Perception for driverless vehicles: design and implementation

Rodrigo Benenson

Collège doctoral
ED n° 431 : Information, communication, modélisation et simulation

N° attribué par la bibliothèque
|__|__|__|__|__|__|__|__|__|__|

T H È S E

pour obtenir le grade de
Docteur de l'École des Mines de Paris
Spécialité "Informatique temps réel – Robotique – Automatique"

présentée et soutenue publiquement par
Rodrigo BENENSON

le 25/11/2008

Perception pour véhicule urbain sans conducteur :
Conception et implémentation

Directeur de thèse : Christian LAUGIER

Jury

M. Philippe BONNIFAIT	Rapporteur
M. Philippe MARTINET	Rapporteur
M. Claude LAURGEAU	Examinateur
Mme. Brigitte d'ANDRÉA-NOVEL	Examinateur
M. Javier IBAÑEZ-GUZMAN	Examinateur

Perception for urban driverless vehicles: design and implementation

Rodrigo Benenson

First draft on 6th December 2007
Phd defense on 25th November 2008
Final version on 30th June 2009

First of all I would like to thank the LaRA team who hosted me and provided me with a great environment to innovate, experiment and learn. I thank Michel Parent and Claude Largeau for giving me the opportunity of doing this PhD thesis. I also thank my mentor Mikael Kais who introduced me to the field of driverless vehicles and taught me the tricks of the trade. I thank my PhD director and reviewers for their advice during the construction of this document. I particularly thank Brigitte d'Andrea-Novel for her guidance during the last months of my thesis.

In the lab, I spent countless hours working on the vehicles and preparing field experiments and demonstrations. Laurent Bouraoui and Armand Yvet were by my side and provided me with their continuous support. Thanks for your good humor and productive spirit all along. I will also remember the good spirit of Stephane Petti, my office mate, with whom I shared long hours of productive discussions about research and life. Along similar lines I have to thank Séverin Lemaignan who stayed at the lab for a shorter time, but made a great impression on us with his creative spirit, his will to build a better world and the great discussions we shared.

I thank the tolerance, good will and effort of the intern students I directed, Tomas Suarez, Sébastien Boissé, Zhiyu Xu,and Gaurav Taank, I have learnt from you more that you probably expect.

Chantal Chazelas helped me many times understand the Kafkian procedures and making them easy as a breeze. Thank you for your professionalism and excellent spirit for all these years. Konaly Sar and Christine Vignaud offered as much support when dealing with the ENSMP paperwork.

I would like to give my greetings to the developers of the open source documents editor LyX, which has made my work life infinitely more pleasant.

Other than the people who helped me during my PhD years I have special acknowledgements for the persons who allowed me to get there. I often remember with nostalgia my school teachers, especially the lessons from professors Tapia, Abelardo, Aravena and my arts teacher Loreto Gonzalez. Of course, I also remember lessons from my university professors, especially those given outside the classroom. Comes to me the most the lessons from professors Jaime Glaria, Mario Salgado, Ricardo Rojas, Claudio Dibb, Juan Hernandez, Carlos Verdugo, and more particularly the ones from professors Michael Seeger, Manuel Young and Javier Ruiz del Solar who introduced me to the world of research.

Other than work, learning and teaching, I have shared, during this PhD years, love, care and joy with family and friends. I could not have succeeded without them. I have to thank the Raskine family for all the love and care they have given me along my life, the Benenson family for their support and their joy, and my friends for cheering me up, encouraging me and sharing good moments. Alvaro, Hugo, Michael, Thomas, Carlos, Bernardo, Juan, Alejandro, Ivan; you are in my mind more often than you would think.

I dedicate this thesis to my mother. The one person who has given me love, care, and who by example and lessons has taught me how to live this life. She is the one person who has always believed in me and that has driven me to pursue my wildest dream. I have, and I will, mom.

A few days after I was accepted to start my PhD on driverless vehicles, I learned about another young man who grew up a few kilometres away from my house. He was an amateur boxer, reared in poverty. In his league, the winner of the matches won a sandwich, the looser went home defeated and with an empty stomach. Other than becoming the national boxing champion, his main aspiration in life was to succeed in finishing the 12 years of basic education. By doing so he would be able to obtain a license to be a professional driver, his dream job. The exact same job I was aiming to render obsolete through my research on driverless vehicles.

Despite the illusions of modernity and comfort, we live in a deeply unfair world. I hope this kind of research work will also help somehow to enhance the life of those who never got the chances I had.

Abstract

The development of driverless vehicles capable of moving on urban roads could provide important benefits in reducing accidents, in increasing life comfort and in providing cost savings. In this dissertation we discuss how to create a perception system allowing robots to drive on roads, without the need to adapt the infrastructure, without requiring previous visits, and considering possible the presence of pedestrians and cars.

We argue that the perception process is application specific and by nature needs to be able to deal with uncertainties in the knowledge of the world. We analyse the particular problem of perception for safe driving in the urban environments and propose a novel solution where the perception process is essentially seen as an optimization process.

Also we propose that the perception process could benefit from collaboration between nearby vehicles. We examine this problem and provide a solution adapted to the constraints encountered in the urban scenario. Here the core issue is formulated as a data association problem.

Both the new perception and collaboration mechanisms were integrated into a full-fledged driverless vehicle system. The results are supported by real world full scale experiments on our automated electric vehicles, the Cycabs.

Figure 1: Two cybercars interacting

Contents

Chapter 1

Introduction

1.1 Social context

> There are three kinds of lies: lies, damned lies, and statistics.
> Benjamin Disraeli

Humans seek happiness and they want to enjoy it for as long as possible. In the last centuries the cultural evolution of the wealthiest countries has evolved up to the point where the average lifespan surpasses 70 years (in 2006 world average was 67 years, with the lowest per country being 35 years in Botswana and the highest 82 years in Japan) [1, 2]. The current trends allow hope for a longer life expectancy worldwide. As the age of death increases the causes of death change, figure 1.1 shows the probable different causes of death as estimated in 2003 in the USA [3]. Current data indicates that *vehicle accidents are one of the main causes of accidental death* in the modern civilized world. The car seems to be one of the most deathly man-made objects used in the civilized world.

In Europe alone road accidents claim about 40000 lives each year and leave more than 1.8 million people injured, representing estimated costs of 160 billion euros (according to 2004 data [4]). Cars, motorcycles, pedestrians and bicycles account for 87% of these deaths. One third of them occur in urban areas.

Worldwide the statistics indicate that around 1.2 million deaths per year are caused by road accidents, making it the eleventh main cause of deaths. Also more than 30 million people per year are injured in road accidents, making them the ninth main cause of morbidity. Considering depression (fourth ranking), it can be seen as the second main human-caused source of morbidity worldwide (higher than war) [5].

Aside from the accidents road transports have other undesirable side effects. They are recognized as a significant source of air pollution and energy consumption [6]. Current trends in world population indicate that the urban environment is becoming the predominant living environment [7]. Even with a restructuration of the productive

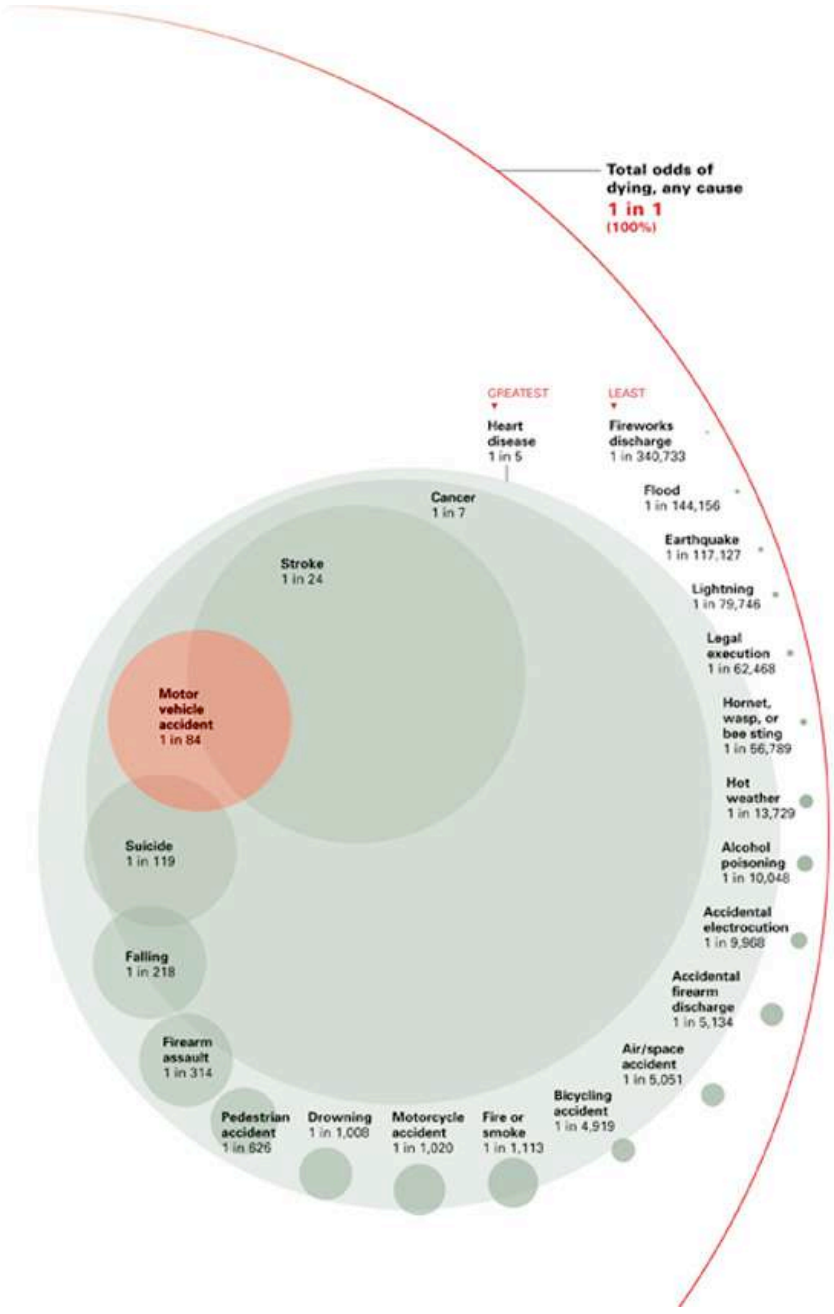


Figure 1.1: Odds of dying in the USA during 2003

system, transport in urban environment is an important function of the city. Addressing the issues of safety, pollution (air, noise) and efficiency (energy consumption, space consumption, peoples time consumption) is a major concern to ensure the quality of life for current and future generations.

1.2 Solution proposal

Existing studies indicate that the human error is one of the main causes of road accidents. It is commonly accepted that human error is involved in the vast majority of car accidents (more than 90% of the accidents) [8].

We believe that the development of robots capable of driving is a viable solution to the problems mentioned. A robot is free of distractions and capable of making objective decisions at every moment in time, ensuring a higher degree of safety. A robot is also capable of driving using quantitative measures of efficiency offering better driving comfort and lower energy consumption. Finally, using telecommunication systems multiple driving robots can have better coordination (at local and city-wide scales) than the local non-verbal communication between human drivers, this would be particularly useful to avoid traffic jams.

In this dissertation we use the term "driverless vehicles" for these robots capable of driving.

1.3 Driverless vehicles

Driving robots now have more than 30 years of research history. In the 1970s the first mobile robots were created, such as the robot "Shakey" by SRI. The first driverless car, developed by the Tsukuba Mechanical Engineering Lab (Japan), was demonstrated in 1977. It ran at 30 [km/h] on a dedicated track. In the 1980s the driverless car created by professor Ernst Dickmanns (working at Universität der Bundeswehr München, Germany) was capable of high speed driving on highways. Since then major developments have followed with large European research initiatives such as PROMETEUS (1987-1995), ARGO (1996-2001), Cybercars (2001-2004, 2006-2008). In the USA the most notable developments come from the CMU Navlab team (1984-present) and the DARPA military research programs DEMO (1991-2001) and Grand Challenge (2004-2007).

In the 1970s the first driverless car appeared. As of 1995 the state of the art allowed high speed driving on highways, over-passing other vehicles, automated parking [9] and following other vehicles. In the 1998 the first fleet of commercial driverless vehicles was put into operation at the Schiphol Amsterdam airport, driving on an dedicated road using infrastructure based localization. The next year Siemens started deploying a new optically-driven bus (following specific markings) where the steering is automated, and the speed is controlled by a human driver. In 2001, Toyota, currently the world's

Figure 1.2: A driverless vehicle, winner of DARPA Challenge 2005

largest car manufacturer, introduced an automated bus system capable of driverless driving on dedicated roads (and still developing it).

As of 2001 the DARPA research programs provided robots able of free driving in rough outdoor terrains. As of 2003 the first driverless vehicle capable of avoiding pedestrian was demonstrated [10]. Using light infrastructure and after a first visit of the area, the vehicle is able to detect moving obstacles (slow speed cars and pedestrian) and update its planned trajectory accordingly. In 2005 the DARPA Grand Challenge shown vehicles capable of crossing 200 [km] of desert roads in a fully autonomous mode at an average speed of 30 [km/h] [11].

In 2006 we designed and implemented a vehicle capable of avoiding moving obstacles outdoors without infrastructure or previous visit of the environment [12]. The same year were held the first demonstrations of vehicles capable of repeating a recorded path without need of infrastructure modifications, the speed is automatically controlled based on the presence of obstacles around the path [13]. Also, the first commercial vehicle with "hands free capabilities" on highways was introduced by Honda.

In 2007 the DARPA Grand Challenge addressed the urban environment (without pedestrians, with low cars density), new commercial outdoor infrastructure based driverless vehicles systems are being built (Ultra PRT at the London Heathrow Airport), and the driver assistance systems still sophisticating.

1.4 Challenges of driverless vehicles in urban environment

Given the current state of the art, proposing the development of driverless vehicles as a solution for the transportation of goods and persons does not seem impossible. The research of the last years has developed enough technology to develop driving robots in controlled outdoor environments, deserts and highways. How to build a robot able to drive in a populated urban environment is still an open problem. How a group (small or large) of driverless vehicles should behave is also still an open problem.

Even when the feasibility of the robotic task has been demonstrated with a prototype, defining the best methods to realize this task, with the lowest constraints and the lowest costs remains a research question.

In this dissertation we will focus on the scenario of driverless vehicles in cities, since it is the most challenging and the one with higher potential impact on daily life. Developing such a robot presents multiple technical challenges at different levels.

1.4.1 Mechanics

Figure 1.3: Toyota Prius uses a drive-by-wire system for the throttle and braking. Steering is automated during assisted parking manoeuvres

First come the mechanics of the robot. The current commercial cars technologies allow to have a thigh relation between the mechanics and the information signals level. Most modern cars have sensors buses, local computers and some of them manage the human commands passing trough a digital representation of the actuation (the so called "drive-by-wire", see figure 1.3). It is reasonable to consider that the mechanical and automation problematics are already solved. Today the bigger challenges are at the signal and information processing level.

1.4.2 People

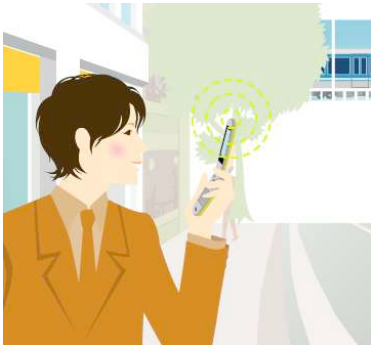


Figure 1.4: A user requests a car from a mobile phone

Independent of the business model employed the cars will be offered directly to the individuals as a ubiquitous system. As such the service will be probably accessed via mobile devices (see figure 1.4).

How to have ubiquitous access to a city sized distributed system?

This leads to the development and design of the hardware infrastructure, the software layers and the communication systems required to accomplish such a task. The actual state of the art where multimedia mobile devices have direct access to the Internet and where web services are provided to thousand of persons simultaneously indicates that creating such an interface is possible. The automatic cars are also mobile devices that can profit from the existing communication infrastructure.

To complete a request for an automatic car it is necessary to specify the human-machine encounter location.

Where is the user calling from?

Mobile devices can specify their location at the city level (which street) using GPS, WiFi localization, by requesting the user to specify his location or by using inexpensive data tags (e.g. 2D bar codes, RFID tags) installed in the city.

Providing access to a city wide driverless vehicles service seems to be feasible with current technologies.

1.4.3 Cars in the city

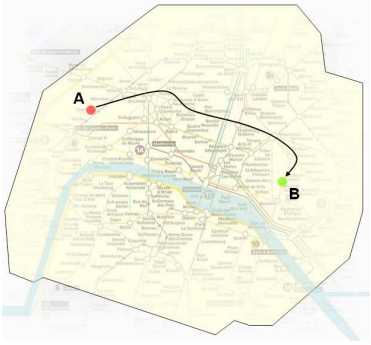


Figure 1.5: Route planning in the city

The displacements of the automated vehicles in the city require a multilevel organization. At the city level the core problem to solve is

Which is the best path to reach point B from point A?

Depending on the situation "best" should be the fastest path or the most energy efficient path. The problem is not trivial because it has to dynamically optimize the path considering the geometric constraints and the traffic congestion in time (which can be predicted based on historical data, current displacement requests and the online network status).

How to predict the traffic flow?

Related to the path finding there are multiple related non trivial optimization problems such as

Which vehicle should serve a request? When to refuel? Where to place non used vehicles?

These logistic aspects are necessary to provide the service to the upper layer. The current state of the art provides algorithms to obtain suboptimal solutions [14], however defining cheaper ways to compute more effective solutions to these large optimization problems remains an open problem.

1.4.4 Cars in the street

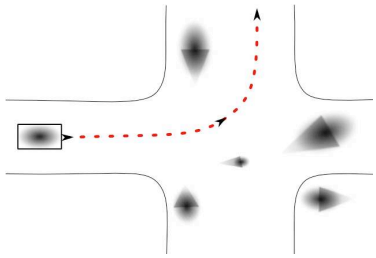


Figure 1.6: Trajectory planning in the street

At the city level the paths are defined as a sequences of streets to choice (i.e., route planning). At the street level the specific trajectory (sequence of states in time) has to be depicted in order to accomplish the objectives defined at the higher level. Trajectory planning is a difficult problem. Safe trajectory planning of nonholonomic robots in dynamic environments with uncertain data is a very difficult problem.

How to find a safe trajectory at real time in an uncertain dynamic environment*?*

This topic is still a very active area of research, both for individual vehicle planning and for collaborative multi-vehicles planning. As a starting point the reader can consult [15].

In order to define a plan the robot has to be able to perceive its environment and it needs to control its movements to follow the decided trajectory.

1.4.5 Perception

Figure 1.7: Example of complex unstructured scenes

The perception module transforms the sensors data into consistent and useful information. The quality and richness of the obtained information will have a direct repercussion over the performances of the control and planning modules. Some of the basic questions that the perception modules has to answer (from the robot perspective) are:

Where I am? (localization)

What is around me? (exoperception, obstacles detection)

Are there indications on the road? (application specific perception)

The ultimate goal of the perception is scene understanding, i.e., identifying all the objects, their location and assigning roles/objectives for each element, this would allow accurate predictions of the future.

The actual state of the art lets us to implement SLAMMOT, Simultaneous Localization, Mapping and Motion Objects Tracking. These systems are able to construct a consistent map of the visited places, localize the robot in such maps and track moving objects in the sensors coverage region around the robot [16, 17].

A key concept to remember is that the perception module does not only describes what we know of the environment but also has to accurately describe the uncertainty of such a representation of the world.

In the following chapters we will discuss in more details the needs of the perception module and how to solve the underlying problems.

1.4.6 Control

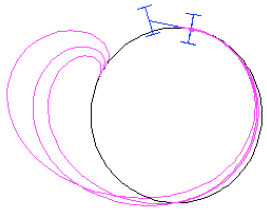


Figure 1.8: Control of a nonholonomic robot

In order to allow the cars displace themselves through the streets it is necessary to have control loop to ensure that the robot follow the physical trajectory defined by the planner.

How to control a car?

The four wheels steerable vehicles are nonholonomic the problem is not trivial.

The control module is tightly coupled to the perception and the trajectory planner. Poor trajectories will be hard to execute, poor environment observation will lead to divergence between the estimated state and the reality and can also force the planner to realize abrupt changes in the trajectories.

It has been proved that classic linearization of non-linear models cannot be used to control car-like systems satisfactorily, thus non-linear control method needs to be used as a starting point the reader can consult [18, chapter 4]. Non linear control of non holonomic robots is still an open problem for the general case, however satisfactory solutions exist for the car-like robot case.

1.4.7 Vehicle to vehicle communication

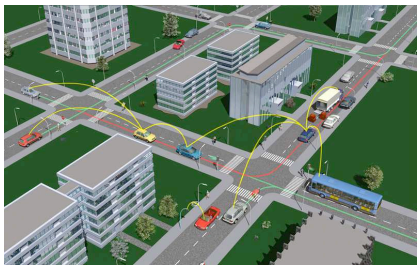


Figure 1.9: Car to car communication illustrated

Nowadays it exists a shared conjecture on the benefits of vehicle to vehicle communications. It is expected that vehicle to vehicle communications will enhance the safety and efficiency of human driven vehicles. Multiple initiatives have been deployed to explore these ideas (Car2car, React, Com2React, WillWarn, Safespot, just to name some European projects). If benefits are possible for human driven vehicles, the same applies to driverless vehicle.

It is expected that by exchanging data the vehicles will increase their knowledge of the situation, and that more information will allow better decisions. Vehicle to vehicle communication for driverless vehicles can operate as an extension or replacement of the driving rules for humans, which are mainly used to allow one human to predict the behaviour of another human. The exchange of data could replace such rules based predictions.

In order to deploy such a system, the first challenge to solve is

How to do vehicle to vehicle communications?

Current commercial communication systems do not provide the expected bandwidth, latency or quality of service for this application. The development of new vehicle to infrastructure or direct vehicle to vehicle communications are required to deploy such ideas. "Network mobility" (NEMO), "Mobile ad-hoc networks" (MANET), "Viral communications", are some of the keywords in this area of research.

Even supposing infinite bandwidth and zero latency, it is still not clear which data will be useful for the particular application. How much preprocessing is needed, which application protocol use, etc.... Upon reception the data has to transformed back into useful information, considering the space and time shift.

Which data to exchange?

How to process the exchanged data?

When trying to enhance the safety or the traffic efficiency, these questions remain mainly unsolved. Even more, the overall conjecture has to be resolved by verifying that the benefits compensate the communication and processing costs.

Such a communication system is likely to interact at multiple levels of the driverless vehicle system.

1.4.8 Integration

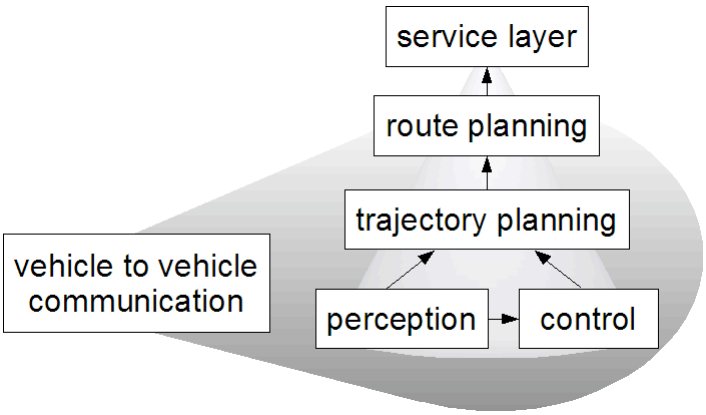


Figure 1.10: Main six areas of the automatic cars system

The described problematics generate six areas of research and development (see figure 1.10) that have to be mixed to generate a final functional and effective automatic cars system.

From an engineering perspective the principal aspects that has to be considered in the final design are:

- *Safety and reliability*
- Ergonomics
- Cost effectiveness
- Deployment issues
- *Automatic cars business models*

Each of the previously listed modules represent by itself a research area with a specific vocabulary, history, a set of open challenges. Inventing solutions to these problems is part of the research, designing a viable system based on existing solutions is an engineering issue. Currently there is no known driverless vehicle capable of driving as good as human do, and thus there is more research to be done.

1.5 Scope of the thesis

In this dissertation we are going to *focus on the perception* module required for robotic driving and it interactions with the other modules.

Setting The application scenario will be driving in populated urban environments such as streets or open areas. We will use as little previous knowledge as possible (assumptions on the scene structure, need of previous visits, etc...). Presence of other cars, pedestrian, bicycles, animals or other mobile objects will be taken into account.

Intelligence and autonomy Some initiatives have declared that "driverless vehicles already exist", since we have automatic cars on highways, or automatic mobile vehicles in industries, or driverless robots on outdoor parcels. Also it has been proposed as a solution for "driverless vehicles in the city" the use of robots capable of repeating a specific path and stopping as obstacles appear. This would be equivalent to a "robotic immaterial trolley". While such a robot is possibly a viable solution for transportation in the cities, its capabilities are considerably more restricted than a "driving as humans" robot. In this dissertation we will focus on developing a robot capable of moving freely and choosing autonomously its motion in order to reach the desired destination while avoiding collisions.

Driving rules We seek to have a robot with driving capabilities "as good as a human", however we will not enforce the respect of driving rules imposed to humans. We disregard them because they are specific to each world region and because it is not clear that they provide advantages when applied to robotic systems (other than having the robot behave "more like humans"). Including notions of safety, passenger comfort and efficiency should be enough to obtain a behaviour "interpretable by humans" without the need to adding specific ad-hoc rules. If needed respecting driving rules could be added as an additional task of the system that we will describe.

Since the task of perception interacts with other modules we will discuss some of them. We will not discuss mechanical aspects or any issue related to engineering, energy sources, economic model or legal issues. All of this topic are of fundamental importance but beyond the scope of this dissertation.

Through this dissertation we will provide a discourse around the following thesis:

"Perception for driverless vehicles in urban roads can be done without specific infrastructure"

This work is distributed in the following parts:

- Chapter 2 discusses the core issue of safety and how it influences the design of the perception, planning and control algorithms.
- Chapter 3 provides a constructive overview of the techniques related to perception for mobile robots.
- Chapter 4 extends these notions to the particular case of driverless vehicles in urban environments and we describe in detail the new perception system proposed.

- Chapter 5 describes the integration of the proposed perception method with state of the art planning and control algorithms in order to drive an electric vehicle. We present some experimental results.
- Chapter 6 explores how perception can be extended by the exchange of information between nearby vehicles. We propose a method on how to achieve this.

