanade, and S. Shafer. Toward autonomous driving: the cmu navlab. i. perception. *IEEE Expert*, 6(4):31–42, aug 1991.

[Thorpe *et al.*, 1991b] C. Thorpe, M. Herbert, T. Kanade, and S. Shafter. Toward autonomous driving: the cmu navlab. ii. architecture and systems. *IEEE Expert*, 6(4):44–52, aug 1991.

[Thrun *et al.*, 2001] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. *Robust Monte Carlo Localization for Mobile Robots*, 2001.

[Thrun, 2000] S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93–109, 2000.

[Urmson *et al.*, 2008] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, M. N. Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas M. Howard, Sascha Kolski, Alonzo Kelly, Maxim Likhachev, Matt McNaughton, Nick Miller, Kevin Peterson, Brian Pilnick, Raj Rajkumar, Paul Rybski, Bryan Salesky, Young-Woo Seo, Sanjiv Singh, Jarrod Snider, Anthony Stentz, William "Red" Whittaker, Ziv Wolkowicki, Jason Ziglar, Hong Bae, Thomas Brown, Daniel Demitrish, Bakhtiar Litkouhi, Jim Nickolaou, Varsha Sadekar, Wende Zhang, Joshua Struble, Michael Taylor, Michael Darms, and Dave Ferguson. Autonomous driving in urban environments: Boss and the urban challenge. *J. Field Robot.*, 25(8):425–466, 2008.

[Villella, 2004] Matthew G. Villella. Nonlinear modeling and control of automobiles with dynamic wheel-road friction and wheel torque inputs. Master's thesis, Georgia Institute of Technology, 2004.

[von Stryk,] O. von Stryk. User's guide for *DIRCOL*: a direct collocation method for the numerical solution of optimal control problems. In *http://www.sim.informatik.tu-darmstadt.de/sw/dircol/*.

[von Stryk, 1993] O. von Stryk. Numerical solution of optimal control problems by direct collocation. In J. Stoer R. Bulirsch, A. Miele and K.-H. Well, editors, *Optimal Control - Calculus of Variations, Optimal Control Theory and Numerical Methods*, volume International Series of Numerical Mathematics 111. Birkhäuser, Basel, 1993.

[Vu *et al.*, 2007] Trung-Dung Vu, O. Aycard, and N. Appenrodt. Online localization and mapping with moving object tracking in dynamic outdoor environments. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 190 –195, 13-15 2007.

[Web-DARPA, 2010a] `http://www.darpa.mil/`, March 2010.

[Web-DARPA, 2010b] `http://www.darpa.mil/grandchallenge04/`, March 2010.

[Web-DARPA, 2010c] `http://www.darpa.mil/grandchallenge05/`, March 2010.

[Web-DARPA, 2010d] `http://www.darpa.mil/grandchallenge/index.asp`, March 2010.

[Wiki-ADAS-de, 2010] `http://de.wikipedia.org/wiki/Fahrerassistenzsystem`, March 2010.

[Wiki-ADAS-en, 2010] `http://en.wikipedia.org/wiki/Advanced_driver_assistance_systems`, March 2010.

[Williams, 1996] James H. Jr. Williams. *Fundamentals of Applied Mechanics*. John Wiley & Sons, Inc., New York; chichester; Toronto; Singapore, 1996.

[Xu *et al.*, 2000] Y. Xu, R. Wang, and J.S. Libling. A vision navigation algorithm based on liner lane model. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, Dearborn, MI, 2000.