Finally chapter 7 provides some conclusions and draw lines for future research.

The core contribution of this work is the design of a perception system that allows a robot to drive move in an urban environment without requiring a detailed map, installing infrastructure or a previous visit.

The proposed solution uses a laser scanner as the main sensor, complemented with some image processing for road detection, fusion with the odometry, inertial sensors for robustness and GPS usage for global localisation. The proposed approach makes no strong assumptions on the geometry of the observed objects (static or moving), but assumes a locally planar road (due to laser scanner limitations).

Chapter 2

Achievable safety of driverless vehicles

> Salus populi suprema lex esto.
> The safety of the people is the supreme law.
> Cicero

The main objective of a driverless vehicle is to move from a point A to a point B in the city. It has to do so while ensuring the safety of the passengers and the vulnerable entities in the surroundings. Surprisingly, *the safety of mobile robots still is a fuzzy topic*. Since most classic robot systems evolve in uninhabited areas or in highly controlled environments the issue of ensuring safety while moving in the city appears as a novel problem to be addressed.

In this chapter we will show that safety depends on the design of the perception-planning-control trio. We will discuss the safety guarantees that can be provided when the robot evolves using its on board sensors, how to achieve this guarantees and which constraints are then imposed over the perception, planning and control modules. This constraints will be used in the design and implementation of the driverless vehicle described in the chapter 5.

Safety relates to perception and perception relates to safety

A vehicle do harm when the wrong decisions are taken. Since the decisions are based on the information provided by the perception module, understanding how to design a harmless vehicle will help us understand what does the perception module needs to do.

We argue that most of existing works do not provide adequate safety guarantees for a driverless ground vehicle evolving in environments such as a city. We will also show that the usual notion of safety based on inevitable collision states (see definition 2.5) does not apply when the vehicle plans based on its sensors input. We argue that in such

a relevant case it is only possible to guarantee that the robot will not harm by action, we cannot guarantee to be harmless through inaction.

The fundamental question underlying this chapter approach is: which the necessary information that the robot requires to drive ? Driving from point A to B in a world free of obstacles is easy. As we shall see, simply reacting (avoiding) to any apparition in the sensors range is not enough. Then, which is the necessary minimum ?

During this chapter we will explain and argue an approach supporting the following asseverations:

- Safety does not depend on the planning method used, but does depend on solving the adequate planning problem,
- The perception module needs to provide a conservative prediction of the future,
- The main output of the perception module is a function estimating the harm of hypothetical future states of the vehicle.

In section 2.1 we provide a definition of safety and the criteria required to evaluate if an approach is safe or not. Section 2.2 defines the notions of uncertain and incomplete world model. Section 2.3 explains the effect of uncertainty on safe planning and how to manage it. In section 2.4 we analyze the safety of a vehicle planning with an incomplete view of the world. Section 2.6 discusses the effects of extra sensorial data on the safety guarantees. Finally section 2.8 offers some conclusions.

2.1 Safety

Mobiles robots, and especially driverless ground vehicles, are capable of harming themselves and their surrounding environment (pedestrians, animals, other vehicles, the streets' infrastructure, etc...). Motion safety relates on how to mitigate and, if possible, avoid harmful contact with the environment.

Definition 2.1: ***(Safe motion)*** *A robot is said to be safe if it can be guaranteed that its motion will not harm himself or its surroundings.*

For ground vehicles harm can be created in three different ways:

- Traversing unfit terrain at inadequate speeds can harm the vehicle itself (running into water, holes, curbs, uneven or too inclined ground, etc... see figure 2.1),
- Colliding with objects (pedestrians, animals, other vehicles, walls, poles, trash can, etc...),
- Pushing the vehicle out of its range of dynamic stability (overturn).

Figure 2.1: Example of harm without collision

In any case harm will raise from setting the vehicle in the wrong state (position, speed, etc...) at a given time. In order to avoid harm we need to define the state of the vehicle in time. A sequence of states in time is named a *trajectory.* A trajectory is said *feasible* if it exists a sequence of commands that allows the robot to reach such states in time.

Definition 2.2: ***(Vehicle motion planning problem)*** *Given an initial state $x(t_0)$, a cost measure $c(\pi)$ over any trajectory π and a world model w, define a feasible sequence of states in time $\hat{\pi}$ that will generate a safe motion while minimizing the cost $c(\hat{\pi})$.*

The vehicle motion planning problem is an optimization problem with constraints. In our setup, the cost function $c(\pi)$ will measure how much we progress on the path defined during route planning. The *world model* is a representation of the world built from input data available to the robot (sensors, a priori information, communications, etc...). In chapters 3 and 4 we explain how to build the world model. Since the planning relates to defining future states of the vehicle, the world model needs to provide information about the future state of the world.

The safety of the motion does not only depends on the defined trajectory, but also on how the world model is built and used, and how the vehicle executes in the real world this trajectory. The safe motion of the robot is a propriety of the perception-planning-control trio.

2.1.1 Safety criteria

In order to analyze the safety of robotic systems three criteria have been proposed [19]:

1. Considering the motion of the robot
2. Considering the motion of the surrounding environment
3. Considering an infinite time horizon

The first criterion indicates that the limitation of the robot motion needs to be considered in the motion planning (maximum acceleration/deceleration, maximum speed, adherence constraints, etc...).

The second criterion indicates that the presence of moving and static obstacles around the vehicle has to be taken into account. The future position of surrounding objects needs to be considered in order to predict the free space available.

The last criterion indicates that, since the relative motion of the robot cannot be changed arbitrarily (first two criteria) at any point in the time the vehicle could be in course to an inevitable collision. Without any particular assumption on the real world checking collisions over a finite time horizon cannot guarantee that the robot is not in an inevitable collision state [20]. It should be noted however that in many scenarios it is possible to define a finite set of verifications (computations) that will guarantee harmless behaviour over an infinite time horizon [21, 22, 23].

When evolving in a environment populated by static and moving obstacles all of the traditional approaches to robot motion (nearest diagram, dynamic windows, velocity obstacles) fail to respect such criteria, and thus, fail to ensure harmless motion [19]. This occurs even when supposing a perfect knowledge of the present and future of the world.

Also, the traditional notion of time to collision is not an useful measure to verify safety under the proposed criteria. Since the manoeuvrability of the vehicle depends on it current velocity, two cars approaching at 15 [m/s] is not the same situation as approaching a stopped vehicle at 30 [m/s]. The driverless vehicle needs to analyze situations in a reference frame attached to the ground and not relative to the robot itself.

As an example we can analyze the system proposed in [24]. They suppose that the world model built with their sensors is complete and that it correctly classify static obstacles and pedestrian. With each new sensor measurement the world model is updated and the vehicle updates its planned trajectory. The trajectory is generated from a set of predefined kinodynamically feasible control commands sequences. For each element of this set the collision with the world model is checked and a distance between the final state and the desired future state is measured. Trajectories leading to collision are disregarded and the sequence of commands leading nearest to the desired state (the goal) is selected. While announced as an approach for “secure driving in dynamic environments”, this method fails to consider the future motion of the pedestrians and provides no guarantee that in the next iteration at least one trajectory free of collision exists. Since it cannot guarantee that no collision will occur, it fails to respect definition 2.1 and should be considered as non safe.

2.2 Building a world model from sensors

In order to solve the robot motion problem (see definition 2.2) we need to be able to evaluate two values. Let $x(t)$ indicate a state of the vehicle and π be a candidate trajectory starting from the initial state $x(t_0)$. Then $c(\pi)$ is the value indicating how well

π allow us to reach the desired goal, it is used to search the most desirable trajectory. The second value $h(x(t), w(t)) \in [0,1]$ evaluates the harm caused by setting the vehicle in the state $x(t)$ given a prediction of the world state at time t.

Definition 2.3: ***(Harm function)*** *Given the state of the vehicle $x(t)$ and an estimation of the world state $w(t)$ the function $h(x(t), w(t)) \in [0,1]$ estimates the harm that would be caused by setting the vehicle in such state $x(t)$. Value zero indicate that no harm is caused, value one indicate that a very important harm is caused. Intermediary values provide a scale from minor to significant harm.*

The notion of harm and its mapping to the function $h(x(t), w(t))$ are highly application dependent. Passing over a car may be fatal for a car, but harmless for a tank. This function h allows the motion planner to search for safe trajectories.

Based on the definition 2.1 we expect to be able to guarantee for a planned trajectory $\pi = \{ (x(t_0), t_0), (x(t_1), t_1), \ldots, (x(\infty), \infty) \}$ that $h(x(t_i), w(t_i)) = 0 \; \forall (x(t_i), t_i) \in \pi$.

Definition 2.4: ***(Traversable space in time)*** *A point p in space is considered traversable at time t if for any vehicle state $x(t)$ that covers the point p the harm value $h(x(t), w(t))$ is zero.*

Thus, one of the main concerns of a perception module, is estimating and predicting the traversable space in time.

A usual notion used when concerned on safe trajectory planning is the concept of *inevitable collisions state* (ICS).

Definition 2.5: ***(Inevitable collision state)*** *A state is considered in inevitable collision if all trajectories resulting from the application of any the possible commands sequence to this state, lead to a collision.*

In our case, since we are also concerned on the good of the vehicle itself (and not only on collisions), we will should speak of "inevitable damage state". We will loosely use both terms as exchangeable. Strictly, we are concerned on avoiding reaching states where $h(x(t), w(t)) \neq 0$ (which, depending on the application, may include more that just collision states). In order to ensure safety the trajectory planner should, at least, only produce trajectories are free of inevitable damage states.

2.2.1 Incompleteness

When using on board sensors the robot is fundamentally limited to access only an incomplete and uncertain representation of the world. This means that some surrounding areas will be left unobserved (due to sensing range or occlusions, see figure 2.2), that the relative position of obstacles will be not necessarily known with precision (distribution of probability of collision over the space) and that the uncertainty in the prediction of future positions of moving obstacles will grow monotonously (since we do not know their future state). Beyond a certain point in the future nothing can be told about the traversable space.

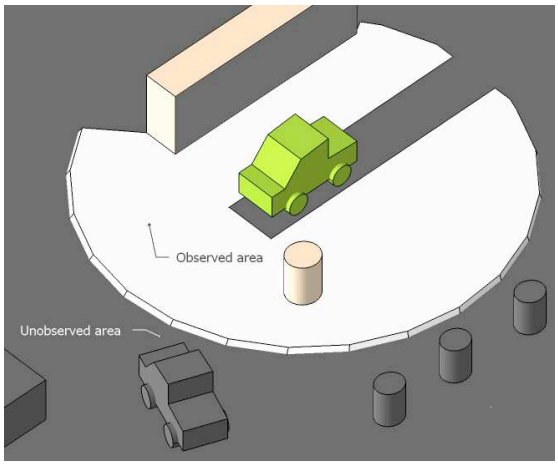


Figure 2.2: Illustration of observed and unobserved areas

Definition 2.6: ***(Incomplete world model)*** *A world model w is considered incomplete if $h(x(t), w(t))$ is not defined for every possible pair $(t, x(t))$.*

Since h is not defined everywhere, an incomplete world model can lead to incomplete or suboptimal plans.

2.2.2 Uncertainty

Sensors are corrupted by noise, thus relative distance measures to static obstacles will be uncertain. Also the state estimate of moving obstacles will be noisy. Even if a deterministic model existed for the moving obstacles, the initial uncertainty in the current state estimate will propagate in time. In many applications stochastic models are used to predict the motion of moving obstacles (e.g. motion of pedestrians or drivers) and thus position uncertainty grows in time. In an uncertain world model the traversability of a point p at an instant t becomes a probability value.

Definition 2.7: ***(Uncertain world model)*** A world model w is considered uncertain when $h(x(t), w(t))$ is not available but the probability distribution $P(h(x(t), w(t)))$ is.

2.3 Uncertain world

Let us suppose by now that we have access to a complete but uncertain world model. As previously mentioned, this means that a given vehicle state in time is associated to a probability distribution of the harm value $P(h(x(t), w(t)))$. Since we seek for harmless motion (where $h(x(t), w(t)) = 0$) we are in particular interested in the value $P(h(x(t), w(t)) = 0)$.

If any moving obstacle exists in the world it future position uncertainty will grow monotonously in time. Supposing that the moving obstacles have a reachable space that cover ours (think of pedestrian and human drivers), some point in the future the whole world model will become uninformative: any traversable area may have become non traversable. What sense does it makes then to verify that no collision will occur up to infinity? Can we define an inevitable collision state in an uncertain world?

2.3.1 Cost function

a usual approach to safety consist on including it as part of the cost function $c(\pi)$. On an uncertain world model one could define such a cost component using the integral of the vehicle trajectory over the probability distribution (in a spirit similar to [25]), as described in the equation 2.1.

$$safety\_cost = \sum_{(x(t_i),t_i)\in\pi} (P(h(x(t_i),w(t_i)) = 0) - 0.5) \tag{2.1}$$

Since defining a collision free trajectory up to infinity does not make sense, let us search for the trajectory to infinity with the lowest probability of collision. This approach is not suitable for multiple reasons. First we should remember that the primary aim of motion planning is reaching a specific point. The cost function will trade off collision risk with getting nearer to the goal, which undesirable if ensuring safety is a must. On the other hand, if collision risk has a very important height compared to reaching the goal, then generated trajectories will be unsatisfactory. When observing an empty area the safest trajectory is to stop as soon as possible. Using directly the cost function neither provides satisfactory safety nor satisfactory trajectories to reach the goal.

2.3.2 Probability threshold

Why should we use a conservative world model ?

In order to provide guarantees on safety, the probability distribution needs to be thresholded in order to define a binary function over space time (in a spirit similar to [26], for instance). Doing so is equivalent to providing a conservative estimation of the traversable space (see figure 2.3). We want to ensure that if some area in space time is predicted as traversable, it will be such in reality. By thresholding we ensure that the vehicle will always evolve in areas were a collision is considered highly unlikely and that the areas where a collision could occur will be left unvisited.

Definition 2.8: ***(Conservative world model)*** *A world model w' is considered a conservative approximation of w if and only if $h(x(t),w'(t)) = 0 \Rightarrow h(x(t),w(t)) = 0\ \forall x(t) \forall t$.*

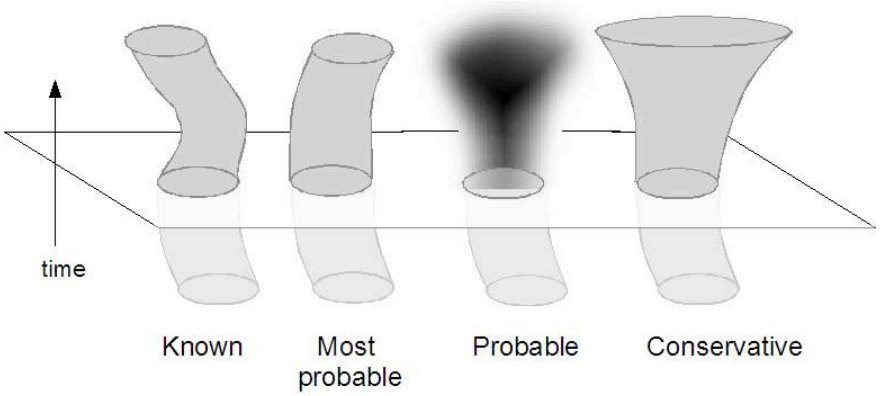


Figure 2.3: Predicting non traversable space in time. The region below the plane was observed, the region on top is predicted

Thresholding $P(h(x(t),w(t))=0)$ allows to convert an uncertain world model into a certain world model by using the function h' described in equation 2.2

$$h'(h((x(t),w(t)))=h(x(t),w'(t))=\begin{cases}0,\ if\ P(h(x(t),w(t))=0)<p_{threshold},\\ argmax\ P(h(x(t),w(t))),\ otherwise.\end{cases} \tag{2.2}$$

We expect h' (and thus, the selected threshold) to provide a conservative approximation of the real world (inaccessible through sensors).

When using a threshold over an incomplete and uncertain world model, it is likely that, over time, the predicted world collapses into a completely non traversable area. In this context the notion of inevitable collision states does not apply. This issue will be discussed in section 2.4.

Pedagogic example

Why not using a conservative world model is a bad idea ?

The notion of using a conservative approximation for may cause some reluctance to some readers. Conservative approximations seems, at first sight, to generate an undesirable behaviour. In section 2.7 we will analyze in detail some relevant scenarios in order to clarify this point.

The figure 2.4 presents a simple scenario that illustrate why not being conservative is inadequate for our application. In this pedagogic example we suppose that at each instant we have perfect knowledge of the present situation. Our vehicle movement

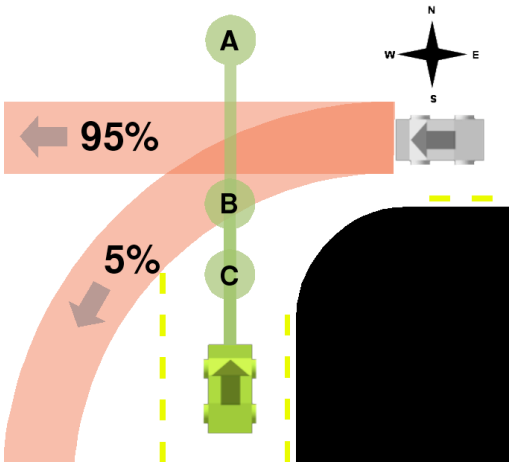


Figure 2.4: Example used to illustrate why conservative prediction are needed. Vehicle on the left may go straight or turn, with known a priori probabilities. The driverless vehicle on the bottom wants to go to the top of the image. Should it adapt its speed to be able to stop at point A, B or C?

is limited to a straight line upwards, we are only interested on controlling the speed profile. As shown in the figure, there is a second vehicle in the scene. In this artificial scenario we accept as a fact that the second vehicle can only follow two specific paths, and that the probability of it choosing one or another is known before hand. We do not know however its future speed profile, it may accelerate or decelerate in an unknown fashion.

Then, as presented in the figure 2.4, we need to decide the planed speed for our vehicle. We can choose a speed that allows to stop the vehicle at points A, B or C. Stopping at point C means having a lower speed that stopping at point A.

Choosing to stop at point A is not a viable solution since this would mean to disregard the fact that we do not know the speed profile of the incoming vehicle. Simply passing through the intersection may put the second vehicle in an inevitable collision state. From its perspective we will have entered abruptly into the intersection, it was not able to slow down, we provoked a collision with him. This is not a safe choice, since we can provide no guarantee about the harm caused.

Choosing to stop at point B seems to be a good trade off between not slowing down too much and still avoiding collision in most of the cases. The problem is that, when the incoming vehicle chooses to turn, given that our speed only allows to stop at B and not before, we will not be able to slowdown in time to avoid the collision. *Choosing B guarantees to have a collision* 5% of the times. Even if this was 1% we do have to remember that we aim for large scale deployment. In a city full of driverless vehicle running all day long, having one collision per each 100 intersection crossing equals to have hundreds of accidents per day.

Choosing C is the conservative option that will allow us to avoid collisions in any considered situation. Adapting the speed so we can stop at C does not mean that we will always stop at C. If the situation allow it, we will simply pass through it (because the second vehicle went straight or passed on the collision area near C before we reached it). Choosing C is to acknowledge that we have to consider all of the possible cases and not just the most probable ones.

2.4 Incomplete world

Since the robot knows only a fraction of the world it is probable that it has not enough information to define a single definitive plan from its current state to the goal. Even if it did, as new information is acquired a better plan could be generated. This leads to the notion of *partial motion planning* [27, 15]. The denomination "partial" indicates that, unlike the usual approaches, the plan does not reach the goal (but, hopefully, leads towards it). Using a best effort approach the robot initially computes a partial plan to reach the goal. As the robots moves a new plan is computed to extend or enhance the previous one.

The criteria mentioned in section 2.1 imply that while the plan may be "partial" in the space dimension, it needs to be collision free over an infinite time horizon. To do so it is only necessary to ensure that every state of the plan is collision free and that the last state of the partial plan is not an inevitable collision state [28]. Verifying if a planned state will generate or not an inevitable collision is not a trivial problem, since we need to do verification over an infinite time horizon.

2.4.1 Static world

When the robot evolves in a static environment using a set of stop trajectories will provide safety, since we know that the vehicle will be able to stop before colliding and then no collision can occur (see figures 2.5 and 2.6). For this case, computing over a finite time horizon provides a guarantee over an infinite time horizon. Considering that a wall could appear at the frontier of the observed space, safety in an incomplete world is guaranteed if the unobserved space is considered as non traversable [21].

Swerving Instead of stopping [29] proposed swerving the expected static obstacles, thus allowing higher speeds with the same sensors range. This approach makes strong assumptions on the maximum size and density of static obstacles, thus it is not suitable for highway or city like environments since the presence of a traffic jam (total road blockage) on a single direction road would generate a collision.

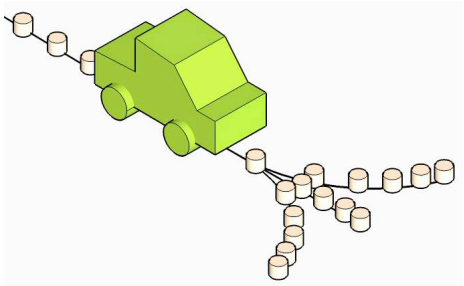


Figure 2.5: Trajectories resulting from some stopping commands for a vehicle. Dots are equidistant in time

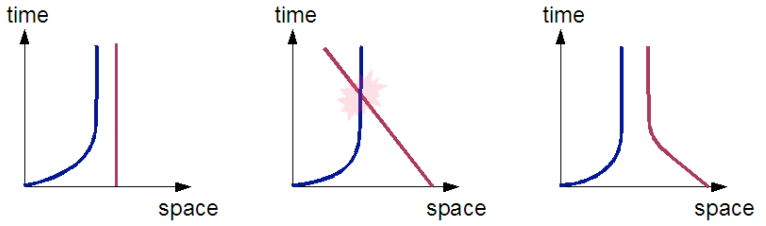
2.4.2 Dynamic world

When the environment may include moving objects using stopping commands cannot, by itself, ensure avoiding collisions.

Imitating manoeuvres Ideally one would desire to have access to the present and future position of all the surrounding obstacles. When this information is available the notion of "imitating manoeuvre" has been proposed as a collision free trajectory over an infinite time [22]. However this approach is brittle since imitating a single moving obstacle could lead to colliding with another obstacle, and thus is not desirable in cluttered environments with arbitrary obstacles motions.

Finite time horizon Knowing the motion from now to infinity of all the moving obstacles does not provide by itself a tractable way of detecting inevitable collisions states. Even if the vehicle can stop without having a collision with any moving or static obstacle, nothing prevents that a few seconds later a moving obstacle collide with the stopped vehicle. Defining a time horizon beyond which the vehicle is guaranteed to be collision free is an open problem in the general case. If it is possible to define a time t_l in the future beyond which no moving obstacle will approach the stopped vehicle, then verifying a collision of the stopping trajectory until the vehicle stops and beyond t_l is enough to ensure safety. This is approach used by [23] where it is supposed that the world is only populated by static obstacles and driverless vehicles, and the partial plan of every vehicle is perfectly known. At each re-planning step the trajectories of every vehicle are verified to be collision free with the previously planned ones and the re-planned ones. Since both the previous and the new partial plans finish with the vehicles stopped, it is possible to guarantee that the system is safe.

Best effort Unless proof of the contrary an moving obstacles may be present at the frontier of the observed space. If we only have a partial knowledge of the surrounding environment a conservative world model needs to suppose the presence of moving



(a) stop in front of wall

(b) stop in front of moving obstacle

(c) both moving obstacles apply the same safety logic

Figure 2.6: Multiple scenarios of two obstacles in one space dimension. Blue and red lines are obstacles in time

obstacles at the frontiers of the unobserved area. Without information about the non observed moving obstacle we cannot compute an avoidance trajectory. The best effort consists then in ensuring that the vehicle is able to stop without colliding. By itself, this does not ensure avoiding collisions, a moving obstacle could collide the vehicle immediately after it stops. Using stopping commands in this case can ensure that we will not harm by action, but only by inaction ("car on a railroad" scenario). This is the approach used in [28, 12].

In this approach we need to make assumptions about the moving objects that will appear, we need a model. This model predicts the possible (not the likely) presence of obstacles in time. The simplest possible model consist on "any obstacle, anywhere, in any direction" but with a bounded maximum speed. Depending on the specific application the model can become arbitrarily complex. For instance, in an urban scenario it could be reasonable to consider that no vehicle drives against the defined circulation directions. Doing so limits the possible appearance of moving obstacles and thus allows for planning higher speeds within the incomplete world model.

Theorem 2.1: *If every moving obstacle uses a conservative world model and plans to stops before entering in a harmful state then no collision would occur and the overall system can be considered safe (see figure 2.6).*

PROOF: Let a and b be sensing moving objects. Both objects have a conservative world model with respect to the real world. The policy indicates that, since the world models are conservative, as soon as a detects b it has a trajectory allowing it to stop without colliding b or any other object. Conversely as soon as b detects a it already has a trajectory allowing to stop without colliding. Both a and b can stop without colliding, and the same applies to any couple of moving objects in the world. By induction since every moving object can stop without colliding, when one object is stopped, it is guaranteed that no other object will collide it, and thus it is a safe state over an infinite time horizon. ■

If we ensure that the vehicle is able to stop given the possible appearance of an obstacle with a given maximum speed v_m and an obstacle appears coming faster than expected,

then the vehicle will not be able to stop before the collision. Depending on the manoeuvrability of the vehicle, the clutter of the scene and the speed of the re-planning algorithm the vehicle may or may not be able to avoid the obstacles. *The safety guarantees are as good as the model* (safety depends not only on the planning algorithm but also on perception and control modules). If the predicted traversable space is not available, the guarantees that the motion of the vehicle will be harmless are lost.

2.5 Safe planning

Based on the discussion of the previous sections we obtain the following definition.

Definition 2.9: ***(Safe trajectory for driverless cars)*** *In an incomplete and uncertain world model of a dynamic environment a trajectory is considered safe if:*

- *Each state is collision free with respect to a conservative prediction of the traversable space in time,*
- *The sequence of states respects the vehicle capabilities,*
- *Its last state has speed zero.*

With the information available and without doing strong assumptions on the non observed areas, this is approach will ensure that no harm by action is done.

Please note that while the safe trajectory leads to stopping, it does not mean that the vehicle will necessarily stop. As the vehicle starts moving, the new sensors measurements will extend its world model. This allows it to replace the current trajectory (in execution) with a new safe trajectory. By doing so repeatedly the vehicle will drive continuously towards the goal and will not stop unless necessary (blocking obstacle). When using this approach the vehicle will automatically adapt its speed and behaviour to the surrounding environment, considering both the observed and unobserved space. It will drive slowly in cluttered and uncertain situation, and faster on uncluttered certain environments.

2.5.1 Unexpected events

Whatever are the passive or active measures taken, in the real world accidents will occur (e.g. “falling crane” scenario). Even using a conservative approach unexpected events will happen. As previously mentioned, if an area predicted as traversable is discovered as non traversable the harmless motion guarantee is lost. Maybe the vehicle will be able to avoid the collision. If not, it is important to be able to consider the cost of a collision. Crashing towards a trash can creates less harm than crashing a wall. Crashing an animal is better tolerated than crashing persons. If a collision becomes inevitable, it is desirable that the robot does not treat each possible smash equally.

This means that the perception system needs at least to localize the vehicle with respect to the planning goal (for planning) and to the plan currently in execution (for control purposes), to estimate the traversable space in time and to constantly monitor the cost of possible collisions.

When planning over a conservative world model, the planner expects that the robot will follow exactly the computed trajectory. Failing to do so would nullify any safety guarantee. This imply that the control module needs to provide a predictable bound on its tracking error. This bound is integrated in the planning stage to ensure that the trajectory is safe even with errors on the control. If the control unexpectedly fails to respect the predicted bound leave the robot in a situation equivalent to a violation of the conservative property of the world model, and a collision may or may not occur.

2.5.2 Safe perception-planning-control

Most of the previous works in the field will fail to respect the definition 2.9. For instance [26] proposed a planning method in dynamic environments able to deal with an incomplete and uncertain world model. However they do not take into account the unobserved area of the environment (non conservative estimation), and provide no guarantee that the vehicle will not collide if their probabilistic re-planning method fails.

In [30] a perception-planning duo was proposed to move in an incomplete and uncertain world model. The approach is similar to the one described here. However their algorithm restricts the movement of obstacle to constant speed vectors. This approximation is non conservative over the obstacles considered in their work (cars intersection) and can be considered a violation of the safety criteria and the definition 2.9.

In chapter 4 (and sections 5.3.1, 5.3.2) we present a solution consistent with definition 2.9 (solution initially presented in [12]). It is explained how the vehicle can efficiently estimate its movement, map static obstacles, detect and track moving obstacles, plan safe trajectories and execute them. All of it was integrated in a real world vehicle as a proof of concept prototype.

More efficient and effective algorithms may be designed. Application specific models may provide more information about the future. Better sensors will provide larger observation areas and less uncertainty, allowing higher speeds. However we believe that safety cannot be enhanced beyond the discussed limits. Real world safety is bounded by how good is the robot at estimating the possible futures.

2.6 Complemented world

In the previous section we discussed safe motion of a robot using its on board sensors, a model of its sensor, a model of itself and a model of the existing moving objects. For the design of a mobile robot it is relevant to understand how other sources of information may affect the safety. When the robot completes the information observable from its point of view with other sources, we say it estimates a “complemented world model”.

Are other sources of information helpful ?

2.6.1 Maps

There is the usual belief that high precision map will help robots navigation. These maps are of little use when concerned about safety. In populated environments such as the cities, if an area was seen as traversable a few hours ago, little can be said about if it is currently obstructed or not. Maps of the static environment are not able to provide reliable information of the future. Maps can be built to model the usual behaviour of moving objects in a region [31]. This information can then be used to provide tighter predictions of the moving objects. Using this maps is still a delicate issue. Being built from observations it is hard to assess their completeness. Failing to predict a possible path for an obstacle, will lead to violations of the model and thus to a potential collision. Even worse, in the city the behaviour of pedestrians (for instance) not only depends on space but also on time (e.g. the Sunday street market) making it less likely to have a complete and reliable map of moving objects paths in space time.

2.6.2 Fixed path

A possible future for driverless vehicles in urban environment is the deployment of "immaterial tramways" [13, 32, 33]. Restricting the movement of the vehicle to a fixed path seems to be perceived by humans as a safer option.

In this configuration the planning is mainly concerned with speed control, a reduced set of commands ease the planning computation. However the needs for perception (building the world model) remain the same (with respect to range coverage or prediction capabilities). Knowing the path may provide a thin enhancement in the localization computation cost and in the collision cost estimate.

With a fixed path the exact same logic discussed in this dissertation applies. The safety issue is not relevantly modified.

2.6.3 Traffic rules

The use of traffic rules between humans seems to be a factor enhancing the safety on the roads. Supposing that other vehicles respect the rules, constraint their possible future movements and thus it is useful information to predict the traversable space. The use of traffic rules per se does not enhance the safety (since it depends on other vehicles respect of such rules), however it allows the vehicle to be less cautious in some situations (e.g. not slowing down at an intersection, because we have the priority).

2.6.4 Vehicle to vehicle communications

The use of vehicle to vehicle communication could enhance the safety in multiple ways. First, more measurements of the environment provides more information and thus probably larger observed areas and less uncertainty. Second, when the surrounding moving obstacles are driverless vehicles too, it would be possible to obtain their current plans, thus providing much tighter bounds than a naive worst case model [23].

If should be noted that the use of vehicle to vehicle communications requires to solve a relative positioning problem between the communicating vehicles. How to solve this problem in an unmodified city remains an open question. Even if a reasonable solution is provided, the relative positions will have a certain degree of uncertainty that will then be propagated over the transmitted data, reducing the benefits of the exchange. We discuss this issue in more details in chapter 6.

2.7 Example scenarios

When respecting a safety constraint the behaviour of the vehicle depends on its perception capabilities and model. Previous works discussed the need for a wide and dependable sensory coverage of the surroundings of the vehicle in order to provide enough information to avoid collisions [34, 35]. In [36, 37] the required detection ranges for typical roads manoeuvres are discussed (u-turns, overtakes, intersection crossing). It is relevant then to understand which perception capabilities relates to which desired behaviour. Let us illustrate this relation through some particular examples.

2.7.1 Open field

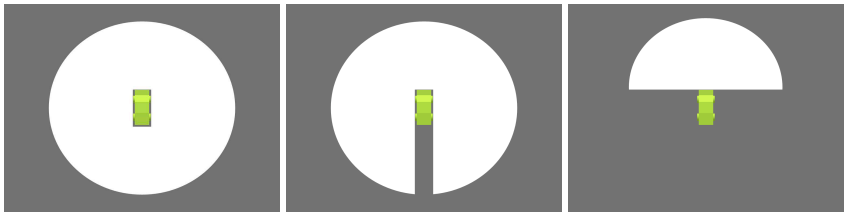
Static world

The simplest non trivial scenario consists of a single vehicle moving over a large traversable surface where a single static obstacle exists.

Let us consider a maximal acceleration and deceleration rate a_{max} of 3 $[m/s^2]$ (which stays in the comfort zone of human drivers [38]). In urban environment, the maximal legal speed v_{max} is around 14 $[m/s]$. Then the maximum stopping distance $d_{stop_{max}}$ is

$$d_{stop_{max}} = t_d \cdot v_{max} + \frac{v_{max}^2}{2 \cdot a_{max}} \tag{2.3}$$

where t_d is the delay between an event and the reaction of the robot. For simplicity we will consider $t_d = 0$ $[s]$ in this analysis. In our example $d_{stop_{max}} = 33$ $[m]$. Then in order to attain safely the designed speed v_{max} the sensors range should reach at least 33 $[m]$



(a) Full 360° visibility (b) All but rear visibility (c) Front visibility

Figure 2.7: Top of view of the open field scenario. White indicates observed space, gray indicates non observed space.

in front of the vehicle. Shorted coverage ranges will limit the maximum safe speed of the vehicle.

The different ranges of visibility illustrated in figure 2.7, relate with more or less limited capabilities of the vehicle. For instance, without a rear view the robot is not able to run backwards (for parking manoeuvres for instance).

Dynamic world

If we consider a scenario where no moving obstacles exist and that the vehicle cannot move to the side, then having only a front view is equivalent to an all but rear view. However if arbitrary moving obstacles may be present, a front view is not enough to provide safe turns since an obstacle may be moving alongside the vehicle. Turning with an obstacle moving at the side would generate an unforeseen collision.

In a static world, supposing the presence of static obstacles anywhere in the non observed area provides a conservative approximation (respecting the definition 2.8). In a dynamic world, moving objects may be present in the open field. A conservative approximation consist on supposing that at the frontier of the observed traversable space lay a wall of moving obstacles approaching aggressively. Supposing that the vehicle move in a straight trajectory, the worst case to analyze consist on a moving objects moving frontally. We come back then to the scenario presented in figure 2.6, from stopping in front of a wall, to stopping in front of a moving obstacle.

In this example we suppose that the current vehicle speed is v_0 and that other moving obstacles have a maximum speed limited by law at v_{max}. Then, we require the vehicle to be able to stop before colliding with the suspected moving obstacle, as illustrated in figure 2.8. Using equation 2.3 it can be shown that the required sensors observation range $d_{sensor\ range}(v_0)$ would be

$$d_{sensor\ range}(v_0) = t_d \cdot v_0 + \frac{v_0^2}{2 \cdot a_{max}} + \frac{v_0 \cdot v_{max}}{a_{max}} \tag{2.4}$$

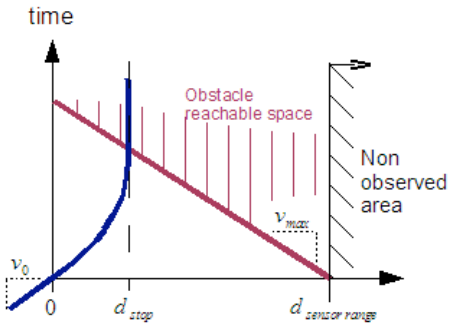
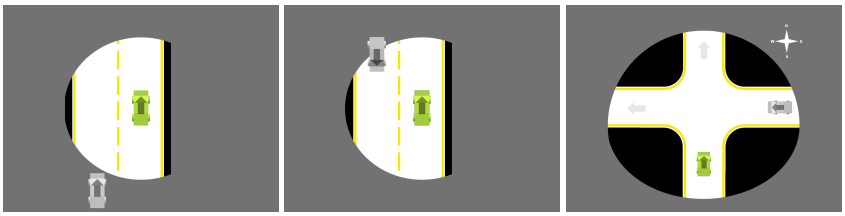


Figure 2.8: Stopping in front of a moving obstacle that appears at the frontier of the observed space



(a) Highway (b) Two directions street (c) Simple intersection

Figure 2.9: Top of view of the highway scenario. White indicates observed traversable space, black indicates observed non traversable areas, gray indicates non observed space, yellow lines indicate lanes

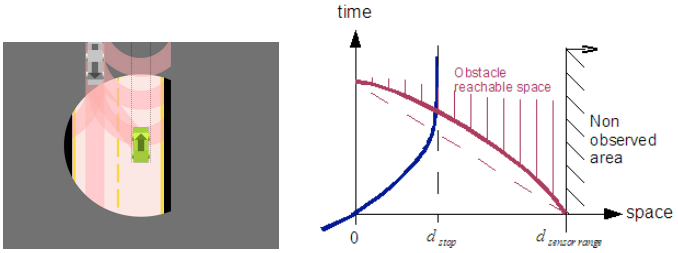
This means that in a totally conservative approach,the robot needs are sensing range of 100 $[m]$ to evolve at maximum speed $v_0 = v_{max}$ in a populated urban environment. The equation 2.4 also provides the inverse relation, given a visibility range (limited by sensors or occlusions), which is the maximum safe speed.

In chapter 4 we discuss how the precision of the sensors (besides their detection range) also influences the final behaviour of the vehicle.

2.7.2 Streets

In contrast to the open field scenario, when in the city the regions where vehicles can circulate are constrained. The apparition of vehicles is then limited to specific regions of the visibility frontier, and the possible path to avoid obstacles is limited. In particular, when circulating on the roads cars are forced to have closer interactions than in the previous scenario, where the vehicle could run wherever it pleases.

Figure 2.9 illustrates some common scenes in which vehicles interact.



(a) An unseen car could come from the opposite lane

(b) Space time diagram of the obstacle and the robot

Figure 2.10: In a two lanes road, a non seen car may aggressively approach our lane. Pink areas indicate reachable space of the moving obstacle, some paths are highlighted, see also caption of figure 2.9

Two directions street

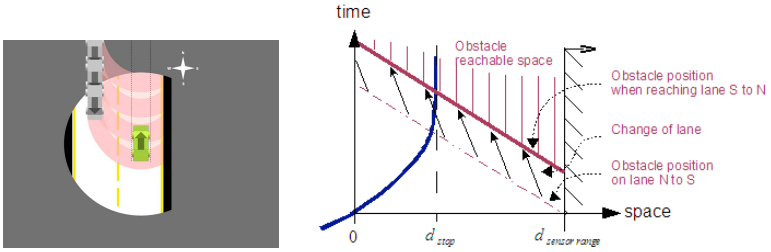
For analysis purposes a highway is similar to a street where every lane goes in the same direction (since lanes with different directions are usually separated by a barrier). We will focus on the more generic case of a two direction street.

In the urban context we expect different classes of moving objects to have different capabilities. In the limits of the detected roads we expect objects to appear with maximum speed v_{max}^{car} while on the sideways objects have a different limited speed $v_{max}^{pedestrian}$. Different hypotheses on where and how different moving may appear radically affects the final behaviour of the vehicle, and are highly application dependent.

Possibly the most dangerous situation a car has typically to manage is crossing a car in a two directions street. A slight change in the trajectory of the approaching vehicle could lead to a high energy frontal collision. Let us analyze this particular case. For analysis purposes we will disregard the presence of pedestrians (which are comparable to small cars with lower maximum speed passing on the sides).

There are two cases to analyze, when no other car is detected and when one car is coming on the reverse lane.

If no car is seen, then a conservative approximation of the future will suppose that a car is about to appear in the opposite lane. Even worse, this car is moving towards our lane with maximum speed v_{max}. This worst case scenario, illustrated in figure 2.10 is essentially a variant of the open field case previously discussed. While being a two dimensional scenario, we can analyze the "shortest possible time to collision" to the planned trajectory, supposing that the robot stays on its current lane. Said in other words, we define the uni-dimensional curve defining "the shortest time in which the moving obstacle can reach any point along my lane". As shown on figure 2.10 this curve will always be superior in time than the frontal case discussed on section 2.7.1, the same logic applies.



(a) A car pass on the oposite lane (b) Space time diagram of the obstacle and the robot

Figure 2.11: In a two lanes road, a car approaches. At any given moment it could take a sudden turn. See also caption of figure 2.10

When a vehicle approaches us, at some point it will be detected and tracked. Then comes the question of the model used for prediction. The worst case scenario would consider an aggressive vehicle, we would then fall into case a similar, but not identical, to the previous one. It takes time to pass from one lane to another. For analysis purpose we consider a slightly less pessimistic case, let us suppose that the approaching vehicle could at any time make a sudden turn and block our lane, as illustrated in figure 2.11. In the space-time graph we present the moving obstacle as a particle, to consider its geometry it suffices to overlap the curve of each extreme of the object and consider the worst case (shortest time to collision).

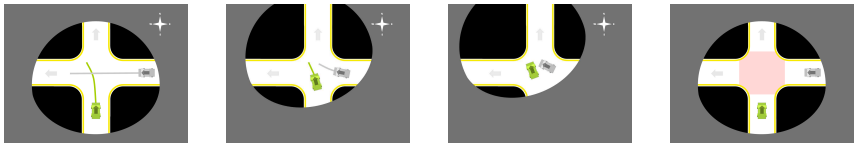
Analyzing the presented case we see that robot will decelerate considerably when crossing another vehicle in a side lane, *but it will not stop*. In the proposed approach, a vehicle only stops if it cannot find any other option. Having a vehicle slowing down notoriously each time it crosses another one seems disappointing, but is the correct action to take when you imagine that all the other vehicles may suddenly invade your lane.

For a more "humans like behaviour" we propose to use an hypothesis on the perception range of other vehicles. If we detect a vehicle it means that it probably can see us too. Under a certain, predefined, distance we will assume that it has detected us too and it supposes that we will stick to our lane, thus not invading it arbitrarily. By doing so, we redefine a bound on the lowest speed reached when crossing another car.

Please note, that to use this approach is not necessary that the vehicle is actually capable of detecting lanes. Using the detected traversable space (road) and an a priori on the size of cars it is possible to define lanes on the streets accurately enough.

Small intersection

In figure 2.9 we present the scenario if a simple intersection. This case follows the same logic described until now, the movement of other vehicles being restricted to the detected roads (and lanes direction, if defined).



(a) Obstacle plans EW and robot has a partial SN plan

(b) Both vehicles are pushed towards a corner

(c) Both vehicles cannot move forward anymore

(d) Pink indicates the "non stop area"

Figure 2.12: A deadlock appearing in a simple intersection, and how to avoid it. See also caption of figure 2.9

With the proposed approach, the “vehicle stopping in a railway” situation may be a concern. This is, the vehicle stops in the middle of intersection, letting itself being collided (harm by inaction). First, as shown in theorem 2.1, if the “train” respected the safety guidelines no collision would actually occur. Second, even if the vehicle actually plans to stop in the intersection, as the vehicle moves its observed area evolves and thus the plan is updated to actually cross the intersection. The vehicle will only stop in the middle of the intersection if no other safe plan is found or if that is it actual target pose.

When no other vehicle approaches the intersection the robot will slowdown it speed considering the apparition of cars at the limits of it detection range. This case follows again the same logic of the open space scenario.

The only inconvenient situation that could arises with the proposed approach (and did arise in the simulations of chapter 6) occurs when another vehicle approach the intersection. It is possible that the robot will engage the intersection with a partial plan, that allow it to stop safely and but not to cross the intersection. As the second car approaches, the robot may find itself cornered and not able to find any new plan allow to move, as illustrated in figure 2.12. This kind of behaviour may lead to a dead lock in an intersection, which is obviously a highly undesirable (but harmless) situation.

We propose as a solution to introduce into the planning the notion of “non stop areas”. If a new plan ends in a non stop area we do not update the current plan (we keep as it is). Non stop areas, are road regions we believe to be a resource shared between multiple lanes. Applying this simple logic the vehicle will stop before entering the crossing and will only cross it once it has a plan to “escape from it”. This solves the dead locks on simple intersections and the “car on the railway” problems.

The different example scenarios discussed show that the proposed approach to safety is viable, even with conservative assumptions. They also illustrate how, when ensuring safety, the final behaviour depends on the hypotheses used when predicting the future of the world.

2.8 Conclusion

Safety is a critical issue for driverless vehicles, however it remains a fuzzy aspect in many proposals. All of the classic planning methods and most of the previous works on driverless vehicles are arguably unsafe. We reviewed the safety issue and how the safety apply to an incomplete and uncertain world model.

From the analysis developed we can assert:

- The safety of the vehicle does not depend on the actual planning method employed, it only matters that the provided solutions respect definition 2.9.
- To ensure safety it is required that the control module provide and respect a bound on the trajectory tracking error.
- In order to obtain a safe motion the perception module needs to provide a conservative prediction of the traversable space in time and an estimate of the current state of the vehicle. These estimations need to be done with respect to a reference frame attached to the ground and not relative to the vehicle itself.
- The range and precision of the sensors does not affect safety as long as they are correctly modeled. They will however influence the final behaviour of the robot, in section 2.7 we provide some reference values for design.
- In order to manage safe motion and unexpected collisions, it is necessary to be able to quantify the future harm by defining the function $h(x(t), w(t))$.

We have shown that when the perception-planning-control trio respects these constraints it is possible to ensure that, within the limits of the used world model, the vehicle will not actively harm. The final behaviour obtained by a robot following the suggested approach will be as safe as humanly possible.

Chapter 3

Perception

> I can live with doubt and uncertainty and not knowing. I think it is much more interesting to live not knowing than to have answers that might be wrong.
>
> Richard Feynman

In chapter 4 we will describe a system that allows the robot to:

- localize itself in the city
- build a local map of the static obstacle
- detect and track the surrounding moving obstacles
- localize itself on the local map

This functions will be used to build a world model satisfying the needs described in the chapter 2 (see section 2.8).

In this chapter we will define what a perception system is, describe the underlying core problem, and show how a single approach encompasses sensing, sensors fusion, localization, simultaneous localization and mapping (SLAM) and the SLAM with moving objects detection and tracking problem (SLAMMOT). Solving the SLAMMOT problem corresponds to effectively be able to do the four tasks listed.

This chapter presents a brief but extensive view of the state of the art on the subject of robots perception in the context of indoor, urban and suburban environments. The review of the existing techniques and their interrelation will provide the foundations for the system proposed in chapter 4.

Readers familiar with SLAM and moving objects tracking techniques may move forward in this chapter and only focus on sections 3.5.1 and 3.7.

This chapter follows a bottom up approach (raw sensors at the bottom, SLAMMOT output at the top) in order to review the underlying difficulties and existing alternatives that lead to design a SLAMMOT system. Sections 3.1 and 3.2 define the perception problem for a ground mobile robot. Section 3.3 reviews existing sensors. Section 3.4 explains how to uses multiple measures to build a single world model, the notions of Bayesian programming, Bayesian networks and sequential Bayesian filtering are introduced. Section 3.5 presents the localization problem and its underlying data association and data representation problems. It is shown that the localization problem can be formulated as a sequential Bayesian filtering problem. Section 3.6 presents the SLAM problem and briefly review existing solutions. It is shown that the SLAM problem builds up on the localization problem by sharing the data association and data representation elements and extending the filtering problem to a larger one. Section 3.7 presents the rather novel SLAMMOT problem that further extends the filtering problem by including the moving objects detection and tracking. Finally section 3.8 presents some though on how to further extend the perception capabilities of mobile robots and section 3.9 review the nuclear ideas presented during the chapter.

3.1 What is "perception" ?

Every robotic system can be seen as a composed by three modules: sensing, planning and acting. In most of the cases the robot will be interacting with the physical world. Machines only have access to this world through the measures provided by sensors. Interpreting the measured values in order to take decisions about our future actions is what perception is all about. Without perception the robot would not be able to take decisions, and thus to fill its purpose.

Unfortunately in English the word "perception" refers to the process of perceiving, the way of perceiving, the representation of what is perceived and the knowledge gained by perceiving. This ambiguity leads to multiple confusions. In this dissertation we will stick to the usage of perception as the process of perceiving. The representation of what is perceived will be called *world model*. The way of perceiving is defined by an instance of the perception system. The knowledge gained by perceiving will be referred with loose terms such as "information" or "knowledge of the robot".

A short definition of perception as a process is [39]

```
sensing, perception -- (becoming aware of something via
                        the senses)
```

As any language definition, it is incomplete (what "aware" means?). In the rest of this chapter we will discuss in more detail what perception is, in the context of ground mobile robotics, and which are the existing techniques to solve this problem.

3.2 Problem definition

For robotics purpose, a more precise definition is

Perception is the process of transforming measurements into a world model

The model built has to be useful to take decisions, either by humans or by another algorithm.

Starting from this definition the perception problem becomes threefold:

- What to measure?
- Which model to build?
- How to transform the measures into the desired model?

None of this question have one single answer since *perception depends on the specific application*. In this dissertation we will focus on the problems related to ground mobile robots, however much of the discussed methods can be directly used other application domains (medical, air, underwater, etc.).

Independent of their specific purpose mobile robots need to ensure their mobility capabilities. Thus they are likely to have some or all of the modules mentioned in section 1.4. Each decision module impose it own requirements on the information that the perception module should provide.

Perception ▶ Control In order to control its movements the robot needs to estimate its state, this can be done using propioperceptive sensors (internal) and/or exoperceptive sensors (looking around). Which are the components of the state vector to estimate depends on the specific robot and on the control method employed.

Perception ▶ Trajectory planning In order to define a trajectory for its future movements the robot needs to know which areas of the surrounding space can be traversed and which not. This usually involves detecting the surrounding obstacles, but also includes detecting banned areas or areas where the vehicle cannot move (e.g. cleaner robot should not enter the elevators, or go into the stairs). If the presence of moving obstacles in the surroundings is envisaged, estimating their future movement is necessary to avoid future collisions.

Perception ▶ Route planning In order to decide the route, a routes network is used. This information can be constructed from previous visits of the robot (large scale mapping problem). Once this information is provided the robot needs to know his position such roads network. Given the change of scale the precision required and methods used are probably different from the one used for control.

Application specific tasks may include for instance, detailed maps building, site recognition, persons detection, persons recognition, human commands recognition.

In any of the previous examples, essentially, *the perception task is a state estimation problem*, ranging from simple single values estimation (e.g. estimating a temperature) to extremely large states description (e.g. colourised voxel representation of the visited areas). For mobile robots, the world model to estimate will usually include the state of the world surrounding the robot and the position of the robot in this world.

Knowledge $\neq$ Certainties It is important to understand that knowledge about the world does not mean certainties about it. Knowledge of the real world is always uncertain due to noise in the measures and errors (or simplifications) in the models. A correct representation of the knowledge acquired has to include the uncertainty of the state vector (positions, sizes, class of the objects, number of objects, etc.). When positions have uncertainties, measures in different reference frames are not equivalent. For instance, having high precision local positioning (street level) does not ensure accurate global positioning (city level).

The presence of noise and uncertainties is one of the reason that lead to the use of the probabilities as a sound mathematical framework to manipulate the low level data.

Defining the parametric form of the world model, the observations made and method to estimate the parameter from this observations considering the limited resources available (computing and measurement capabilities per time unit) is the core problem of perception.

3.3 Sensors

Sensors are the basic element of the perception, they transform physical signals into signals understandable by the machine. The data obtained by the sensors is the basis of the perception system and thus it is also what defines its limits.

The choice of the sensors for a particular task is non trivial, criteria for their selection are cost, precision, range of measures, energy consumption, effect over the environment.

For our presentation we will separate the sensors in three classes: passive, active and infrastructure based. Passive sensors receive environment energetic signals and transform them (e.g. video cameras). Active sensors emits signals to then analyze the response of the environment to them (e.g. radar, ladar, structured light). Infrastructure based sensors require a previous intervention of the environment to insert tags, transponders or emitters which are later used by the sensor to construct its output (e.g. electromagnetic tags, GPS, IR reflectors).

In urban areas the cost of infrastructure installations can be considerable and it restricts flexibility to the perception system (limited areas of operation). Active sensors have to

be designed in order to not disturb the normal operation of the city (no disturb for humans, animals or previously deployed systems). In the past most work has been limited to applications where the use of infrastructure or active sensors are not restrictive, the state of the art research try to obtain better results while reducing the intrusive nature of the robot.

For the perception process it is important to understand the capabilities and limitations of a sensor and to be able to model its behaviour, i.e., the relation between the elements to be measured and the sensor output. Such a model will be the base for an accurate use of the provided data. Accurate modelling of the sensors will lead to better perception systems.

In the following subsections we will briefly present existing sensors with respect to our desired application and discuss the selection criteria. In sections 3.3.2 and 3.3.3 we introduce some of the notions required to use the measurements information.

3.3.1 Internal measures

Using measures of its internal state and a model of its dynamics the vehicle is able to estimate its displacement. For this task measures of wheels rotation (differential encoder), steering angle (absolute encoders), motor control signals (via voltmeters or digital control buses), speeds (based on wheels rotations and gyroscopes), accelerations (using accelerometers, measures of the acceleration command) etc. are used. The measurements of this low level instrumentation, common in industrial electronics, is then introduced in an inference machine to estimate the actual state of the vehicle (position, orientation, speed, etc.). The design and implementation of such an inference machine is discussed in the section 3.4.

Modern cars include from the factory instruments that measures most of the internal state variables, and such data is accessible in real time via a local data bus (usually a CAN bus [40]).

In order to estimate the actual state of the system, both measurements and a model of the system are required. Errors in the estimated displacement are induced essentially by noise in the measurements and errors in the models. For car like systems the correct use of internal sensors leads to a good estimation of the displacement for straight lines trajectories (less than 1% errors). However when realizing curves the sliding effect of the wheels, habitually omitted in the models, introduce important errors in the orientation, that finally degrade the total Cartesian displacement estimation.

Independent of the quality of the model and precision of the measures the presence of noise in the measurements and in the dynamic system are inevitable. This noise causes a continuous divergence, the uncertainty grow is constant and unbounded. This essential limitation makes necessary to always use external measures to relocate the vehicle in the space, correcting the executed path estimation and reducing its uncertainty.

It is also obvious that internal measures are a poor element for perception because they only give proprioceptive information and no knowledge of the external world which is a requirement in almost any application.

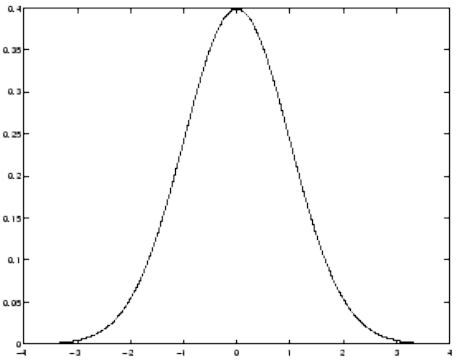


Figure 3.1: Gaussian distribution $\mathscr{G}(0,1)$

Since internal captors are cheap and almost always available (because of the internal control loops of the system) internal measures are commonly employed, at least to compute a rough initial estimate of the ego-motion.

3.3.2 Noise model

For mathematical and computational simplicity the most commonly used noise model is the white Gaussian noise. The measures of the physical magnitudes are supposed to be corrupted by a white noise signal whose distribution can be approximated by the Gaussian distribution (also called normal distribution), is illustrated in the figure 3.1 . The distribution of the signal is supposed to have a zero mean value and a variance to be determined experimentally.

The correct modelling of the noise that affects the signals will have a direct impact on the quality of the information extracted from them.

In most of the cases the white Gaussian noise is simply a rough but useful approximation. However this approximation should be verified, because wrong noise modelling (wrong assumption) can lead to large errors in the estimation of real world objects.

The so called "white noise" is a noise signal that is not correlated in time, i.e., its actual value give no information about the next value. Not every physical noise signal is white, not even in general.

The univariate Gaussian distribution is a function of two parameters, the mean value μ and the standard deviation σ. Thus the probability density function is

$$p(x) = \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \sim \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Thus if a noise signal X can be described by a Gaussian we write $X \sim \mathscr{G}(\mu,\sigma)$ or, alternatively, $X \sim \mathscr{N}(\mu,\sigma)$.

In general the noise of a system will be described by the presence of noise vectors. Then a multivariate Gaussian distribution is employed. This multivariate distribution is parametrized by a mean vector $\vec{\mu}$ of dimension d and a covariance matrix Σ of dimension $d \times d$. The expression of the probability density function is then

$$\begin{aligned} p(\vec{x}) &= \frac{1}{(2\pi)^{\frac{d}{2}} \cdot |\Sigma|^{\frac{1}{2}}} \cdot \exp\left(-\frac{1}{2} \cdot (\vec{x}-\tilde{\mu})^t \Sigma^{-1} (\vec{x}-\tilde{\mu})\right) \\ &\sim \exp\left(-\frac{1}{2} \cdot (\vec{x}-\tilde{\mu})^t \Sigma^{-1} (\vec{x}-\tilde{\mu})\right) \end{aligned}$$

Commonly the vector notation is omitted as the univariate case $\mathscr{G}(\mu, \sigma)$ is unambiguous respect to the multivariate case $\mathscr{G}(\mu, \Sigma)$.

An important concept associated to the standard deviation is the "confidence interval". These intervals define ranges of values where we have a given confidence that the real value of the variable resides into. It can be calculated that the range $\pm\sigma$ contains 68% of the instances of the modeled event, $\pm 2\sigma$ contains 95% of the instances and $\pm 3\sigma$ correspond to 99,7% of the instances.

Thus having a good estimate of the variance (σ or Σ) it is possible to bound up to a 99% of certainty the range where the real value of interest is placed. This principle is commonly used to pass from probabilistic description, to non-probabilistic descriptions (using a threshold for the certainty).

3.3.3 A simple car model

As we will see in section 3.4.4 in order to use the inertial measures it is necessary to also have a model of the system itself. Let us then introduce such a model.

Cars and car-like vehicles have complex dynamics, including combustion processes, mechanical connections, torsion of materials, fluid dynamics (fuel in then tank, air friction, wheels model), friction and sliding with the road, etc... From an external point of view and with the interest of modelling the motion of the vehicle as a particle, very simple models are used.

The simplest model utilized (which is the most employed) does not consider friction or slipperiness, it supposes that an internal control loop is able to regulate the dynamics and only consider the non-holonomic capabilities of the automobile [41, 42]. Discarding the slipperiness of the road allows to collapse the wheels (see figure 3.2), thus this model is sometimes called the "bicycle model".

All the simplifications lead us to a nonlinear kinematic[1] model of the form (3.1) where the state of the vehicle is represented by a vector in $\mathbb{R}^4$. This vector is composed by:

x, y Cartesian coordinates of the middle point between the rear wheels,

[1] "Kinematics is the branch of mechanics concerned with the motions of objects without being concerned with the forces that cause the motion. In this latter respect it differs from dynamics, which is concerned with the forces that affect motion." [43]

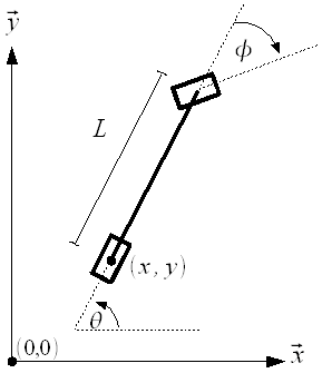


Figure 3.2: The bicycle model: a simplified kinematic model of a car

θ vehicle orientation (with respect to the Cartesian plane) and

ϕ steering angle of the front wheels (which are supposed to be parallel).

$$(\dot{x}, \dot{y}, \dot{\theta}, \dot{\phi})^t = f(\ (x, y, \theta, \phi)^t,\ (u_1, u_2)^{\,t},\ \eta) \tag{3.1}$$

The input that affects the state vector is a bi-dimensional vector composed by

u_1 driving velocity (mean value of the rear wheels) and

u_2 steering velocity.

Finally the model has two parameters the intrinsic noise vector η and the length of the vehicle L measured between one wheels axis and the other, as presented in the figure 3.2.

The equation 3.2 establish the nonlinear relation between the presented magnitudes.

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ \frac{1}{L} \cdot \tan\phi \\ 0 \end{pmatrix} \cdot u_1 + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \cdot u_2 + \eta \tag{3.2}$$

Clearly this model is the simplest one, after the punctual mass, and it is useful for our task because it integrates the limitation in the displacement and discard all the other spurious elements. Models with the same objective but with incremented complexity are the bi-steerable model presented in [44, 45] and the four wheels presented models at [46].

3.3.4 Satellite based positioning systems

After the internal sensors, the second most used sensor are the infrastructure based satellite global positioning systems (such as the American GPS, Russian GLONASS, Chinese Beidou or European Galileo [47]). Developed in order to locate objects (boats, cars, planes, satellites) all around the world in a common reference frame these systems are today part of the daily life of individuals in the developed world.

The precision of a cheap commercial receptor is measured in various meters (five, ten meters). With the help of a city map the GPS measures can be good enough to localize a vehicle in a specific street, but not to localize the vehicle inside this street (lane, distance to intersection, etc.) [48, 49]. The standard GPS resolution can be enhanced by diverse methods [50], requiring local infrastructure. Through these methods centimetric resolutions can be achieved (few centimeters of uncertainty). However these methods are strictly limited to local regions since they use ground infrastructure. They are normally used as ground truth to compare with other localization methods (see [51] for a good introduction on the subject).

Together with a measure of the position the GPS (and the other systems) provides to the receptor a precise measure of the time (which is used internally to calculate the position). Between two close GPS receptors that observe the same satellites, the relative time measure error is in the order of a few milliseconds (or lower).

The standard GPS is a good tool to localize a vehicle in the large, however in urban areas it tends to be unreliable because it require lines of sight with at least three satellites. In dense urban areas the surrounding buildings commonly interfere in the electromagnetic waves propagation, blocking the access to the satellites and thus to new location estimations. Another factor of unreliability is caused by the trajectory of the satellites. As the GPS satellites are orbiting around the earth, at the same location but in different time the GPS measures can become unavailable. Experimental results show that in dense urban areas reception regions can cover as little as 50% of the total surface [52].

The low cost of the GPS receivers (with respect to a vehicle) and the possibility to use them for global positioning make them a common sensor for vehicles perception. However the mentioned limitations do not allow to use them as the solely localization system and, similarly to odometry, they have to be used in combination with other methods.

The habitual model for the GPS is to suppose that data will be reliable and that the measures are corrupted by Gaussian noise $\mathcal{G}(0,\sigma)$. The standard deviation σ is adjusted to the values that the manufactured indicate (e.g. ± 5 $[m]$ at 95% of certainty). The Gaussian assumption is more or less correct, however a common mistake is to suppose the GPS measure error as a white noise signal. The measurement errors are principally associated to the position of the satellites with respect to the receptor and to the characteristics of the communications channel (atmospheric, multiple paths, etc.). Both of these variables evolves smoothly in the time, and those the noise signal has an important component correlated in the time.

Applying filtering methods that supposes white noise (as Kalman filtering does) will

Table 3.1: Comparison of external sensors

Name	Range± precision [m, degrees]	Rate [Hz]	Robust-ness	Applicability for driverless vehicle
Sonar	5±0.1, 0º ±10º	20	–	Very poor performance
Short range radar	50±2, 50º ±1º [55]	20	√√√	Poor for pedestrians
Long range radar	150±0.5, 20º ±0.7 [56]	10	√√√	Poor for pedestrians
Ladar	100 ±0.05, 240º ±0.5º [57]	20	√√	Fragile and expensive
Velodyne	100 ±0.05, 360º ±0.1º [58]	15	√√	Fragile and expensive
Solid state ladar	20±1, 50º ±1º [59]	50	√	Good, but unavailable
Mono vision	- , 40º ±0.25º (variable)	30	–	Complex image processing
Stereo vision	(distance^2) , (variable)	20	–	Noisy, no error model
Omnidirectional	- , 360º ±0.5º	30	–	Complex image processing
Structured light	(distance^2), (variable) [60]	1	√	None (in visible spectrum)
Infrared	- , (variable)	30	–	Complex image processing

lead to considerable errors or even divergence. For this kind of signals H_∞ filtering seems to be a more appropriate method [53, 54].

3.3.5 Comparing external sensors

Up to now we have discussed the user of internal and global positioning sensors. None of these provide information about what is around us, yet this is the most critical information that allows us to build the harm function estimate and thus decide the actions to take.

A detailed review of existing sensors and their technology seems out of the scope of this dissertation. Instead we present two condensed tables. In table 3.1 we present representative values of the performance of external sensors used in commercial cars and robotics applications. Then in table 3.2 we present a sample of research work related to different perception functions implemented using specific sensors.

From the results in the tables ladar and vision seems to be more promising sensors. Radars are commercially used in modern cars but offer technical limitations to their use on driverless vehicles. Ladar sensors (see figure 3.3) seems to be the more promising solution, with good solutions today and better ones being developed (see figures 3.4 and 3.5).

Vision is probably the ideal solution. Through passive sensing with comparatively cheap sensors it provides very rich information. However, unlike the ladar, it does not provide direct access to the distance to obstacles information. Despite extensive research, correctly estimating the distance to obstacles (and its error) has proven to be a very difficult problem in the general case.

In the short term laser based sensors are probably going to prevail in robotic applications, but in long term vision based perception is the best known solution.

Table 3.2: Use of sensors for different perception tasks. "Common use" indicates that solutions are provided by commercial companies. The referenced works are an arbrirary sample of the large corpus of existing related research

Name	Localization	Displacement estimation	Obstacles detection	Mapping (SLAM)
Sonar	Poor	Poor	Common use	[61, 62, 63]
Radar	No	No	Common use	[64]
Ladar	Yes	Yes	[65]	Common use
Solid state ladar	Yes	Yes	[66, 59]	[67, 68]
Mono vision	[69, 70]	[71, 72]	[73, 74, 75]	[76, 77, 78]
Stereo vision	Possible	[79, 80]	[81, 82, 83]	[84, 85]
Omnidirectional	[86, 87]	[88]	Possible	[89, 90]
Structured light	Possible	Possible	Possible	[60]
Infrared	Possible	Possible	[91]	Possible

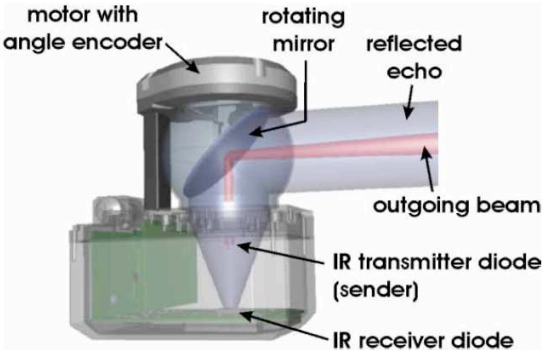


Figure 3.3: Laser-scanner schema. From [65]

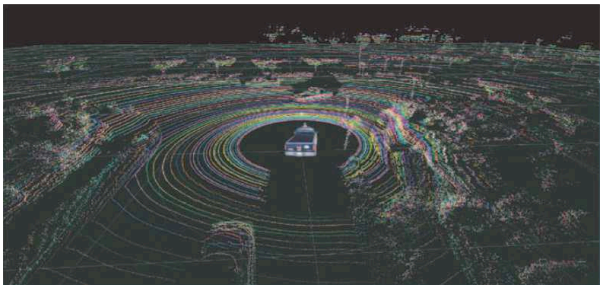


Figure 3.4: Velodyne high definition 360° lidar output. From [58]

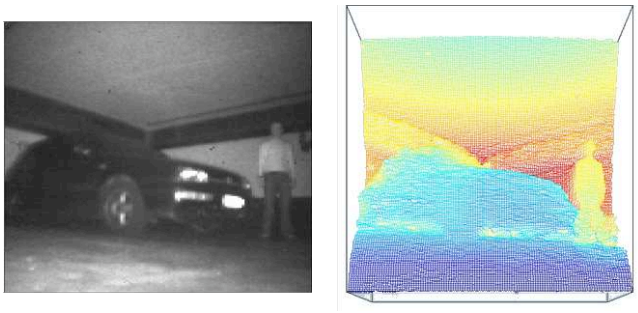


Figure 3.5: Depth measures from a state of the art active vision system. Left side, the real scene, right side, the depth map. From [92]

Figure 3.6: Sensor data processing stages

3.4 Sensors fusion

As seen in the previous section no single sensor is able to give all the required information of the world. Most of the time different sensors gives complementary information of the environment, when redundant overlapping measures are available it is possible to enhance the resolution exceeding the sensor intrinsic resolution.

Sensor fusion deal with the merge of different sources to obtain a consistent unique representation of the world.

It has to be noticed that fusion can be done at the different levels of processing and that the output of different algorithms applied to one same sensor are also subject to be fused. This lead to the more generic concept of *data fusion*.

Data fusion at different level can be lead up to fusing heterogeneous decisions over one situation. In this text we will center the focus on the fusion systems at the perception level.

3.4.1 Sensor fusion stages

At the perception layer the data is acquired and processed in a sequence of operations. A possible sequence is presented in the figure 3.6 [93].

First the obtained data is reconstructed at the adequate scale and put in the correct reference frame (example: laser scan binary packets are transformed into point in the space). Then the raw points are transformed to match the internal representation (see

section 3.5.1 for an overview of different data representations). In this stage raw data is analyzed and used to construct map of features, or to feed a grid of evidence, for example. This representation is then processed to extract objects (as defined in the world model) that will be classified (assignation of a model for a specific objects) and whose state will be estimated (state of the dynamic model selected).

Of course this schema is just a generic representation and has to be adapted to specific cases. The interesting point is to observe that the heterogeneity of sources can be helpful at every stage of the processing, passed the reconstruction level.

To be able to fusion the data from different sensor all the reconstructions have to be made on the same unique reference frame. This means that a previous calibration step is required in order to reconstruct the data in the same spatial referential and that the measures of different sensors have to be tagged with the same time counter. Sometimes omitted this aspect is crucial in practice.

The diversity of sensors, data transfer protocols, calibration procedures and the requirement of common calibration and synchronized timestamping is creating pressure for the development of a common sensors framework in the automotive automation market and for the robotics domain in general [93].

Having mentioned these practical issues it is time to discuss more conceptual aspects. At the most basic level it is not trivial how to merge heterogeneous information, how different sources will lead to only one decision. Even more sensor measures are imprecise and can be incoherent with respect to other captors. Different theories and algorithms have been employed for sensor fusion (fuzzy logic, decision theory, Dempster-Shafer theory), however in this dissertation we will stick to only one approach. Bayesian probabilities seems to be the more principled, sound and popular theory to manage the uncertainty [94] and thus is the best candidate for sensor fusion.

In this text we will use the Bayesian approach as a conducting line through the incremental complexity of the problematics and to show the relationship between them.

3.4.2 Bayesian programming and Bayesian networks

Many works have been done in robotic perception and in probabilistic robotic perception. Such a proliferation of works has scrambled their interrelation due to different nomenclatures and different presentations.

In order to make this dissertation coherent and structured we will present the different probabilistic approaches using two tools : Bayesian programs and Bayesian networks.

Bayesian programs are a simple but clear and powerful formalism to present Bayesian models (probabilistic description of the interrelation between a set of variables and the attributes of these variables). On their side Bayesian networks are a special kind of graphical models that allow to represent graphically Bayesian models. Bayesian programs are a more powerful and auto-contained presentation of Bayesian models that the Bayesian networks. In this dissertation we will use Bayesian networks as a complementary (and voluntarily redundant) visual support for the Bayesian programs.

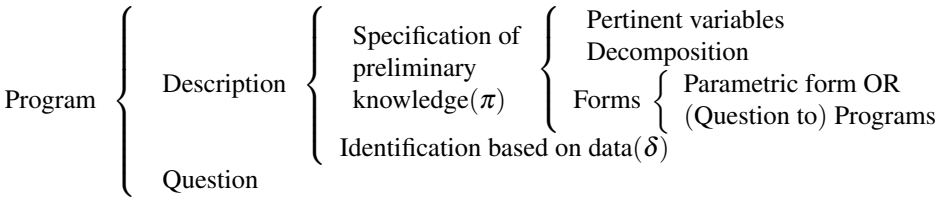


Figure 3.7: Structure of a Bayesian program

In the rest of this subsection we will do a very brief introduction to these representations. The reader is invited to consult [95] and [96] for a more pedagogic and complete presentation of these tools.

Bayesian programs

A Bayesian program is composed by a description of a Bayesian model and a question over it. The description itself presents :

- the variables involved,
- a structure in the joint distribution of the variables, that lead to a specific decomposition
- the form (family of distributions) of the different component of the joint distribution
- the method of identification (of parametrized or non-parametrized forms)

The graphical representation of these elements is illustrated in the figure 3.7. We will exemplify this figure with the sensor fusion example.

Bayesian networks

Bayesian networks are a specific kind of graphic model. This drawing encodes in a directed graph the dependencies between random variables. The independence between variables allow to decompose a joint distribution into a product of lower order joint distributions, which is commonly the information presented in the decomposition field of a Bayesian program.

Continuous variables are represented by circle nodes, discrete variables are represented by square nodes, observable variables are grey and dependence relations are represented by directed edges, as shown in the figure 3.8.

The careful reader will have noticed that Bayesian networks encodes dependencies while inference computation is lightened by independence conditions. The independence conditions are available in the graph but are non trivial to read (we have to play Bayes ball), see [96] for an introduction on the subject.

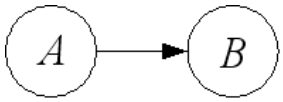


Figure 3.8: Variable *B* depends on *A*

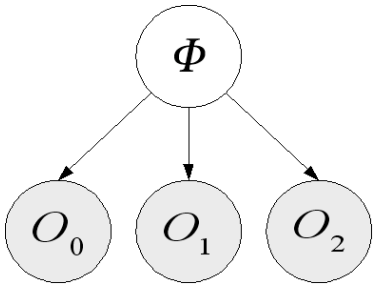


Figure 3.9: A Bayesian network for sensor fusion

3.4.3 Bayesian sensor fusion

From the Bayesian perspective the essential sensor fusion problem is a quite simple one. We will employ it to exemplify the formalisms presented in the subsection 3.4.2.

Let us suppose that an event Φ occurs at a given instant. This event is measured by three sensors, retrieving observations O_0, O_1 and O_2. The values of the observation variables O_i *depends* on the value of the event Φ. This relation is encoded visually in the Bayesian network of the figure 3.9. The nodes of the observations are grey because their value is accessible, while the node of the event is non grey because its value is unknown.

Having access to the observations, the purpose of the sensor fusion is to employ the information contained in the different measures to obtain the best possible estimation of Φ, the value of the event. This is illustrated in the figure 3.10 under the formalism of Bayesian programs.

The decomposition of the joint distribution $P(\Phi\ O_0 \ldots O_N)$ can be deduced from the Bayesian network, and vice versa. It is also relevant to emphasize that the Bayesian program sets up a problem, but does not pretend to give an explicit solution to it. The factorization of the joint distribution is a technique to reduce the computation of the Bayesian inference problem (that is NP hard in the general case, i.e., intractable).

In sensor fusion the question is

$$P(\Phi|O_0 \ldots O_N) = ?$$

Applying Bayes theorem over the joint distribution $P(\Phi\ O_0 \ldots O_N)$, we can write

Prog
- Desc
  - Spec
    - Variables :
      - Φ Event
      - $O_0, \ldots, O_N$ Observations from different sensors
    - Decomposition :
      - $P(\Phi\, O_0 \ldots O_N) = P(\Phi) \cdot \prod_{i=1}^{N} P(O_i|\Phi)$
    - Forms : any
  - Identification : any
- Question :
  - $P(\Phi|O_0 \ldots O_N)$

Figure 3.10: Sensor fusion as a Bayesian program

$$P(\Phi|O_0 \ldots O_N) = P(O_0 \ldots O_N|\Phi) \cdot \frac{P(\Phi)}{P(O_0 \ldots O_N)}$$

Then applying the conditional independence (expressed in the decomposition and in the Bayes network)

$$P(O_0 \ldots O_N|\Phi) = \prod_{i=1}^{N} P(O_i|\Phi) \Rightarrow P(\Phi|O_0 \ldots O_N) = \prod_{i=1}^{N} P(O_i|\Phi) \cdot \frac{P(\Phi)}{P(O_0 \ldots O_N)}$$

Commonly the distribution $P(\Phi)$ is assumed uniform, thus it corresponds to a fixed coefficient. On the other hand it has to be noticed that the distribution of interest $P(\Phi|O_0 \ldots O_N)$ corresponds to a function $f(\Phi)$ over the range of Φ. Then, the factor $P(O_0 \ldots O_N)$ corresponds to a fixed value, independent of Φ. These two observations allow us to write

$$P(\Phi|O_0 \ldots O_N) \propto \prod_{i=1}^{N} P(O_i|\Phi) \tag{3.3}$$

The proportionality factor is unknown and depends on the actual observation measures. However this is not important since it can be deduced from the restriction that the integral of a probability density function has to be equal to one. The important fact from equation 3.3 is that it shows that the solution from the sensor fusion problem is the multiplication of known probability distributions, where $P(O_i|\Phi)$ corresponds to the model of each sensor i.

One of the favourite forms for the distribution $P(O_i|\Phi)$ is the Gaussian distribution. When two Gaussian distributions $\mathcal{G}(\mu_1, \Sigma_1)$, $\mathcal{G}(\mu_2, \Sigma_2)$ are multiplied the resulting Gaussian distribution $\mathcal{G}(\mu_3, \Sigma_3)$ has a mean vector and a covariance matrix defined by

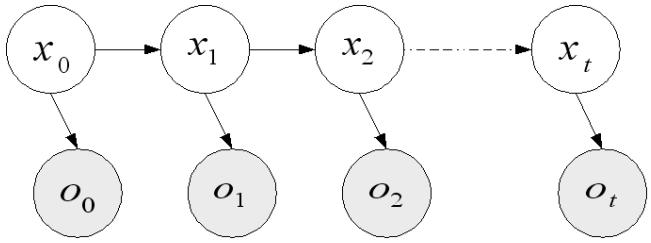


Figure 3.11: Dynamic Bayesian network of the observation of an evolving system

$$\begin{aligned} K &= \Sigma_1 \cdot (\Sigma_1 + \Sigma_2)^{-1} \\ \mu_3 &= \mu_1 + K \cdot (\mu_2 - \mu_1) \\ \Sigma_3 &= (I - K) \cdot \Sigma_1 \end{aligned}$$

From these expressions it can be seen that the fusion of multiple measures (ruled by Gaussian distributions) will consistently reduce the variance of the result, thus enhancing the precision of the estimation of the value of Φ.

3.4.4 Sequential Bayesian filtering

Multiple simultaneous observations of an event give more information about it. Multiple observations in the time of a time invariant element is equivalent to the multiple simultaneous observations. Sequential observations of a system evolving in time give more information about its actual state that a single observation of it. This is the core idea behind the sequential Bayesian filtering.

Bayesian filtering allows to use past and present measures to enhance the estimation of the actual state of a system. Such measures can be originated from a single source or from a diversity of sources, leading to the sensor fusion case.

The figure 3.11 presents a Bayesian network that shows the evolution of a discrete system and the observations made from it. The associated Bayesian program is presented in the figure 3.12.

As the Bayesian networks presents the relation between variables evolving in the time, it is commonly named as a “Dynamic Bayesian network”. Notice that the presented graph illustrates the Markov assumption, i.e., the current state depends only on the past one.

The presented Bayesian network is the simplest one related to a Bayesian filtering problem. In general we will have more that one observation of the state, or multiple partial observations of it. Also some applications will have measures depending of two states or access to inputs of the system. All those cases will be managed with the same unifying approach.

Prog
- Desc
  - Spec
    - Variables
      - $x_0, \ldots, x_t$ State vectors in time
      - $o_0, \ldots, o_t$ State observations in time
    - Decomposition
      - $P(x_0 \ldots x_t \; o_0 \ldots o_t) =$
      - $P(x_0)P(o_0|x_0)$
      - $\prod_{i=1}^{t}[P(x_i|x_{i-1})P(o_i|x_i)]$
    - Forms : any
  - Identification : any
- Question :
  - $P(x_t|o_0 \ldots o_t)$

Figure 3.12: Bayesian Filtering

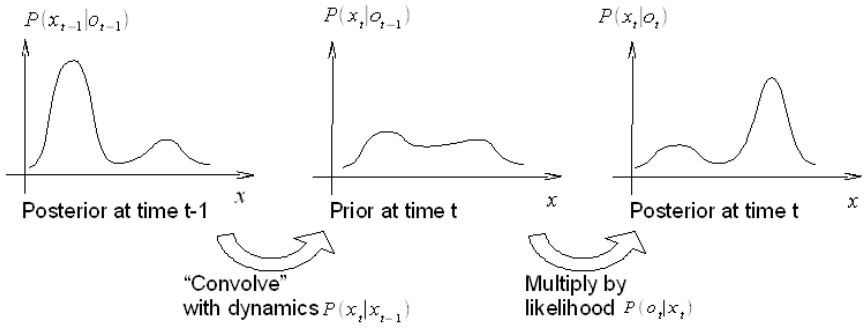


Figure 3.13: Sequential Bayesian filtering

For the sake of completeness it worth mentioning that the Bayesian program of the figure 3.12 is specific for Bayesian Filtering. When the question is $P(x_{t+k}|o_0 \ldots o_t)$ then such a task is called Bayesian Prediction, and when the question is $P(x_{t-k}|o_0 \ldots o_t)$ it is called Bayesian Smoothing (k is a positive integer).

An important property of the Bayesian Filtering problem is that it can be solved through an iterative algorithm. The recursive relation is presented in the equation 3.4, see [95] for its derivation.

$$P(x_t|o_0 \ldots o_t) = P(o_t|x_t) \cdot \sum_{x_{t-1}} [P(x_t|x_{t-1}) \cdot P(x_{t-1}|o_0 \ldots o_{t-1})] \tag{3.4}$$

This algorithm can be interpreted as transformations over distributions of probability. Using the evolution model $P(x_t|x_{t-1})$ we obtain a distribution $P(x_t|o_0 \ldots o_{t-1})$ prior of the new measurement (this distribution is commonly called "the prior"). Then introducing the new measure we obtain the desired result $P(x_t|o_0 \ldots o_t)$ (commonly called "the posterior"). This process is illustrated in the figure 3.13.

In order to compute the sequential Bayesian filter a definition of $P(o_t|x_t)$ and $P(x_t|x_{t-1})$ is required. $P(o_t|x_t)$ corresponds to the sensor model and is commonly known in advance. $P(x_t|x_{t-1})$ is a description of the dynamic of the system. A parametric model of the system dynamics can be estimated (online or offline) from previous samples or known a priori.

For any implementation solving an inference problem the description of the probability density functions has to be chosen. Common choices are unimodal Gaussian distributions, mixture of Gaussian distributions and set of particles. Also it has been proposed the use of binary trees [97], splines and wavelets.

Kalman filter and variants When the sensor model is Gaussian and the dynamic model is linear with Gaussian noise then the sequential Bayesian filtering algorithm leads to the well know Kalman filter (KF) [98, 99, 100]. When the dynamic models are not linear, Kalman filter cannot be applied directly (since a Gaussian posterior at $t-1$ uncertainty becomes non a Gaussian prior at time t). An approximation of the Bayes Filter, called Extended Kalman filter (EKF), consists on linearizing the model at the current state estimate (a linear projection of Gaussian distribution stays a Gaussian distribution). A better way to approximate, named Unscented Kalman Filtering (UKF) [101], projects the Gaussian distribution through the non linear model and then estimate the first and second moments of the output (instead of using a projection over the first derivative).

Particles filter A classical method to deal with non linear systems and arbitrary probability distributions is the so called Particles Filter [102]. Instead of representing the probability density function by a parametric function, it is represented by a group of particles that sample its surface. The more particle, the better represented will be probability function. The core idea of the particle filter is to manage the set of particles in order to keep them on the areas of maximum likelihood.

Particles filtering is a very powerful method that can manage any distribution (notably multi modal ones) and any non linear function. Defining the good number of particles required and ensuring that they still sampling correctly the high likelihood regions is in general not well defined. However they are used over many practical problems using a "high enough" number of particles.

Presenting in details all of this Bayesian Filtering instances requires lengthy equations. As this are classical methods we refer the reader to the previous citations to explore each method in detail.

A classical application of data fusion and sequential filtering are the GPS + Odometry global localization systems. Noisy measures of the robot position and displacement are available and the use of fusion and filtering dramatically enhance the positioning precision. The reader can consult [103, 104] for a detailed discussion on both the theoretical and practical aspects of this application.

3.5 Localization

Mobiles robots to localize themselves to follow a defined path, attain a specified objective, or to put in context its measures of the surrounding area.

As seen previously, there exist technical solutions to implement global localization system (e.g. GPS). Fusing the information of others sensors with a global localization system the uncertainty in the position with respect to a global reference framework can be reduced by a factor 5 or more.

Without a global positioning system integrating the measures of the vehicle (know as "dead reckoning") will lead to growing unbounded errors. The core idea that allows to bound the errors in position is to put in relation the actual observations with past observations. When relating current observations with a prior map we talk about localization, when relating current observations with past observations using a method to bound the error, we talk about SLAM (see section 3.6).

Matching observations with a prior map is a common problem in various domain, it is a pattern recognition problem where many possible patterns are available (somewhat like biometric systems), in medical applications sparse measures are matched to organs/bones models, it is similar to recognize a figure in a video sequence, etc...

The localization problem itself is commonly separated in two problems: to find out the global location in a map without prior knowledge about the actual pose (also know as the "hijacked robot problem" or the "lost robot problem"), to correct the global location in a map using a previous valid approximation (e.g. the last know pose).

Imagine yourself in an foreign city with only a map where there are no tags in the streets and where you can ask to no one your location. To find your position you need to explore the streets and match the observed portion of the city with the map until you can pinpoint unambiguously your location. Using this method the "lost robot" problem can be solved, but clearly it is a non sense to do so for city sized localization. In this text we will suppose that for outdoor problems a rough global localization is available through user input, GPS measures, or, via messages passing between localized cars and disoriented vehicles.

Data association, data representation The localization is essentially a *data association* problem. How to put in relation (associate) the actual measures with the a priori information (the map) ? The answer to this is closely related to how the map is defined. In order to solve the localization problem we need to address the related *data representation* problem, i.e., how to represent the rough measure and prior maps in order to manipulate more easily the data. Depending on how we represent the knowledge the data association can be more or less easy, robust and effective. The data representation problem is also common to the SLAM and SLAMMOT topics because they also deal with manipulation of maps and association of new measures.

In general, to localize globally from one observation is very unlikely, even in such a case repeated observations allow to enhance the location precision. Commonly, a prob-

abilistic approach is employed to estimate the location from the group of past observations, posing the problem in a Bayes filter formulation. In the following subsections we will present the different data representation options, the algorithms used to relate the measures to such representations and then the different methods to estimate the location using this association.

3.5.1 Data representation

The data representation is an important topic in the localization, SLAM and SLAMMOT problems. The choice of a representation is related with both in theoretical aspects (setting assumptions) and practical aspects (defining computational requirements and implementation complexity). In practice the data representation will affect the efficiency (meeting of real-time requirements, memory usage) and efficacy (more or less valid assumption) of the methods and thus is a strong differentiating element.

We now present a list of the different data representations found in the literature and the basic concepts behind each one.

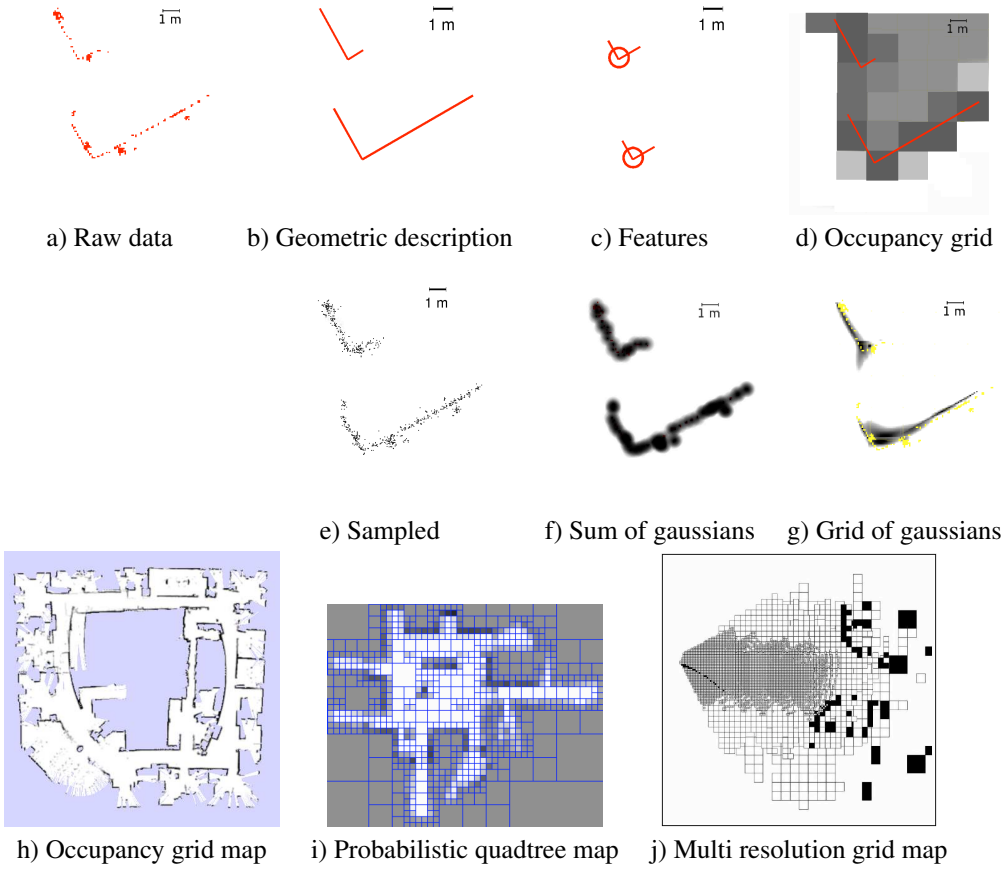
Localized features In the simplest case the world is described by a set of precise, localized, and uniquely identifiable tags (vision features, laser, sonar or IR tags, etc). This is a common supposition of mathematical formulations of SLAM. If tags are uniquely identifiable there is no data association problem and then the localization is trivial. Using a Bayes filter allow to reduce the uncertainty associated to the tags measures or to reject false measures.

If the tags are measurable but not uniquely identifiable then special methods have to be employed to solve the localization with unknown data association problem. Such common features are image point of interest, detected corners, specific objects detectors (e.g. posts or semaphores), scanned objects, etc...

The localized features method has a very compact representation of the world. However this method supposes that such tags are available which commonly requires special installations or strong suppositions on the environment. Localization methods based on features map are sensible to errors in the features recognition.

Geometric descriptions This corresponds to the common maps that represent the world as sets of lines, circles or other basic geometric constructs. Robots measures are analysed to estimate basic geometric forms that are correlated to the topological map.

This representation is conceptually similar to the localized features with unknown data association. Maps of buildings are commonly available in this form, however there is no guarantee that the robot will be available to measure the described objects (this is different to the localized features that are defined as measurable). Through range scans the robot can estimate surfaces to correlate with the map.



a) Raw data b) Geometric description c) Features d) Occupancy grid

e) Sampled f) Sum of gaussians g) Grid of gaussians

h) Occupancy grid map i) Probabilistic quadtree map j) Multi resolution grid map

Figure 3.14: Some representations of the static environment

This method is not very good because it is limited to environment composed uniquely by simple geometric objects, and the analysis of the measures is troublesome and not very robust.

Occupancy grids This is one of the most common approaches. The explorable space is discretized in a finite grid of cells (habitually rectangular cells) where each cell stores a probability of occupancy. Probability 0.5 reflects no information about the cell, probability 0 indicates absolute certainty that the cell is free of obstacles and probability 1 the opposite (see figure 3.14).

In this representation it is easier to integrate uncertain measures to construct maps, and it is possible to correlate measures (using sensors models) to existent maps.

This method presents some limitations due to the extensive use of memory to store regions with good enough resolutions. Also the discretization of the space introduce artifacts in the correlation, specially when estimating the orientation of the robot.

Introduced fifteen years ago [105], this simple and clear method still is an interesting option for real world implementations. Occupancy grids are sometimes appealed as "occupancy maps" or "evidence grids".

Quadtree One the main drawbacks of the occupancy grids is their large memory use, even when most of the region are not accessible. To solve this situation a "probabilistic quadtree" approach was proposed [106]. Instead of decomposing the space in a regular tree it is divided in a sequence of regular area subdivision. This sequence of subdivisions is stored in a tree where each node corresponds to a division and each leaf to a final area (see figure 3.14). Storing the probability of occupancy of each leaf leads a to representation similar to occupancy grids but with a more efficient memory usage. Non measured regions are kept non divided and thus the large "blind areas" are compacted.

The gain in memory space is evident, however it is not clear how efficient is to employ such a structure to, for instance, correlate measures to the map.

Multiresolution pyramid Another limiting aspect of the occupation grids is the regularity in the discretization of the space. For scans correlations this is not troublesome, but when using the map for robot planning a multiresolution approach is more adequate. In [107] the author propose a representation with a pyramidal series of multiresolution maps. The use of different resolutions allow to enhance the calculation of trajectories and to solve an important flaw of rigid cells. In rigid cells if an object is not detected when far measures are realized this will add evidence of free space. However it is possible that the sensor is not able to detect small objects up to when they are near. Using different grids (with different resolutions) to store measures in different distances ranges solves this important issue.

The computation cost of this representation is not much higher, however the increase in memory usage is considerable. The important issue it solves, the ease for navigation

and the fact that multiresolution of maps is usually required anyway (for higher level analysis) make it an interesting option for real world implementations.

Graphs Geographical information systems commonly represent roads information as a graph. At the city level the space is described by nodes and edges that describe both a logical network and a macroscopic geometrical description of the streets. This representation are mainly used with satellite based positioning systems since it is difficult (but not impossible) to put in relation the robot measures with this world representation.

Former research at indoor localization, when computational resource were scarce, used a graph representation of indoor corridors. In that representation each node represents a distinguishable place and some measures associated to it, and the edges represents the existence of connecting paths. Such a graph could be useful to relocate the robot when reaching one point to another. This method is not very used today because more dense representations allow better localization.

Raw data If city graphs is the most abstract map, raw data storage is the least one.

When measures correspond to high resolution range and angle acquisitions it is possible to represent the measure by a set of points. Thus a map can be constructed by the simple aggregation of measured points. Of course this is a simplistic brute force approach, because the amount of data recollected can grow very fast creating fat maps.

Anyway this concept can be used as a starting point. Having a cloud of points it is possible to correlate measures to the cloud and thus localize the robot. Also, the cloud of point can be analysed for decimation or transformation.

Of course the important memory usage and the lack of a precise representation of the uncertainty in the measures do not make this approach very popular.

Sampled Environment Map A manner to represent the uncertainty of a punctual measure is to use samples of the probability density function of the sensor error. Projecting all the samples of the successive measures in a plane creates a sampled distribution of the spatial occupancy's probability density function [108]. Even if the memory usage can be considerable (depends on the number of samples per measure and the number of measures) this technique leads to an accurate representation of the detected objects, in the sense that it correctly represents the uncertainties of their localization. Furthermore this data can be analysed to construct other, more compact representations (e.g. segment representations [109]).

One the most obvious problems of this representation is that the amount of data to realize the correlations is too important for practical online applications. The memory usages seems to be simply wasteful and more compact options have to be evaluated.

Sum of gaussians An idea to reduce the amount of data of sampled environment map is to approximate the sampled distribution by a Gaussian one (i.e., approximate the measure error by a Gaussian distribution). The aggregation of sampled points in the plane is represented in a "sum of gaussians" form [110]. This representation leads to a more controlled memory usage and to efficient correlation methods (because the local gradient can be computed directly).

In the same work the author suggests to divide the sum of gaussians maps into different objects that can be used as local features, transforming thus the kind of representation (scans are matched to objects, thus recuperating their position and orientation). This idea is interesting when the environment is composed by clearly separable objects, which is not the common case in urban areas.

Unfortunately this work left open the problem of merging multiple scans to avoid the grow of stored gaussians. This limits the accuracy of the maps or make grow the computational cost during the correlations.

Grids of gaussians Recently a new data representation approach named "Normal distributions transform" has been proposed in [111]. This method is closely related to the works presented in the last two subsections. The core idea is to represent the environment with a fixed density of overlapping gaussians. The set of previous measures are used to estimate local and overlapping gaussians creating a smooth and accurate representation of the spatial occupation probability.

Fixing the density allow to control the precision of the representation and to limit the memory usage. It is also possible use optimized correlations methods (because local gradient can be calculated) to provide real-time performances in localization and mapping.

This data representation concept merge many interesting properties making it one of the best candidates for real world deployment.

Hierarchical representation It is possible to group all the qualities of the presented data representations using a hierarchical representation [112] where different methods are used at different levels of abstraction.

In [112] the representation is divided in three classes: feature based, grid based and direct methods. Having noticed that non of the class of representation is good enough for the pretended application (city sized SLAM) the design of a hierarchical method is proposed. [112] uses direct methods to correlate successive scans (estimate the vehicle displacement), grid maps to construct successive local maps and a feature based representation for the construction of global maps (composed by the local maps).

The stratification of the representation (and the associated information processing methods) simplify the analysis of the problem and make large scale systems tractable.

3.5.2 Data association

Data association is one of the important issues to solve in the development of perception systems. This issue appears at different levels. During sensor fusion it is required to associate different measures to the same source, in localization current measures have to be associated to data in the map, in SLAM current measures are related to past measures and in SLAMMOT there is additionally the association of measure to the tracked objects (which can be cluttered or changing its dynamic).

In this section we will present data association methods employed for localization which are also pertinent for the SLAM problem. The data association of multiple moving objects (targets) is a more complex problem that will be discussed in section 3.7.

When using a feature based data representation if the features are uniquely identifiable then the association problem is trivial. In the other case we will talk about the "features association" problem. When using dense measures data representation the problem has a different nature and we will call it the "scan matching" problem.

Features association

When detecting features in the environment each new observation can be classified as:

- the measure of an already observed feature (a landmark l_i)
- a new feature that entered in the field of view
- a spurious measurement

The feature association is then essentially a classification problem. Many techniques have been proposed to solve it.

In the worst case feature association is an exponential problem where all the possible combinations are tracked during the time required to reduce the uncertainty down to an acceptable level. In the simplest case landmarks are very sparse in relation to the measurement error and one to one association can always be done, then the problem is trivial. The characteristics of the problem depends of the particular case.

Nearest neighbour The most naive approach to data association is to choose the nearest measure to the expected landmark place, without considering exclusivity between measures and landmark (one measure could be associated to more than one landmark). This naive approach will fail when a measure from a landmark miss.

Gating To cope this case the common approach is to allow only near measures as valid. This is named "gating". Having a distance measure d, being μ_{l_i} the expected location of the landmark l_i and being z_k one of the point measured in the environment, then z_k will be associated to l_i only if

$$d(z_k, \mu_{l_i}) \leq G$$

where G is the gate parameter. If two or more measures are in the acceptable region then the nearest one is chosen or a weighed average is taken as the valid result.

The distance measure used is arbitrary, in general a good choice is the Mahalanobis distance (distance that weighs the dimensions) between the measure and the gaussian $\mathscr{G}(\mu_{l_i}, \Sigma_{l_i})$ that describe the uncertain knowledge of the landmark location. The distance measure becomes:

$$d^2(z) = (z - \mu_{l_i})^T \Sigma^{-1} (z - \mu_{l_i})$$

Then the gate G can be calculated to cover an arbitrary percentage of cases (e.g. 95% of all the correct cases). The gate for d^2 corresponds to a χ^2 distribution.

It is also possible to define a distance that includes both the uncertainty in the expected position and the measure uncertainty.

Even if gating can be seen as a simplistic approach it is very used because of it principled simplicity. It is also always used in more complicated approaches to purge the very improbable cases.

The concepts presented up to now can be found under the name of "nearest neighbour", "gating", "strongest-nearest neighbour", "probabilistic nearest neighbour", "maximum likelihood data association".

PDAF When more than a measure is accepted by the gating, another method (known as "probabilistic data association" or "all-neighbours approach") prefers to collapse the measures into one unique result, combining both the location of the measures and the uncertainty associated to them. Tracking a single object in movement when the measures are collapsed using this approach is called "probabilistic data association filter" (PDAF).

The classic reference on the data association (for single and multiple target tracking) are the works of Bar-Shalom [113, 114].

For localization in feature-based maps these simple methods are good enough most of the practical cases. There are more sophisticated approaches, that can be applied in more difficult scenarios, for instance, when tracking simultaneously multiple moving objects. We will see these more advanced methods in the section 3.7.2.

Scan matching

When the measures of a range and direction sensor are dense measures (a scan) then the data association problem is called "scan matching". A scan has to be matched to a prior map of obstacle or two scans have to be matched between them. The association process returns the relative location and orientation of the scan with respect to the reference (map of previous scan).

The scan matching problem is a common problematic in 3D reconstruction and some medical applications (registration of a patient model with respect to a real-time measure). Methods from such domains are applied to the robot localization and conversely.

ICP The most classic algorithm for scan matching is called ICP, Iterative Closest Point, and was originally developed for 3D reconstruction applications [115]. An equivalent algorithm was used for first time in the robotic domain by Zhang [116] (in this domain it also know as Iterative Point Matching or Iterative Dual Correspondence algorithm).

The most valuable characteristic of the ICP algorithm is its simplicity. Starting with two clouds of points, an initial guess of the relative transformation (translation and rotation) and a stopping criterion the algorithm iterate over the following operations:

1. Associate the points of one scan to the points of the second one (using a nearest neighbour criterion)
2. Estimate an optimal translation and rotation to minimize a cost function (mean square)
3. Transform the points using the estimated parameters
4. Iterate (re-associate the points, search new displacement, etc...)

This simple algorithm generated a big number of variants where each step was tweaked to some purpose. Authors proposed methods to reduce the number of points considered (trying to select only relevant points in order to accelerate the computation), alternatives to the point matching step (simple distance, adding robustness to outlier, using slope information, forcing one to one association, etc...), to the cost function (weighting the points under a goodness criterion), introduced steps to enhance the robustness, and applied domain knowledge to accelerate the convergence during the optimization. A good compendium of variant methods can be found in [117].

All these alternatives are interesting and some of them provides considerable speed and robustness enhancement over the original algorithm (see for instance [118]). However their assumptions are not well adapted to the robotic case. The most evident flaw is that the measurement uncertainty is not taken in consideration. This is specially evident in long distance measures (some meters) where the angular resolution introduce a considerable effect in the measure uncertainty. This omission in the noise modelling

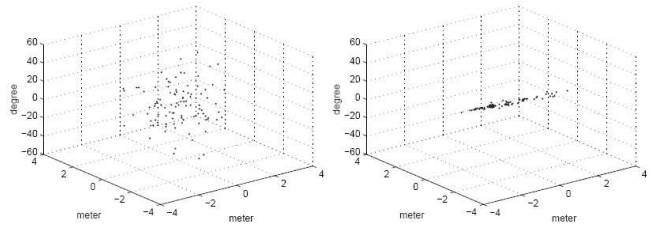


Figure 3.15: Sampling based uncertainty estimation . Left: randomly generated initial transformation samples (translation and rotation). Right: the transformation estimates after applying the registration algorithm. From [17]

affects the robustness, precision and speed of convergence of the matching method, however it does not prohibit to use this method in the robotic domain, where it remains one of the most used algorithms.

Matching as argmax Another popular approach consists to setup the matching problem as a maximum likelihood problem [119, chapter 2]. Having an occupation map m_t, a previous robot pose x_{t-1} and a scan measure z_t, the question is to search most probable actual pose $\hat{x}_t$ that would better explain the measure z_t. This can be written as

$$\hat{x}_t = \text{argmax}(x_t)\{P(z_t|x_t, m_t) \cdot P(x_t|x_{t-1})\} \tag{3.5}$$

The probability is then maximized using a hill climbing algorithm. This method is more principled and can be faster because it does not require the point to point association steps. However it supposes that the initial guess is within the global maximum region and the result gives no clue about the uncertainty in the resulting value $\hat{x}_t$. This is the approach commonly employed when occupancy grids are employed for data representation. In this case an additional limitation appears do to the spatial discretization of the grid and its effect on the estimation process.

Uncertainty estimation In order to estimate the uncertainty in the result of the ICP algorithm or variants, it has been proposed [17] to use the large computational power available to execute N time the ICP algorithms with the same scans but using different initial guessed for x_t. Sampling a normal distribution around the best prior guess, the N results obtained will permit to quantify the input sensibility of the output, and thus the uncertainty in the most probable result $\hat{x}_t$ (see figure 3.15). Of course this method is computationally intensive, but with today computers it is possible to implement it for real-time applications.

Searching more principled approaches and trying to bypass the difficulties of the ICP or grid map approaches Bailey [110] proposed the usage of sum of gaussians to represent both the reference map and the current scan. Using this representation the matching problem becomes a problem of maximum likelihood of the cross correlation of two

sums of gaussians [120]. The computation of the cross-correlation (and the search of its maximum) is potentially computer intensive, however some simple optimization and minor approximation make it tractable. This sounder approach allow to correlate the scans without using point to point association or quantizing the space, which is clearly an advantage. Also this method allow to estimate a Gaussian distribution around the maximum likelihood result, allowing to use it in probabilistic mapping methods. However this method has the problem that, as mentioned in the previous sections, the sum of gaussian is not a suitable representation for map building, and the maximum likelihood computation, even if tractable it can threaten real time implementations. It worth mentioning that as any maximum likelihood approach, this method is sensible to multimodal distributions, that can be commonly found in the computed cross-correlation. The original author proposes a particle filter method to cope this issue [110].

When using the "Normal distribution transform" [121, 111] for data representation (see "Grids of gaussians" in the section 3.5.1) a similar approach to correlation of sum of gaussians can be used. The problem of equation 3.5 can be then formulated as a convex optimization problem solved through a gradient descent. This approach is of course very effective, in [121] the author report a time between 4 and 20 milliseconds for aligning two scans.

All the mentioned scan matching algorithms suppose only rigid transformations between the scans (as it is common case in urban scenarios). When this assumption is not true (as could occur in medical applications) more free parameters have to be considered in the optimization process. [122] proposes an extension of the ICP algorithm for non rigid objects matching, applied to the registration of 3D objects (human bodies in particular).

It exists a relation between the scan matching methods and the data representation chosen. This aspect has to be considered when choosing the representation. Having a data representation and a data association method, it is possible to localize the robot in the map, and to extend maps with the measures of the robot.

3.5.3 Indoor and outdoor localization

Having a previous map (in a chosen data representation) and being able to correlate the robot measures with this map (data association), then the localization problem becomes a Bayesian estimation problem. Notice that this perspective is generic, independent of the kind of map, the kind of features detected or the kind of sensor available, the localization remains modeled in the same way.

Composing positions

To realize the data association the robot first estimates the location of its observation in the global map compounding the uncertainty of its actual position and the uncertainty of its measure.

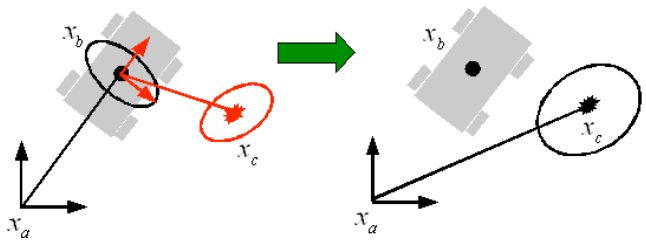


Figure 3.16: Compounding location uncertainty and measure uncertainty. When passing from the vehicle reference frame to the global reference frame the uncertainty on the position of the observed feature augments

Using the notation introduced by Smith [123] and employed by Lu, Milios and Wang [124, 17], we will denote this compounding operation as

$$\text{landmark\_estimation} = \text{robot\_position} \oplus \text{observation}$$

using a more orthodox notation, related to the figure 3.16, we can write

$$x_{ac} = x_{ab} \oplus x_{bc}$$

As the robot position x_{ab} and the observation x_{bc} both contain uncertainties in their value, it would be more adequate to rewrite this in a probabilistic notation. The density of probability of the position x_b with respect to the reference x_a can be wrote as $P(x_b|x_a)$, which is the probabilistic notation for the deterministic vector x_{ab}. The compounding operation can be written in terms of densities of probability, resulting

$$P(x_c|x_a) = \sum_{x_b} P(x_b|x_a) \cdot P(x_c|x_b) \tag{3.6}$$

This generic relation allow us to compound arbitrary distributions. As illustrated in the figure 3.16 the uncertainty of the observation with respect to the global reference is always bigger than with respect to the robot.

Using the landmark position estimate $P(x_c|x_a)$ a data association method is used to associate it with a specific, already mapped, landmark l_i. The position of the identified landmark has an uncertainty $P(x_{l_i}|x_a)$ which is related to the precision of the map used. Anyway this uncertainty is expected to be low because the map was constructed from multiple measures or with higher precision sensors.

Once data association was done, using the measure $P(x_c|x_b)$ and the landmark position $P(x_{l_i}|x_a)$ it is possible to estimate the robot position in the global reference frame $P(x_b|x_a)$. Using the position algebra notation we can do this using the compounding operator $\oplus$ and the inverse relation operator $\ominus$ (see figure 3.17). Then we can write

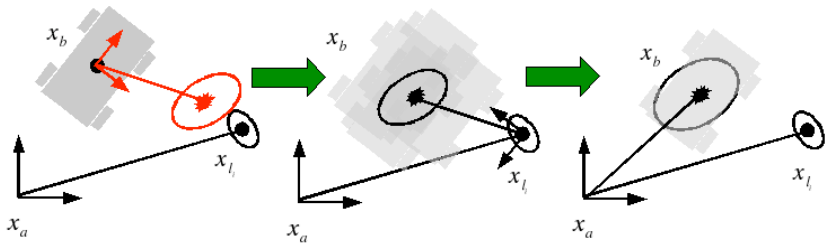


Figure 3.17: Deducing the robot position from one observation of an identified landmark. Once data association is done, the vehicle position uncertainty in the global reference frame results from compounding the global position of the feature and the relative measure position

$$x_{ab} = x_{al_i} \oplus x_{l_i b} = x_{al_i} \oplus (\ominus x_{bl_i})$$

or in plain English

$$\text{robot\_pose\_estimation} = \text{landmark\_position} \oplus (\ominus \text{observation})$$

The results of the compositions (that could also be written in probabilities notation) gives a new estimate of the robot position $P(x_b|x_a)$ that can be used as a measure in a Bayesian filtering process (see section 3.4.4). Even when the sensor precision is low or the landmark position uncertainty is considerable the important thing is that this estimate provides a *bounded uncertainty measure* of the robot position.

Integrating repeated measures allows to have a good estimation of its position (as long as the sensor has zero mean error). In the limit (infinite measures) the error of the robot position will be the one of the landmark in the map, $P(x_{l_i}|x_a)$.

In practice the uncertainties are usually represented as gaussians, the position operators corresponds to trigonometric transformations. The reader can consult [123] and [17, section 2.1] for the detailed equations. As a side note, we should mention that gaussians can be rotated exactly using nonlinear equations [125, appendix A], however most of the time this transformation is simply linearized, introducing additional error in the estimates.

When the map and the measures are based on range scans instead of features poses, the position estimation is used to collect an area of the map (or a set of previous scans) that could match the actual scan and a simple scan matching is realized. Then again the merge of the previous map uncertainty with the scan matching uncertainty leads to a position estimation, and thus data fusion can be applied.

The important result from the analysis presented is that once the data association is realized and the previous information is used the localization becomes a data fusion problem. It is important to realize that *localization contains a Bayesian filtering problem*

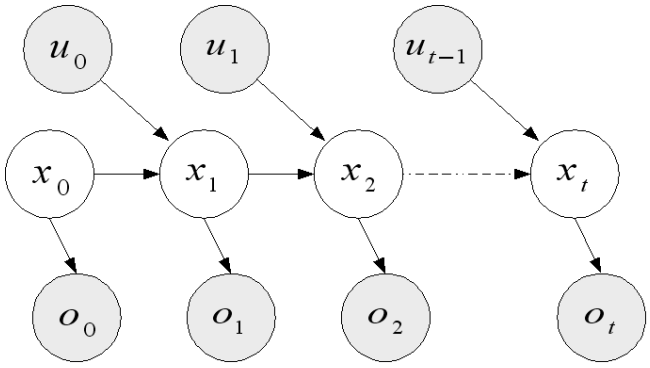


Figure 3.18: Markov localization Bayesian network

and that Bayesian filtering problem contains a data fusion problem. This hierarchical relations repeats all along this chapter.

Markov localization

Simple localization use prior data and data matching to reduce the localization problem to a data fusion one in order to estimate the actual state of the robot (pose and orientation). Common measures, apart from the one matched to the map, are odometry and GPS (as seen in the previous sections). Another useful information commonly available is the command u_t applied to the robot. This command indicates how we want to displace the robot and thus give important information about its probable next position. When considering the command the robot model becomes $P(x_{t+1}|x_t\ u_t)$.

When including this information in the Bayesian filtering the localization method is called "Markov localization", however it is only a specific case. We present in figure 3.18 the dynamic Bayesian graph related to this case, and the iterative solution in equation 3.7. The Bayesian program can be deduced from this two elements.

$$P(x_t|o_0 \dots o_t\ u_0 \dots u_{t-1}) = \alpha_t \cdot P(o_t|x_t) \cdot \sum_{x_{t-1}} P(x_t|x_{t-1}\ u_{t-1}) \cdot P(x_{t-1}|o_0 \dots o_{t-1}\ u_0 \dots u_{t-2}) \tag{3.7}$$

Monte Carlos localization

Notice that a fusion based approach supposes that we have a prior estimate about the robot location. To solve the "hijacked robot" problem, a simple idea would be to impose a uniform prior in all the explored area.

In the common applications the distributions of probability are represented by unimodal gaussians and the observation and evolution models approximated using simple

linearization or sigma-point approximations (unscented transform) [101, 104]. Since normally only one observation does not lead to a unique position in the whole map, the unimodal representation of the probable location of the robot does not allow to solve the "hijacked robot" problem. Using multimodal distributions increments the computation, but allows to manage various hypotheses until the information collected indicates a notably more probable location than previous hypotheses. Classic multimodal implementations of Bayesian filtering are mixture of gaussians and particle filters. When applying particle filters for localization is called "Monte Carlo localization" (since a particle filters is a "Sequential Monte Carlo method"). A good tutorial of particle filter for robot localization can be found at [102].

Selection criteria

The Bayesian approach to localization is very generic and essentially indifferent of the nature of the environment. What makes a method suitable or not for outdoor or urban areas are the suppositions employed to make tractable the computation, the data representation chosen and the robustness of the data association methods.

Simplistic approaches would approximate every density of probabilities by unimodal normal densities, linearize every non linear function, use corner detection features as data representation and use nearest neighbor for data association. Of course such simplifications will be not usable outside static, simple indoor corridors.

Robust approaches would use particle filters for density distributions and estimation considering nonlinear models, occupancy grids or grids of gaussians for representation and sampled correlations to estimate the density of probability of the data association. Of course such an approach is extremely computer intensive (both in memory and computation length).

A review of the different variants (data representation, data association, filtering method) of the Bayesian localization and some experimental results about them can be found at [126, 127] and [128, 129].

3.6 SLAM

Mobile robots are supposed to automatically displace themselves in order to accomplish a task. To decide the movements to execute and to close the control loop during their execution the robot needs to constantly know which obstacles surround it, and where it is located on the space with respect to its planned path.

Localization is based on the use of current robot's measures and a previously existing map. In most of the real world cases maps of the environment to explore do not exist, are not accurate enough for the task, or simply represent information that the robot cannot use efficaciously. Expecting to have precise and adequate maps of the environment before robots deployments is a constraining requirement.

In human modified environments robots should be able to construct their own maps. This has two clear advantages: it is not required to provide a previous map and the map constructed by the robot is already well adapted for its tasks. Usually maps created by robots can also be interpreted by humans.

As we will explain in the next paragraphs the SLAM acronym : "Simultaneous Localization And Mapping", stands for the problem of constructing a map of the environment starting from measures. While constructing the map, the robot also obtains information about its localization. The SLAM problem is also sometime called CML or "Concurrent Localization and Mapping".

In the context of the SLAM problematic it is supposed that the environment is *static and immutable*. Dealing with moving obstacles is discussed in section 3.7. Dealing with changing environments remains an open problem [130]. Having accepted this basic assumption we will discuss the resting difficulties, how the SLAM problem can be formalized and which are the existing approaches to solve it.

3.6.1 Core issues

Imagine a noise free, deterministic, perfectly measurable and perfectly modeled world. In this world the robot does measures of the objects around it and measures its displacement.

As the world is noise free and the deterministic robot dynamic is perfectly modeled we know exactly where the robot is located with respect to its previous location (movement in the Cartesian plane and change of the orientation). Then simply superposing the different measures would create a map as accurate as the environment sensor (supposed perfect).

In such an imaginary world simultaneous localization and mapping, is not a problem.

Displacement measure noise Let suppose now that the estimation of the robot displacement has a considerable amount of uncertainty (imperfect models, finite precision of measures, noise in the physical phenomenas), however the environment measures remains almost perfect.

In this new scenario the robot needs to analyse its data in order to accomplish its localization and mapping task successfully. Simply using the displacement measures would lead to a continuous unbounded increment in the localization error, which will also affect the quality of the map.

In order to keep a bounded estimation of its current position the robot needs to relate previous environment measures with the actual one. If the robot is able to make a data association between previous environment measures and actual measures it is able to estimate its displacement. Thus successive measures associations allow the robot to have successive estimation of its position, and at the same time allow it to interconnect the different environment views, i.e., to construct a map of its environment.

If the measures of the environment are precise enough for the application, data association is a core issue for successful localization and map construction. It is also necessary to have correct uncertainty models of the robot displacements and the measures to create good hypotheses about the possible data association targets.

Notice that localization and map construction are entangled. Solving one problem lead to the solution of the other one, and conversely. SLAM is not two problems, but one problem with two dual forms.

Environment measure noise If the displacement estimations are precise but the environment measures are not, for a while the robot will be able to construct a rough map. However if data association between environment measures is not made, the error in the position estimation will grow slowly but without bounds, leading finally to unacceptable errors. Data association between various measures of the same point allows to fusion them reducing the uncertainty in the position of the point and thus allowing to construct more precise maps. Of course the possibility of repeating measures of the same point in the space will depend on the kind of sensor and signal processing used.

When both displacement and environment measures are imprecise the data association between successive observations becomes harder since more configurations will seem plausible. *Accurate representation of the uncertainty, pruning of low probability hypotheses, and detection of wrong decisions are key elements for the success of SLAM in noisy conditions.*

Even when data association between successive measures is made with success, a small error in the estimated displacement still existing (related to the limitations of the environment sensor, the data association method, and the number of fused measures). This small error is accumulated in time (just like odometry integration) making more difficult to associate the actual position with previously observed areas if no correction is made during a long period. This difficulty is called the "loop closing" problem, or "data association in the large".

Each time a data association is made a relation is established between different landmarks. This geometric relation is corrupted by the measurements noise. As the robot explore the environment different landmarks are observed and the relation between them is established. As the construction of the relation set is incremental, the position of the first landmark is somehow related to the position of the last landmark. If a correction over the position of the first landmark is made, then the position of the last one should be also modified. This relation is valid for every landmark in the map. Thus if when a new observation is made the ideal computational cost would be $O(N)$ where N is the number of landmarks in the map. Supposing an homogeneous distribution of landmarks in the space, the cost of map updating would be in the best case, linear with the size of the map. Trivially, the memory usage of any SLAM algorithm should also be $O(N)$.

As we will see in the next sections achieving $O(N)$ algorithms was for a long time a utopian target. Common algorithms use $O(N^2)$ for memory and computation, or even $O(N^3)$ computation time in some initial approaches. The quadratic aspect of the

SLAM solution appears because, as mentioned, the position of a landmark is related to the position of all the other ones. Thus, managing explicitly these relations creates algorithms with $O(N^2)$ complexity. A deeper insight into the SLAM problem structure allows today to propose algorithm with $O(N)$, $O(log\, N)$ or even $O(1)$ complexity when some approximations and trade-off are made.

Achieving acceptable computation complexity in SLAM algorithms is a non trivial issue. Even if computational complexity is low, time of computation in real implementations can be high. Thus, in general, the computation cost of the SLAM algorithm is a major concern for real time implementations.

In this section we presented the core issues of the SLAM problem:

- Uncertainty management (modelling, data representation, sensor fusion)
- Data association (in the small, in the large)
- Computational cost

In the next sections we will present a more formal view of the problem and the different approaches to solve it.

An important idea to keep in mind: *it exists a duality between localization and map construction*, this duality is the essence of the SLAM problem.

3.6.2 Problem formulation

From a conceptual point of view the SLAM problem is a "most probable explanation" problem (MPE problem). Given all the observations, the robot try to deduce the most probable path and the most probable map (deciding one fix the other).

As the robot moves its state x_t evolves in the time, its internal sensors allows it to do measures m_t of its displacement and its external sensors allow it to do measures of the environment z_t. Of course, the current observation z_t will depend of the environment real map M and of actual robot pose described by x_t. The SLAM problem consist on estimating the real map M and the current robot position x_t using only the available measures $m_{1...t}$ and $z_{0...t}$. In the figures 3.19 and 3.20 we present the associated Bayesian Network and the Bayesian program, respectively.

The robot command signal $u_{1...t}$ was omitted to lighten the nomenclature, and because is not relevant in the discussion. It inclusion changes the term $P(x_i|x_{i-1})$ by $P(x_i|x_{i-1}\ u_i)$ allowing to reduce the uncertainty of the robot position prior to measurements inclusion.

To denote the reconstructed map at time t we will use the notation M_t.

$$M_t = p(M|\ z_{0...t}\ \ m_{1...t})$$

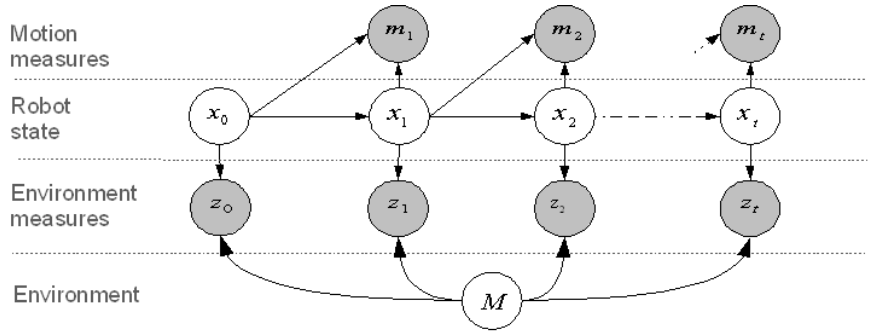


Figure 3.19: The SLAM dynamic Bayesian network. Notice that the environment does not change in the time

Prog

Desc

Spec

Variables

$x_0, \ldots, x_t = x_{0\ldots t}$	Robot state vectors in time
$m_1, \ldots, m_t = m_{1\ldots t}$	Robot motion measures in time
$z_0, \ldots, z_t = z_{0\ldots t}$	Measures of the environment in time
M	Map of the environment

Decomposition :

$$P(x_{0\ldots t}\ M\ m_{1\ldots t}\ z_{0\ldots t}) = P(x_0)P(z_0|x_0\ M) \prod_{i=1}^{t}[P(x_i|x_{i-1})P(m_i|x_i\ x_{i-1})P(z_i|x_i\ M)]$$

Forms : any

Identification : any

Question :

$$P(x_t\ M|m_{1\ldots t}\ z_{0\ldots t})$$

Figure 3.20: The SLAM problem

The Bayesian program of the figure 3.20 is a very general description of the SLAM problem. It does not specify how the map M will be represented and gives no clue about how its estimation is supposed to be done. In the next section we will present an overview of the most relevant solutions to the SLAM problem and discuss their representation and resolution methods.

3.6.3 Solving the SLAM problem

Why solving the SLAM problem matters ?

As mentioned in the introduction of this chapter solving the SLAM problem is a keystone for mobile robots technology. Understanding the existing solutions serve us as a foundation when approaching the SLAMMOT problem. Both problems have shared difficulties, shared solution elements and are part of the same area of exploration.

Why solving the SLAM problem does not matter ?

Much of the focus on the SLAM community has been to build large, precise and consistent maps. However, as discussed in section 2.6.1 the availability of such maps is of limited use for our application.

Most of the existing work is focused on how to optimize a map given a sequence of associated measures and dismiss the fact that doing the association is comparably difficult. For the work that does account for possible errors in the data association, the relevant corrections are obtained when the robot revisits a place ("closing the loop"), but not while the robot simply move forward (as in our application).

When only concerned on local maps (short term SLAM) simple solutions based on good matching methods and error management work fine enough.

State of the art

Variety of solutions There exist a considerable amount of methods proposed to solve the SLAM problem, resulting from fifteen years of research on this topic. Much creativity have been applied to solve this problem and also to name the solutions (RatSlam, FastSlam, ScanSlam, SpMap, TreeMap, D-Slam, DP-Slam, IPSlam, CpeSlam, CEKFSlam, GMapping, TORO, etc.). The diversity of solutions creates a complex fauna of approaches, methods, vocabularies and results. One solution is different of the other ones in diverse ways.

Different solutions uses different models for the constructed map (see subsection 3.5.1). They also make different hypotheses about the environment (indoor, outdoor, mines, structured, with a defined density of specific features, flat, locally flat), about the available measures (type of sensors, noise model) or about the robot dynamics. Most works

suppose that data association was already carried successfully but some others integrate this critical element into the proposed method. Each work proposes a specific algorithm to construct the map itself, sometimes incrementally (online) or using batch computation (offline), sometimes providing exact solutions at each iteration or providing approximate partial solutions during the processing. Finally, different algorithms result in different computational complexities and different computational costs.

Between all the different works it does not exist a clear comparison method, a standard benchmark or a formal evaluation of the robustness of each method. The most common comparison elements used are the base assumptions, the computational complexity or cost, and the length of the largest closed loop. To help the comparison between different methods some data sets were published [131] and some implementation provided as open source [132], however their usage is not widespread and the evaluation remains qualitative.

Taxonomy Historically the SLAM algorithms have been differentiated by their representation. There was a first stream using feature based representation [133, 123] and another one using the poses based representation [124]. The feature based approach emphasize the fact that the robot moves in the environment and try to maps specific features (landmarks) found on it, the poses based representation use the robot trajectory (its poses at specific instants) as the landmarks themselves.

It can be shown that both approaches are somewhat equivalent since the robot poses can be considered as landmarks by themselves and that the measures of different features in the environment can be used to construct a sequence of poses. The core point is to understand that the SLAM problem relates with a set of local spatial relations, the robot try to get a global figure from a set of local observations. The precise nature of the points related (robot position or feature positions) is not relevant for the analysis since the procedure for solution will be same ones.

The seminal papers [124] and [133, 123] defined the exploration line for most of the SLAM research done in the last years. In this text we will try to present the SLAM algorithms from a slightly different point of view than the common reviews. Following the spirit of the analysis of Udo Frese [134] we propose to group the different SLAM algorithms in simply three groups: the algorithms based on the Covariance Matrix, the algorithms based on the Information Matrix and the algorithms based on Particles. This classification is based on the representation of the uncertainty. As we will see the covariance matrix is the inverse of the information matrix and both classes correspond to the approaches proposed in the mentioned seminal papers. The algorithms based on particles correspond to a more recent approach which finds its best representative in the works of Montemerlo [135].

For the sake of brevity we will not review all of the existing methods in detail. We will content ourselves to succinctly explain the different groups of algorithms, make references to some of their relevant variants and highlight the one we consider most relevant.

Algorithms using the Covariance matrix

As seen in the dynamic Bayesian network of the figure 3.19, the SLAM problem has a time evolving dimension. Thus a natural choice to solve the problem consists to take a Bayesian filtering approach, where the observations are used to estimate unobserved variables (robot position and map). In order to make the Bayesian filtering tractable some simplifications about the model of the system and the probabilities distributions have to be made.

Kalman SLAM One of the simplest approaches consists to solve the SLAM problem using a Kalman filter [133, 123]. The approach is very natural, it provides incremental solutions and for linear models with Gaussian noise it is known to be optimal (in the sense that it computes the best estimation possible).

For Kalman SLAM the state of the world is represented by a state vector W_t that contains information about the robot pose and the landmarks pose. As new landmarks are discovered the state vector is expanded consequently. If the number of landmarks at time t is defined as $n(t)$ then we define the state vector as

$$W_t = \begin{bmatrix} \vec{x}_t \\ \vec{l}_{1,t} \\ \vec{l}_{2,t} \\ \vdots \\ \vec{l}_{n(t),t} \end{bmatrix}$$

In order to sequentially estimate W_t as new observations are realized this method uses linear models with Gaussian noise for the measure relations $P(z_t|x_t\ M)$, $P(m_i|x_i\ x_{i-1})$ and for the evolution relation $P(x_i|x_{i-1})$. It also maintain a covariance matrix Σ_{W_t} that represent the uncertainty of each value in W_t and the correlation between them. For the formulation details the read is invited to consult [123].

The covariance matrix stores the relation between each state variable, it is of size N^2 where N is the number of estimated variables (dimension of the state of the landmarks time the number of landmark plus the dimension of the robot state). In general the update the state estimation has a computation cost $O(N^2)$.

Having a quadratic computational cost is the principal limitation of this method. In practice the areas mapped will be strictly limited by the computing power available. Current implementations can update maps with one hundred landmarks in a few milliseconds.

Another clear limitation of the approach is the use of linear models which are quite unrealistic. For instance, features are commonly detected using range and bearing sensors, where the sensor noise is modeled as independent zero-mean Gaussian over the distance component and over the angle component. Once projected on the Cartesian coordinates the sensor noise does not correspond to a Gaussian.

Relaxing the linear model constraint A simple solution to overcome this limitation is to use first order approximations of the non linear models. This approach is called Extended Kalman Filtering and is commonly used (EKF Slam). However the linearization process introduce errors proportional to the error of the linearization point estimate. If the initial estimate is not good enough the systematic error introduced can make the estimation non convergent [136, 137, 138].

A possible solution would be to recompute the linearization for at each new estimation of the state (this method is named Iterated EKF), however the computational cost is increased. Another approach to deal with nonlinear models is to use a more robust approximation. The Unscented Kalman Filtering [101] use an approximation based on the first and second moments preservation (instead of using a projection over the first derivative), this approximation introduces less error [139].

Reducing the computational burden A common strategy proposed to reduce the quadratic computation time consist on decomposing the state estimation problem into multiple sub-maps estimation one and the composing them together to recreate the full map ($N^2 > (N/2)^2 + (N/2)^2$). This decomposition introduces a reduced convergence rate or a conservative uncertainty representation, however they allow building larger maps.

The idea of decomposing a large map into a set of sub-maps, where a separate estimation is made for each sub-map and a global process merge all the maps together, has been proposed explicitly numerous times [140, 141, 142, 143, 144, 145, 146]. All of these works play with a trade off between discarding information, delaying information propagation and having accurate results. Delaying or discarding information allow to reduce the computation at each iteration, but creates suboptimal solutions. Many of these works try to ensure the consistency of their solution (convergence to a near optimal result), however it has to be kept in mind that data association is realized using current estimations of the world. Having suboptimal estimates creates more ambiguity for data association of new measures and thus, in practice, put in risk successful localization and mapping.

Algorithms using the Information matrix

The covariance matrix approach came from viewing the SLAM problem as an estimation problem. An opposite view was proposed by Lu and Milios [124], where the SLAM problem can be interpreted as an optimization problem.

This come from directly taking the SLAM formulation as a maximum likelihood problem.

$$\hat{W}_t = \text{argmax}_{W_t}\ P(W_t | m_{1\ldots t}\ z_{0\ldots t}) \tag{3.8}$$

After defining the observation model and the uncertainty model the problem becomes a pure optimization one. This formulation provides a solution based on every past

measures; it uses a batch computing and thus does not provide incremental updates of the state estimate $\hat{W}_t$. However, when the optimization achieves the global minimum, this formulation provides the ground truth of the best estimation possible.

Linear formulation When using Gaussian noise with a linear (or linearized) measurement model the optimization problem becomes a quadratic optimization problem of the form

$$\text{argmin}_{x_t} \; f(x_t) = x_t^T \cdot A \cdot x_t + x_t^T \cdot b + c \tag{3.9}$$

Here x_t is a state vector that includes every robot pose and every landmark position, and thus can be considered as an extended version of the state vector W_t. The matrix A contains the relations between the elements of x_t, vectors b and c contain the measurements' information.

The equation 3.9 has its minimum at $x_t = A^{-1} \cdot b/2$, which is found by solving the a large set of linear equations

$$A \cdot x_t = b/2 \tag{3.10}$$

A strategy to deal with non linear measurement models consist on iteratively linearizing, solving and repeating until convergence.

Relation between Σ_{W_t}, $\Sigma_{W_t}^{-1}$, A, and A^{-1} The interesting point is that it exists a relation between the matrix A and the covariance matrix Σ_{W_t}. Indeed the covariance matrix Σ_{W_t} used in Kalman filtering corresponds to a sub-matrix of the inverse of the matrix A used when solving the optimization problem (see figure 3.21). Thus there exist a link between all the optimization based approaches and the Kalman filtering based approaches.

The matrix A stores the relations between all the robot poses and the landmarks, and thus grows as the robot moves (not only when the map grows). If N is the number of landmarks in the map and t is the number of robot poses, then solving 3.10 takes $O((N+t)^3)$ in the general case. It should be noted that A is exactly sparse since it encodes directly the uncertainty from the measures, which is bounded and local. Using the sparseness property the computation cost of equation 3.10 is $O((N+t)^2)$.

The reader can notice that the matrix A does not grow proportionally to the map but proportionally to the path of the robot. This can be quite inconvenient when the robot revisit places because the computation will grow when the map does not. To avoid this the previous robot poses can be marginalized out from the matrix A. This is equivalent to computing a Schur complement over the matrix A. The resulting (smaller) matrix will be named $\Sigma_{W_t}^{-1}$. This reduced matrix corresponds to the inverse of the covariance matrix.

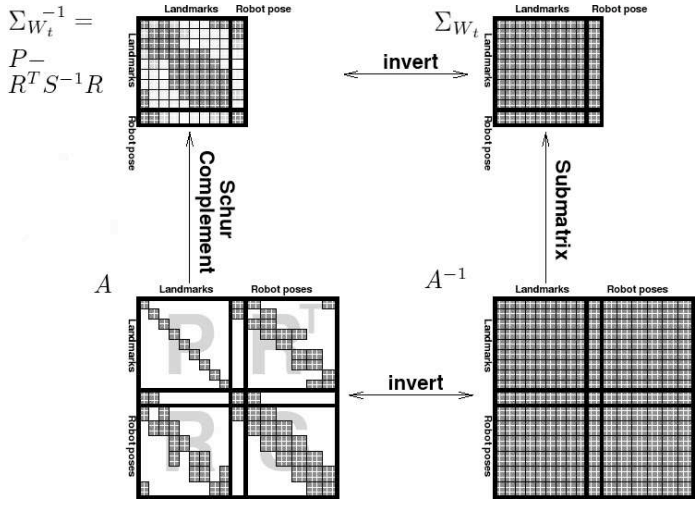


Figure 3.21: Relation between different uncertainty matrices. From [134]

When marginalizing the past robot positions the resulting matrix is not sparse anymore since the different landmarks become increasingly linked as illustrated in the figure 3.22. The entries in the $\Sigma_{W_t}^{-1}$ matrix represents how much information one variables provides about the other ones. Because the relation between measures decays exponentially with the distance except when closing a loop the matrix $\Sigma_{W_t}^{-1}$ will become nearly sparse (most of the off diagonal elements have near zero values). Figure 3.23 illustrates a covariance matrix Σ_{W_t} for a real map and its corresponding inverse $\Sigma_{W_t}^{-1}$.

The matrix $\Sigma_{W_t}^{-1}$ is called the *information matrix*. It exist a dual formulation of the Kalman filter (which uses Σ_{W_t}) named the Information Filter that can be applied directly over $\Sigma_{W_t}^{-1}$.

Variations of the linear formulation Multiple variations on the naive formulation of equation 3.9 exist. We have seen that $\Sigma_{W_t}^{-1}$ is a reduced form of A. A naive application of the Information Filter implies a cost of $O(N^3)$ per update. Thrun [148] and Paskin [149] propose methods to sparsify the information matrix in order to reduce the computation time towards $O(N \cdot \log N)$ or even $O(1)$ at the cost of information loss. Eustice [147] proposes another variant for A that includes explicitly the robot poses in order to construct a strictly sparse matrix and obtain $O(N)$ and $O(1)$ update cost. Wang [150] instead propose to separate A in two sparse matrices, one for the landmarks and another for the robot positions, obtaining then $O(N)$ update time. All of this method differ in the kind of approximations used to reach memory and computation scalability.

Enhancing the optimization procedure Instead of modifying the optimization problem, some works have focused their effort on enhancing the optimization procedure.

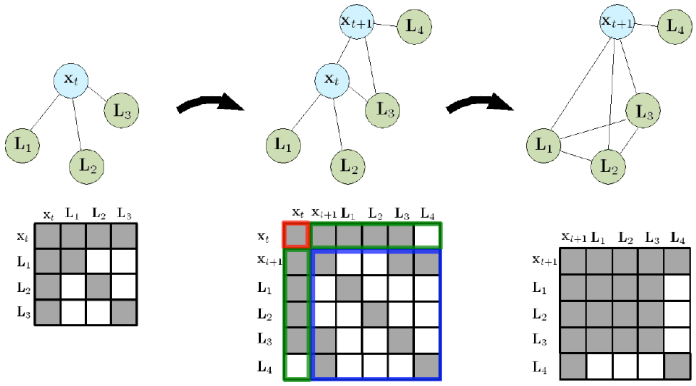


Figure 3.22: Illustration explaining why the information matrix is dense. x_t corresponds to the robot pose at time t and L_i corresponds to the pose of landmark i. From [147]

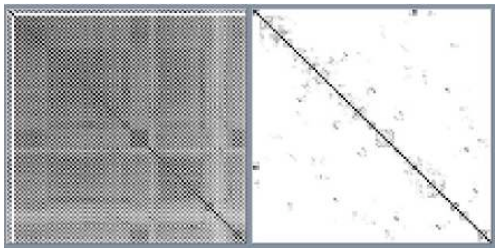


Figure 3.23: On the left size the values of the covariance matrix Σ_{W_t} are shown. On the right the equivalent information matrix $\Sigma_{W_t}^{-1}$ is shown. From [148]

Konolige [151] uses preconditioning methods over the sparse matrix A in order to transform the $O(N^2)$ cost into $O(N \cdot \log N)$. Using these methods maps including 60 thousand poses are optimized in a few minutes. On a different approach, Duckett [152] suggests using a relaxation based algorithm to solve the non linear optimization problem 3.8 instead of using an iterative linearization algorithm. This approach has the advantage of considering directly the non linear problem, being extremely simple to understand and it allows a total control of the computation cost. While $O(N^2)$ operations are theoretically required to find the optimal solution, using only $O(N)$ have shown to provide good enough experimental results. Frese [153] extends this approach towards a multi-resolution scheme, thus lowering the constant factor in the computation cost. With this multi-resolution method maps with one thousand poses can be updated in a few centiseconds. Finally the use of stochastic gradient descent methods [154] has been proposed to solve the SLAM problem [155, 156, 157]. This method provides the best state of the art result for large map optimizations.

Lazy data association Up to now, we have visited various different ways to solve the SLAM problem. All of them have strong mathematical relations and corresponds to different approaches over the same mathematical object. It has to be noticed however that all of these methods supposes that the data association between different landmarks has been realized successfully. It is known that data association is a non trivial task and it is critical for the creation of a consistent map. Most of the methods provides uncertainty measures over the actual landmarks in order to compute the probability of each data association choice. Commonly three options arise: choose the more likely data association at each step (this is what is almost always done), track multiple hypotheses (requires computing in parallel N hypothetical maps) or use retroactive data association. This last option, also named "lazy data association" [158], evaluates at each update the likelihood of the current map. If new measures make drop this likelihood under a predefined threshold then alternative maps (corrections over the data association) are evaluated, and the most likely map is kept. This lazy approach has the advantage of explicitly dealing with errors and reducing the computation cost by evaluating alternatives only when it is needed.

Algorithms based on covariance matrix and the ones based on information matrix correspond to duals. Methods of one class directly relates to the other one. In the next section we will see a different way to solve the SLAM problem that is more dissimilar.

Algorithms using Particles

As previously discussed SLAM can be seen as a Bayesian filtering process. Usually, for computational issues, the models are approximated as linear and with Gaussian noise. When such approximations are not good enough, the second most common option is to represent the probability density functions by a set of particles that samples the function. The objective is to make evolve the particles in such a way that a defined portion of them stay on high probable estimation of the true state. Using particles

filtering we can estimate any probability density function and make it evolve under any nonlinear dynamic model.

In the SLAM problem each new pose (feature or robot pose) adds a dimension to the joint distribution, this create an exponential explosion in the size of the exploration space. In order to provide good results it is required to keep the number of particles proportional to the exploration space. Applying particle filtering directly to the SLAM problem is clearly intractable.

Particle Filters SLAM A first proposal to use particles filtering from SLAM was presented by Thrun [159]. The idea is to use incremental maximum likelihood (choose the most probable alternative at each step) to construct the map of the explored area and use particle filtering to estimate the current position of the robot in the map (like when doing Monte Carlo localization, section 3.5.3). The particles are used to estimate the current robot pose (not the map) and to close loops in the map. The particles are used as initial guesses to perform data association between the actual measure and previous measures on the map. Once a new loop closing is detected an optimization method is applied to propagate the correction over all the map. Doing such a propagation has a quadratic cost but all the other operations have constant update time.

FastSLAM The method that popularized the use of particle filtering for SLAM was created by Montemerlo [135, 160] and it has the fancy "FastSLAM" appellation. In this method each particle represents a path of the vehicle. Due to the duality between mapping and localization; estimating the distribution of paths of the vehicle is equivalent to estimating the possible maps. Using this conditional independence property it is possible to estimate the most probable map with a cost of $O(p \cdot \log N)$ per measure. N corresponds to the number of landmarks in the map and p is the number of particles used in the algorithm.

The main drawback of this method is that there is no defined procedure to determine the sufficient number of particles p. On the other hand, this method manages explicitly the non linear models, does manage errors in data association and has been shown to be fast enough in practice.

Originally formulated for landmark-based maps, it has also been applied to maps of laser scan matches [161, 162] as illustrated in figure 3.24.

Highlighted solutions

As we have seen that many different solutions for the SLAM problem exist. The basic requirements of consistency and linear time, linear memory usage have been achieved by some algorithms, yet no one offers the perfect solution and different trade off have to be made.

Three methods seems particularly interesting:

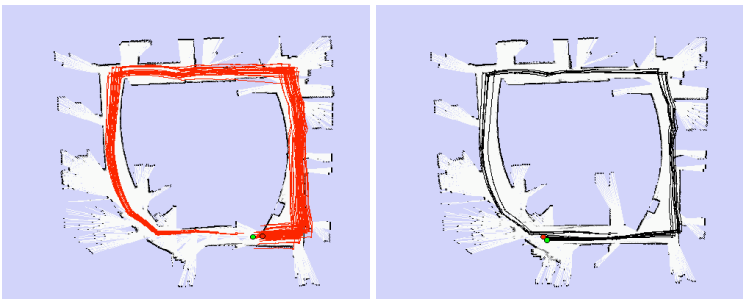


Figure 3.24: Loop closing in FastSLAM applied to laser scans. From [119]

- The relaxation based method [152] offers a very simple implementation, control over the computation cost, explicit use of non linear observation model and has been shown to be good enough for real world applications. It seems to be a good trade-off between simplicity, efficacy and efficiency.
- The stochastic gradient descent methods [155] are more complex to implement but they provide state of the art efficiency.
- Finally the lazy data association approach [158] is one of the few who actually put emphasis on the robustness of the result with respect to data association errors, thus enhacing the efficacy.

As the SLAM solutions acquire more maturity the researches focuses on more ambitious objectives. The logical extension is to consider a non static environment, as discussed in the next section.

3.7 SLAMMOT

To navigate, a robot needs to know what surrounds it: which is the geometry of the buildings, which mobiles exists and how they are moving. The Simultaneous Localization, MApping and Moving Objects Tracking (SLAMMOT) task pretends to provide this information.

Up to now we have supposed that the environment is static and thus data association and map building can be made without special considerations. Urban areas are, by definition, populated of moving objects. The presence of a moving objects will corrupt the measures introducing noise during the data association and adding spurious elements on the constructed maps.

Being able to separate moving objects from static objects during the SLAM process allows to enhance the results. On the other hand being able to do SLAM while tracking objects allows to estimate better their global speed and position and thus to better estimate their trajectory.

In this section we will first present the SLAMMOT from a mathematical formulation and discuss why it is a difficult problem. Next we will present some elements of the subtopic of moving objects tracking. Finally we will present the works realized on SLAMMOT. First publications on SLAMMOT are less than six years old, which means that this topic remains very young and on the cutting edge of the state of the art.

3.7.1 Problem formulation

The SLAMMOT problem integrates a SLAM and a moving objects tracking problem. At each new acquisition the measures have to be classified as observations of static objects or of dynamic objects. Static objects measures will be used to refine the robot localization estimate, and the dynamic objects measures will be used to refine the dynamic state estimate of the moving objects around the robot.

From a Bayesian perspective the SLAM inference problem is incremented in an additional dimension. In SLAM robot motion measures and environment observations are used to simultaneously estimate the actual state of the robot and of the static environment. If the static environment is described by a set of landmarks then estimating the state of the environment corresponds to defining the pose of the landmarks in a global reference system.

For presentation purpose we will assume that the world is composed by geometrically bounded "objects". In the extreme cases objects can be a corpuscular simplification (landmarks) or on the other hand a regular decomposition of the geometry of the objects (building blocks).

What differentiates SLAMMOT from SLAM is that the objects that compose the environment can be static objects or moving objects. Estimating the state of the environment corresponds to estimate the state of each compounding object. The state of the objects can be described by a discrete and a continuous component. The discrete component ζ_t^i classify the object i as static or moving at the discrete time instant t, the continuous component s_t^i defines the parameters related to the object position, its speed, etc...[2] The motion mode ζ_t^i can be "static", "moving at constant speed", "accelerating", "stopped", "random movement", etc... The set of motion modes will depend on the application.

The dynamic Bayesian network corresponding to the presented setup is illustrated in the figure 3.25 and the related Bayesian program is presented in the figure 3.26.

This SLAMMOT approach was called "SLAM with Generic Objects" when first presented by Wang [17], in this text however we will call it "SLAMMOT using Generic Objects" to avoid out-of-context confusions.

Clearly the SLAMMOT problem inherits all the difficulties of SLAM and increment them by expanding the dimensionality of the joint distribution involved in the prediction step of the Bayesian Filter solution formulation. This dimensionality expansion

[2] ζ is called "zeta"

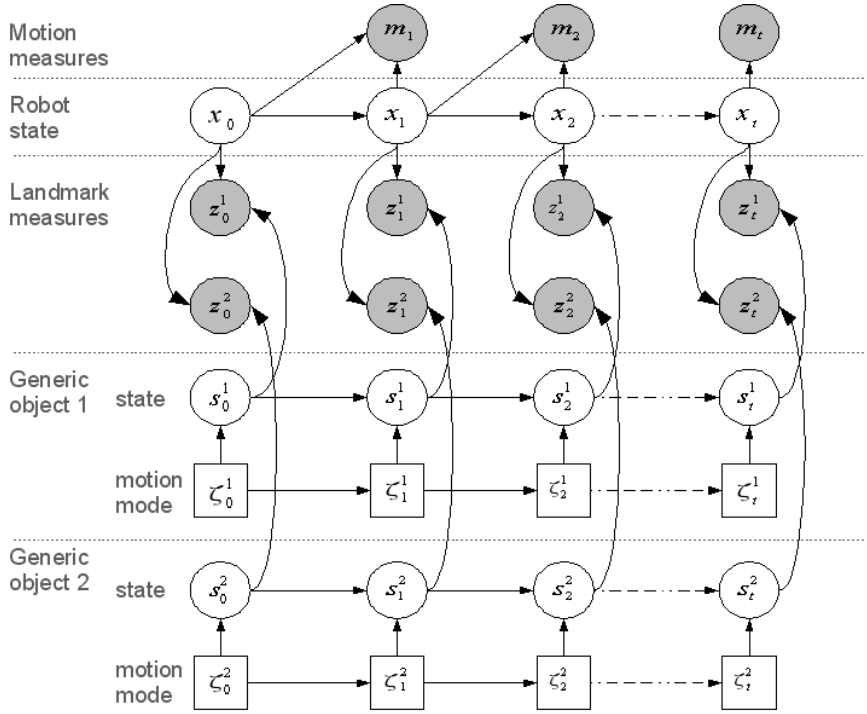


Figure 3.25: SLAMMOT using generic objects Bayesian network. Scenario composed by two objects. Circles correspond to continuous variables and squares to discrete variables

Prog
- Desc
  - Spec
    - Variables
      - $x_0,\ldots,x_t$ Robot state vectors in time
      - $m_1,\ldots,m_t$ Robot motion measures in time
      - $z_0,\ldots,z_t$ Measures of generic objects in time
      - $s_0,\ldots,s_t$ States of generic objects in time
      - $\zeta_0,\ldots,\zeta_t$ Motion modes of generic objects in time
    - Decomposition
      $P(x_0\ldots x_t\ m_1,\ldots,m_t\ z_0,\ldots,z_t\ s_0,\ldots,s_t\ \zeta_0,\ldots,\zeta_t) =$
      $P(x_0)P(z_0|x_0\ s_0)P(s_0|\zeta_0)$
      $\prod_{i=1}^{t}[P(x_i|x_{i-1})P(m_i|x_i\ x_{i-1})P(z_i|x_i\ s_i)P(s_i|s_{i-1}\ \zeta_i)P(\zeta_i|\zeta_{i-1})]$
    - Forms : any
  - Identification : any
- Question :
  $P(x_t\ s_t\ \zeta_t|m_1,\ldots,m_t\ z_0,\ldots,z_t)$

Figure 3.26: SLAMMOT using Generic Objects

is related to a computational explosion that put a direct, generic solution of Bayesian SLAMMOT problem out of the actual capabilities. As for the SLAM problem approximations and restricted base hypotheses can reduce the general SLAMMOT problem up to a tractable one.

The core idea of SLAMMOT is to obtain a mutual benefit from SLAM map construction and moving objects tracking.

In the following sections we will discuss the techniques developed for moving objects detection, moving objects tracking and finally tractable solutions for SLAMMOT implementations.

3.7.2 Moving objects detection and tracking

Moving objects tracking is a classic problematic in radar systems and video surveillance.

When using features as data representation the SLAM problem becomes a sort of moving objects tracking, because from the robot perspective, the static landmarks are moving.

The urban scenario has particular difficulties with respect to classic tracking applications. First we do not know a priori the background of a scene. Radar applications commonly do not have background (air medium), video surveillance systems suppose that they have a capture of the empty scene. Mobile robots in urban scenes do not know the background a priori, worst, from their perspective all the objects are moving and it is difficult to distinguish static buildings from persons or cars transitorily stopped. Our application require to make an explicit distinction. Another difficulty is that the objects do not behave in the same way, the dynamic of a car, a person, a dog, a motorcycle or a bicycle are notoriously different, and thus each case has to be identified and treated separately. Additionally pedestrians and vehicles tend to create groups of coordinated behaviour (specially at high density) and suddenly detach from these groups. This creates the effect of sudden apparitions of individuals on a scene.

In the following subsections we will review some of the most common techniques to do tracking, starting from simple moving objects detection, tracking of single targets and tracking of multiple targets.

Moving objects detection

Many tracking works supposes that the measures correspond uniquely to moving objects and then focuses on data association problems. However most of the real applications include spurious elements in the measures or presence of static objects. Radar application have ground noise (climatic perturbations, floor of the sea), video images have non stable backgrounds (trees on the wind, changing light conditions, moving camera), ladar measures includes non moving targets or spurious floor measures.

Detecting correctly the moving objects is a critical aspect of a moving objects tracking system. In particular for SLAMMOT in urban environment using a ladar separating moving objects from static objects is key point in order to reduce the computation down to a tractable scale.

In the following paragraphs we will discuss different techniques developed to detect moving objects in the particular case of SLAMMOT with ladar measures.

The simplest approach consist to detect moving objects using specific heuristics. Indoor pedestrians can be recognised by their geometry [163] or because they enter on previously free areas [164]. This method can be useful on restricted environments, but it is clearly not well suited for outdoor conditions (where a tree is similar to a pedestrian).

Using simple geometric observations it is possible to define a consistency based moving objects detection (see [165] and [17, section 6.2]). This method can be resumed in three simple rules:

- If an object is detected on a location previously observed as free space, the object is moving;
- If free space is detected on a location previously occupied by an object, that object was moving;
- If an object appears in a previously not observed location, then we can say nothing about that object;

Until evidence demonstrates the opposite, we will suppose that new objects are static. This approach is simple, clear and can be implemented in real-time.

Haehnel [166, 119, chapter 4] proposes a more robust approach using likelihood maximization. He defines a likelihood function that includes discrete terms that classify each measures as observing static or moving objects. Using expectation maximization it resolves the optimization problem for a group of scans. This method can accurately separate the static objects map and the moving objects map. However the computational cost involved limit its application to offline post-processing. Also this method does not cares about identifying specific objects or defining their trajectory.

Wolf [167] proposes a grid map alternative that allow on-line construction of moving objects grid map and static objects grid map. The static objects grid map is updated depending on the new observations and previous grid state. The probability that a cell contains a static object augments when:

- The location was not previously observed and the measure indicates the presence of an object
- The location was previously observed as occupied and the measure indicates the presence of an object

In any other case the probability of occupancy by a static object will decrease.

The moving objects grid map update its values from previous values of it, of the static objects maps and of new measures. When a new measure indicates that a cell is occupied, but the previous static objects map indicates that cell was free then the probability the cell is occupied by a moving objects is incremented. In any other case it is decremented.

This simple set of rules allow a real-time update of both static and moving objects grid map allowing to rapidly increment static map, detect moving objects and correct invalid statics maps. As any grid based approach this method suffers from spatial discretization and the memory usage associated to fine grained grids. With the online detection of moving objects enables, with a subsequent processing, the tracking of different objects.

The core idea to detect moving objects in ladar measures is to put in relation the static objects maps, the non observed areas map, the new measures and the knowledge about actual moving objects.

Moving object tracking

Once moving objects are detected and measures of their position are done it is desirable to track them. Tracking objects allows to aggregate measures in order to enhance the estimation of their state, and a better estimation of their current state allows a better prediction of their future position. The state vector can include position, speed, acceleration, its geometric description, a classification of its sort, etc... Tracking is essential to have good estimates of the moving objects and to realize predictions of their behaviour.

Tracking consists of two sub-problems: Bayesian Filtering and data association. In the following paragraphs we will discuss the models commonly employed and the methods used to solve the data association. Most of the presentation here is based on the work of Maskell [100], we refer the reader to it for a more formal and in depth discussion of the topic.

Moving objects models

During tracking the state vector of the moving objects will be estimated. The structure of such a state vector is defined by the model we attach to the entity.

It exists a trade off between the model complexity and the accuracy of the estimation. The more precise is our model, the best it will describe the behaviour of the moving object and then it will be possible to estimate its future trajectory more accurately. If the model is more complex, it will include more parameters to estimate online. For a fixed number of measures, the more parameters to estimate the less accurate the estimation will be. Because of this simpler dynamic models are preferred. The common models used are, stationary pose, pose perturbed by random noise (also known as Brownian motion), constant velocity perturbed by random acceleration, constant acceleration perturbed by random changes, constant acceleration with random changes for a bicycle model [168, 17].

In order to provide the better estimate at any time the moving objects can be modeled as having changing behaviours, for instance: stopped, running, accelerating, turning, etc... Multiple models are used over one same object and at each time step the best fit is selected. Using this multiple models approaches the best estimate is available at each time since the complex models are used only if they provide a better prediction of the objects movement than the simpler models.

State estimation

When a single model is used to approximate the dynamics of a mobile objects then the state can be estimated using filtering algorithms such as the Kalman filter, Extended KF, Unscented KF, Particles filter, etc...

When multiples models are used the state estimation becomes more delicate. The system model has a discrete part, that evolves in times indicating the actual behaviour (stopped, moving, accelerating, etc...), and a continuous part that indicates the current state vector of the dynamic system. Hybrid models are sometimes called switching modes models. An illustration of the dynamic Bayesian network can be found in the figure 3.25, as a Generic Object strip.

Unfortunately the state estimate for the hybrid model cannot be reduced to an iterative equation as in the Bayesian filtering case. Estimating the current state require evaluating exponential possibilities and marginalizing from all the past measures, which means that the estimation becomes intractable in just a few steps. Again, approximations and simplifications are the key to get a tractable option.

If the model supposes linear models and Gaussian noise then the exact result for $P(s_t|z_{0...t})$ is a mixture of gaussians, where the number of gaussians in the mixture grows exponentially in the time.

A common approximation is the so called Generalized Pseudo Bayesian method. This method tries to approximate the result by collapsing the mixture into only one gaussian, depending on the "degree" this collapse is done sooner or later.

The Generalized Pseudo Bayesian method of first degree (GPB1) keeps only one gaussian to estimate the actual state. After making an update for the k motion modes the new estimation is a mixture of k gaussians which are collapsed into only one, so the next iteration can start.

The GPB method of second degree (GPB2) keeps the current estimate a mixture of k gaussians. After each new estimate k^2 gaussians are available, which are again collapsed into only k estimates.

IMM The most commonly used method, named Interacting Multiple Model (IMM). Provides a trade off between GPB1 and GPB2. It only compute k gaussians as in GPB1 but it has as output a mixture of k gaussians as in GPB2. The reader can look at [168, 17] for the detailed equations of these methods.

Another common approximation method consist to choose the most probable alternatives instead of collapsing all of them into one Gaussian. Then Multiple Hypothesis Tracking can be used to track a set of possible modes transitions. As parallel alternatives are keep the implementation cost is somewhat higher than IMM. Particle Filter can also be used in a similar fashion, since each particle is a probable alternative and we keep viable options alive.

In the context of single target tracking, data association focuses on distinguishing a real measure from a spurious one. The methods employed essentially correspond to the one mentioned in the section 3.5.2; nearest neighbour and "probabilistic data association filter" (collapsing in one Gaussian the set of candidates, similar to GPB1).

Of course the scenario where one and only one target exists in the environment is purely theoretical. In practice multiple targets will be found and tracked simultaneously.

Multiple objects tracking

Multiple objects tracking is essentially the same problem that with one target, however the complexity to manage data association is considerably incremented. With multiple objects there is more occlusion, it can be difficult to distinguish one target from the other one, and the complexity to solve data association grows exponentially with the number of targets in the scene. Of course, this last parameter is also unknown and has to be deduced from the observations.

Multiple targets data association The data association for multiple targets consist to define if each measure corresponds to an already known object being tracked, to a spurious measure or to a new objects in the scene that will be started to be tracked.

A first approach consists on using one Kalman filter per target and solve data association independently for each track, using the methods previously mentioned (nearest neighbour, gating, probabilistic data association). This method is simple and direct, but it does not provide guaranties that a single measure will be associated over different targets.

To avoid this "one measure to many targets" effect the different data association possibilities have to be evaluated. The mapping of possible associations between the set of measures and the set of possible target grows exponentially with the number of targets (and measures). Thus exhaustive evaluation of the probability of each mapping has a computational cost that grows exponentially.

GNN Searching for the optimal data association (mapping between measures and targets) can be done in polynomial time instead of exponential time if a search algorithm such as Hungarian, auction, JVC, or Munkres are used [114, 169]. Independent of the employed algorithm this approach is called Global Nearest Neighbour, since as for the single target case it search the optimal data association.

JPDAF When more than one option is plausible the Global Nearest Neighbour will only choice the best one, without reflecting the ambiguity in the choice. The multiple targets equivalent to the probabilistic data association is the the joint probabilistic data association which assign the observations and their uncertainty considering not only their match over the predicted position but also joint relations as coherence (one target one measure) and exclusion (one measure one target). An enhanced version of Joint Probabilistic Data Association Filter (JPDAF), named Joint Likelihood Filter, has been proposed [170] to cope with its independence and homogeneity hypotheses.

MHT A third possible approach is to implement Multi Hypothesis Tracking. Instead of keeping the best option or reflecting in the uncertainties the ambiguity in the choices, we can keep a set of filters for the N most probable alternatives. This method is quite simple and provides good results.

Particles Filter All the method that are applied to Gaussian distributions have been generalized for particles representation, the reader can consult [171, 172] to see some examples of such variants. Particles filters, are more adequate to represent the multi-modal uncertainty of this problem, and to cope with the nonlinear issues. However the use of large number of particles can sometime generate a high computational cost.

All the previous methods supposes that the number of existing targets is known. However knowing this value is not trivial. The most pragmatic approach uses the gating around the uncertainty in the target poses to decide if a measure corresponds to a new object or not and then expand the filter accordingly. More sophisticated approaches try to include the uncertainty of the number of targets directly in the filtering process (see [100, section 2.7]).

The multi-target tracking problem has been a topic of research for many years, specially for military application. As such many sophisticated, complex and expensive methods have been proposed. However in the practice simple ones have proved to be quite effective [17, 173].

Since we have discussed about SLAM (section 3.6) and Multiple Objects Tracking (section 3.7.2), now we are able to introduce the works done to solve the SLAMMOT problem.

3.7.3 Simultaneous localization, mapping and moving objects tracking

As previously mentioned a SLAMMOT system is the minimal requirement for autonomous mobile robots in urban areas. In order to reach its objective the mobile robot needs to know in real time the presence of static obstacle, of mobile obstacles, and how such mobile objects will behave in the future.

Up to date very little work has been done on SLAMMOT. Much of the last years efforts have been focused on solving the underlying core issues related to sensor fusion,

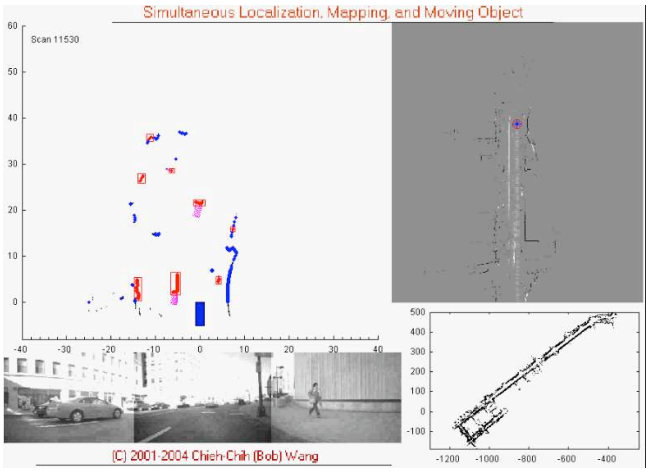


Figure 3.27: state of the art SLAMMOT system. From [17]

robot localization, SLAM in static indoor areas, SLAM in outdoor areas, and tracking moving objects. As SLAM techniques are maturating the research are slowly shifting to SLAM on dynamic environments problematics, and we can expect in the next year the growing apparition of works on SLAMMOT design, development and applications.

Probably one the first teams that identified the need for SLAMMOT was the Prassler group [174]. In their work about automated wheel chairs they developed the first system for static and dynamic objects detection, tracking and avoidance. They explicitly specified the need for such a system and developed the first prototypes.

Some years latter Haehnel [119] presented a method for pedestrian tracking and extraction during 2D and 3D SLAM models reconstruction [175], and a Maximum Likelihood approach for offline static objects map construction with detection moving objects [166]. Event if not able to provide on-line information, or predicting moving objects behaviour, Haehnel proposed a theoretically sound method for separating moving and static objects.

Wang [17] was the first researcher to develop outdoor real-time city sized SLAMMOT. Indeed he coined the term. First called “SLAM with Detection And Tracking of Moving Objects” [176], Wang studied the core issues related to SLAMMOT and developed a functional prototype for urban environments [17] (see figure 3.27).

One of the most recent works related to SLAMMOT was presented by a Montesano [164] and is dedicated to wheel chairs automation. They implement a real-time SLAMMOT system for indoor areas that exploits the moving objects information when matching the laser scans. Objects detection is made using a simple heuristic and the tracking via constant speed Kalman filters. In despite of the simplified environment, they present the first work that relates a real time SLAMMOT system with a planner module for a mobile robot.

Obviously SLAMMOT systems are just in their infancy. There is still many room for improvements and exploration. As computational power increases less simplified methods will be used, enhancing robustness and precision. Long term execution issues have not yet been well studied, how to continuously enhance/rectify the constructed map, and how to do the best use of the collected data. There will always be room for incremental enhancements through the inclusion of the latest advances on the underlying problematics of SLAM, MOT and sensors fusion.

3.8 Scene understanding

In the context of perception for mobile vehicles, scene understanding relates to the capacity of the perception to explain what causes the movement of the dynamic objects in relation to the static objects.

The aim of scene understanding is to assign a "purpose" or "role" to each moving object in order to predict as well as possible their future trajectory. Some instance of urban scenes understanding are "the group of children will cross the street to enter at the school", "there is a car that wants to enter into the parking place", "the person A wants to cross the street because his mate B already crossed it", "the static person A wants to cross the street because he looks at the cars arriving".

Humans are naturally capable of interpreting with high success the intentions of other persons. This seems to be an important factor of decision while driving a car, and thus automated system should try to extract this information too. For a machine, interpreting the intentions of human beings is a very difficult task.

Being at the higher level of complexity for a perception system this is one the least explored aspects of the perception systems for mobile vehicles.

A first approach can be done specifying a finite set of possible actions and relations for the objects in the world and posing the scene understanding problem as a classification problem [177]. Of course this approach is error prone when non predefined cases appears, when the set of possible actions is too big, or simply because even in finite sets, the classification problem of possible behaviour is a difficult one.

A more tractable approach consist to define a set of probable paths for a target and estimate the actual probability of being following one of thus or not. This approach does not consider specific interrelations between objects (young persons with educational institutions, for instance) and thus reduces the problem complexity. Under this perspective scene understanding consists on assigning a set of probable path to an object. In the ideal case one path will become highly probable and thus its future trajectory will be predicted with high certainty.

Since urban moving objects tends to repeat similar paths (specially cars), the set of probable future paths can be estimated from previous observations. Multiple works [178, 179, 180, 181] show that, when enough statistics are available, this idea can give surprisingly good results. The mix of this kind of approaches with a SLAMMOT system still to be done.

When robots are interacting in the same scene, the use of communications can help the scene understanding, since each robot may be able to exchange its current goal and plan. This idea is discussed in chapter 6. Human driven cars using navigation systems could also exchange their current route.

Future advances in scene understanding probably will be generated by a better understanding of the human perception process and/or by the exponential grow of the computational power available for real world applications.

3.9 Conclusion

In this chapter we have reviewed the perception process from the direct sensorial input up to the high level understanding of an urban scene. In the middle we have presented a considerable number of methods and algorithms developed to resolve the different problematics encountered. From this wide view some core ideas have to be recapitulated.

It is a fact that every observation of the world will be uncertain. The existence of this uncertainty is not a problem by itself, the real problem is *quantifying the uncertainty* from the measures and through all the state of processing, up to the moment of taking a decision.

From the review it can be noticed that the perception problems have a hierarchical structure. Each problem includes the preceding one explicitly. Scene understanding includes SLAMMOT which includes SLAM which includes Bayesian Filtering which includes Sensor Fusion which includes sensing.

Dealing with uncertainty is a difficult problem. Most of the non trivial tasks are confronted to explosive computational cost. This particularity constrains many applications and is one of the principal research motivations. In order to overcome the computational burden it is necessary to make simplifications. Then, the art consists on finding to the adequate approximations that will lead to a tractable and effective solution for the particular problem attacked.

Nowadays dealing with the uncertainty is recognized as one of the core elements in robotics. Because of this, the usage of statistics and probabilities tools is omnipresent on the domain. This fact is seen as part of a major trend in robotics, the so called Probabilistic Robotics [182].

In the next chapter we propose a new system to meet the specific needs of a perception module for robotic driving in urban roads and some experimental results.

Chapter 4

Perception for urban driverless vehicles

> Clearly, then, the city is not a concrete jungle, it is a human zoo.
> Desmond Morris

In this chapter we will present the design of a perception system suitable for the safe navigation of robots in urban roads, discuss the underlying problems, propose suitable solutions to these problems and present some experimental work. In chapter 5 we will use this perception system to compose a driverless vehicle.

The urban environment is a challenging environment for robotic applications. The robot is not physically bounded to a specific area, and the expected length of itineraries can be count in tens of kilometers. Even if the cities have some elementary structure (buildings, space for humans, space for transports) they present a large diversity in shape, colors, textures and local architecture. The general aspect of a street vary largely between different cities and can vary largely even inside one same city. An important notion to keep in mind is that cities are a living space that is *constantly changing*, new buildings appear, different objects are installed on the road sides, underground works are done, streets are modified, etcetera. At last but not least, the city is inhabited by persons and animals generating a large set of moving objects around the robot.

4.1 Problem definition

As discussed in the chapter 3, the design of a perception module for a specific application needs to answer three questions: what to measure, which model to build, and how to transform the measures into the desired model?

In the last years most work on mobile ground robotics has been focused on the SLAM problem. As discussed in section 2.6 the availability of maps built with SLAM are of

no use for safe navigation in urban areas. Since the robot evolves in unknown environments with moving objects we need to design an application specific SLAMMOT system.

Let us review again the needs that the three decision modules mentioned in section 3.2 impose over the perception module, in the context for urban roads.

4.1.1 Perception for route planning

To decide the best route it is necessary to have a graph of roads where the vehicle can pass, a prediction of the traffic on the road network traffic, the position of the destination and the current position of the vehicle on the roads graph. Of these, the only information that the vehicle can provide by itself is its localization in the known graph of roads.

Since the roads in the city are usually largely inter spaced a history of position measures of ± 10 $[meters]$ precision over a few hundred meters should be enough to localize the vehicle on a specific road.

Using GPS, a cheap inertial sensors and filtering (see section 3.5), current commercial navigation products are able to provide a successful localization in most conditions. Coupled with odometry or with exoperceptive motion estimation methods current technology is be able to provide a good enough localization for routing. So, essentially, perception for route planning is a *solved problem*.

If dependency on satellites is not desired, existing solutions for visual place recognition [183] or WiFi based localization [184] are able to provide precise enough measurements to use as replacement for the GPS.

Creating the roads graph is usually done using aerial image processing and ground measurements. How to apply robotic methods to lower the cost of building and updating such a map is out of the scope of this dissertation. WiFi or image information enrichment of current maps can be done using a GPS based system.

There is a second issue related to perception and route planning. Once the route has been defined, it provides short term goals for the trajectory planner. However route position has low precision, while goal precision needs to be high precision. How to pass from one to another will be discussed in chapter 5.

4.1.2 Perception for trajectory planning

Just as many work focuses on SLAM to enable mobility, many work seem to focus on detecting obstacles as the key element to allow trajectory planning. Such approaches fail to consider one important element, the time dimension.

In chapter 2 we defined the requirements on the perception system to allow the computation of safe trajectories. In the particular, we need a world model w capable of

providing a conservative approximation of the harm function $h(x(t), w(t))$. Since trajectory planning consists on defining a sequence of *future states* for the vehicle, the world model needs to provide predictions. Even more, as discussed in chapter 2, the world model needs to provide *conservative predictions.*

We need a SLAMMOT system capable of:

- Defining the current traversability of the surrounding space
- Detecting and tracking moving objects (in order to predict their movement)
- Estimating the state of the vehicle with respect to the ground
- Evaluate the harm function h

The current traversability will be given by the mapping of static obstacles and the analysis of the ground surface (supposed immutable). The future traversability will be based on the moving objects state estimation and the use of a priori behaviour models.

In order to provide a conservative approximation, the uncertainty of the current world model estimate needs to be correctly estimated.

The evaluation of the harm function will be based on the classification of different objects (e.g. low value for holes and vegetation, medium value for walls and trees, high value for other cars and animals, very high value for humans).

Sensing range and precision

How much ? In chapter 2 we discussed how the visibility range of the sensors used by the robot limits its behaviour. A perception system should ideally be able to do measurements all around the vehicle (360 degrees of coverage).

How far ? Based on the discussions of chapter 2 and visibility range between ~30 to ~100 meters would allow maximum speeds of ~50 [km/h]. Seeing less will lower the maximum reachable safe speed.

Which precision ? The precision of the sensors will affect the uncertainty of the traversable space estimation. When using a conservative approximation larger uncertainty imply smaller traversable space.

Considering a straight road of 4 meters wide, a vehicle 2 meters wide, a stopping distance $d_{stop} = 30\ [m]$ then the vehicle needs to be able to distinguish at d_{stop} meters of distance obstacles with a precision of at least ± 1[m] in order to not being slowed down by insufficient precision on its sensors. At shorter distances a higher precision is desired in order to be able to perform tight manoeuvres.

How often ? Given a conservative approximation of the world and a safe planning method, arbitrarily low sensors updates are acceptable. Even if the sensors are abruptly stopped the vehicle will still have a harmless behaviour. A refresh rate around $\sim 10\ [Hertz]$ seems to be enough for an agile behaviour of the vehicle.

Since the perception system is expected to run online in an embedded platform (the robot), the implementation needs to respect computing constraints on CPU and memory usage while ensuring a real time execution.

4.1.3 Perception for control

The planning module will have defined a sequence of states to track. In order track them the control module needs to estimate the current state of the vehicle. Since the trajectory is defined with respect to the local environment, the state estimation consist on measuring the speed, orientation and position of the vehicle on local reference frame. As previously discussed in section 3.6, creating a local map and estimating the local displacement are the same problem. Thus the needs for control are already covered by the needs for trajectory planning.

The planning needs to consider the vehicle geometry and the possible errors on the tracking. Due of its prevenient nature, the planning needs to assume a guaranteed bound on the maximum tracking error of the controller. This means indirectly that over the horizon of a planned trajectory the localization error needs to have a known bound.

4.2 Proposed solution

4.2.1 Assumptions

The approach we use needs some assumptions to be considered valid:

- Sensor fusion of internal sensors of the vehicles is good enough to allow little errors on the scan matching,
- A computational power of 1 gigaflops, 1 gigabytes (as orders of magnitude) is available on board,
- Using a locally planar world model for navigation is a “good enough” approximation,
- Measures of moving objects observed in urban scenarios are easily separable between them,
- At the city level GPS is uniformly available on 50% of a path.

4.2.2 What to measure ?

As discussed in §4.1 the main quality that the robot needs to quantify is the traversability of the surroundings.

In outdoor mobile robotics, the sensors commonly employed to observe the surrounds are video cameras, radars and laser scans (see section 3.3 and [34]). We choose the last one as the main sensor due to its larger range (more than 180° and 40 meters) and high precision ($\pm 1°$ and ± 0.1 meters). Notice that the laser scanner measures provide information about the presence of obstacles and the existence of free space. Since we are supposing that urban environments are locally planar and a 2D representation of the environment is good enough for navigation purposes.

Due to its 2D nature the laser scanner is not able to provide all the required information. Any obstacle lower than the height of the laser scanner risk to be missed, the same goes for holes. Even a full 3D map of the world may be not enough to classify traversable areas, e.g. grass areas may not be distinguishable in a 3D map. Due to this intrinsic limitations we propose to complement the laser scanner measurements with a visible spectrum vision system.

In addition to this two sensors, we will use the usual battery of commercial grade odometry sensors, IMU, GPS receiver and magnetic compass.

4.2.3 Which model to build ?

We build the model associated with the SLAMMOT problem (see figure 3.26). The laser scanner measurements will be used to estimate the displacement, the free space and the presence of static and moving obstacles. The traversability of the static part of the world will be also fed by the output of the vision algorithm.

4.2.4 How to transform measurements into the model ?

Section 4.3 describes a novel SLAMMOT formulation that allows a model estimation with few assumptions about the world, and with low computation footprint (CPU and memory).

Section 4.5 describes a complementary process focused on estimating the traversability of static elements of the world based on vision.

4.3 Efficient SLAMMOT

For a description of the SLAMMOT problem and existing methods, please consult section 3.7. The following text will describe a new solution tailored to fit the particular needs described in §4.1.

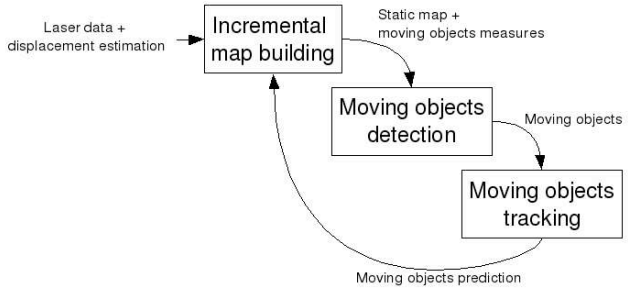


Figure 4.1: Laser based SLAMMOT algorithm succinct diagram. The outputs are the static map and the tracked moving objects.

Key idea

The key idea of our approach consist on realizing that, since the vehicle only cares about the information in its surrounding, using an Incremental Maximum Likelihood approach and correctly managing the errors in the map is a good enough solution. We use an adequate data representation that allows to build such a map rapidly and implement over it a moving objects detection method. The results of the moving objects detection are used to do moving objects tracking. The predictions of this last module are fed back into the map building method in order to gain robustness and have a single coherent SLAMMOT solution. See figure 4.1.

The key ingredients of our proposal are the data representation of static obstacles, the laser scan matching method that allow the displacement estimation and the method for detecting and tracking moving objects.

4.3.1 Data representation

We propose to use the grid of gaussians representation (see section 3.5.1) and adapt it to allow moving objects detection. This dense representation has three key advantages:

- It correctly manages the uncertainty related to the sensor, the 2D structure, the displacement error;
- At similar discretization levels, it provides a more accurate description than a traditional occupancy grid;
- It allows a fast scan matching and incremental map updates.

As previously mentioned the grid of gaussians representation split the space in overlapping cells where the occupancy evidence is stored. Then inside each cell a Gaussian distribution is estimated, as illustrated in figure 4.2.

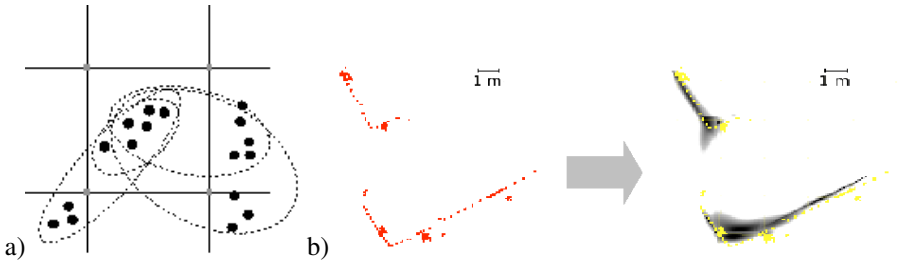


Figure 4.2: a) Diagram of gaussians in a four intersections grid. Dots are measure points and dashed ellipses are 90% contour of the Gaussian distributions b) Example of raw data and the resulting grid of gaussians representation. Darker pixels indicate higher occupancy probability

Having a Gaussian distribution per cell instead of a uniform distribution as in the usual occupancy grid makes all the difference. For the same cell size the occupancy distribution is better approximated. Since the distribution is estimated from the accumulated data this distribution embeds the occupancy, the sensor noise and the localization errors.

Since the Gaussian distribution parameters can be updated incrementally with each measure, given a set of l measures updating the occupancy grid has a cost $O(l)$.

4.3.2 Incremental computation of the grid of gaussians

Given a localised laser scan measurement, its impact points are positioned over the grid of overlapping cells. Each time an impact point is added to a cell its associated Gaussian distribution is updated.

As mentioned in section 3.3.2 a 2D Gaussian distribution is defined by its mean vector mean vector $\vec{\mu}$ of dimension 2 and a symmetric covariance matrix Σ with three informative elements. We search then to be able to incrementally compute the mean elements μ_x, μ_y and the covariance matrix elements $\sigma^2_{xx}, \sigma^2_{xy}, \sigma^2_{yy}$. To do this we use a simple recursive formulation that employs the helper variables n and $n\sigma^2_{xx}, n\sigma^2_{xy}, n\sigma^2_{yy}$.

Given an impact point at coordinates (p_x, p_y) we compute:

$$
\begin{aligned}
\delta_x &= (p_x - \mu_x)/(n+1) \\
\delta_y &= (p_y - \mu_y)/(n+1) \\
\mu_x &= \mu_x + \delta_x \\
\mu_y &= \mu_y + \delta_y \\
n\sigma^2_{xx} &= n\sigma^2_{xx} + n \cdot \delta_x \cdot \delta_x + (p_x - \mu_x) \cdot (p_x - \mu_x) \\
n\sigma^2_{xy} &= n\sigma^2_{xy} + n \cdot \delta_x \cdot \delta_y + (p_x - \mu_x) \cdot (p_y - \mu_y) \\
n\sigma^2_{yy} &= n\sigma^2_{yy} + n \cdot \delta_y \cdot \delta_y + (p_y - \mu_y) \cdot (p_y - \mu_y) \\
n &= n + 1
\end{aligned}
$$

All values are initialized to zero. Then at any time we can retrieve $\vec{\mu}$ and Σ by simply computing:

$$\begin{aligned} \mu_x &= \mu_x \\ \mu_y &= \mu_y \\ \sigma^2_{xx} &= n\sigma^2_{xx}/n \\ \sigma^2_{xy} &= n\sigma^2_{xy}/n \\ \sigma^2_{yy} &= n\sigma^2_{yy}/n \end{aligned}$$

If $n < 3$ the estimated values are considered non valid.

The first laser scan provides an initial set of valid Gaussian distributions, constituing an initial estimate of the grid of gaussian. This map will be used to estimate the displacement of the second laser scan before using the laser impacts to incrementally update the gaussians in the grid.

4.3.3 Displacement estimation

The traditional method to estimate the displacement consist on matching successive scans between them (using a method like ICP, see section 3.5.2). The use of the grid of gaussians representation allows to use a better alternative. Instead of matching two clouds of points, the laser scan is matched over a distribution of probabilities indicating the probable presence of an object at each point of the space. This approach avoids the expensive closest point search, provides more robust results and has a quadratic convergence rates instead of linear rates of the traditional ICP [185].

As it names indicate the Grid of Gaussians is composed by a set of overlapping Gaussian distributions. Each bi-dimensional Gaussian is defined by its mean vector q and its covariance matrix Σ. A laser scan measure is defined as a set of l points x_i. Let g be the number of Gaussian distributions in the grid. The score function between a scan and the occupancy distribution can be written as equation 4.1.

$$score = s(p) = \sum_{i=1}^{l} \sum_{j=1}^{g} \exp\left(-\frac{1}{2} \cdot (x'_i - q_i^j)^T \Sigma_i^{j-1} (x'_i - q_i^j) \right) \tag{4.1}$$

q_i^j, Σ_i^j are the mean and covariance of the gaussian j. Due to the locality of the estimated gaussians, the ones with little or no influence can be detected using their position in the grid and omitted to accelerate the score evaluation.

The term x'_i describes the scan point x_i in the map reference frame, translated using $p = (p_x, p_y, p_\phi)$ the current scan pose estimate and the rigid transform function $T : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ defined in equation 4.2

$$x'_i = T(p, x_i) = \begin{pmatrix} \cos p_\phi & -\sin p_\phi \\ \sin p_\phi & \cos p_\phi \end{pmatrix} \cdot x_i + \begin{pmatrix} p_x \\ p_y \end{pmatrix} \tag{4.2}$$

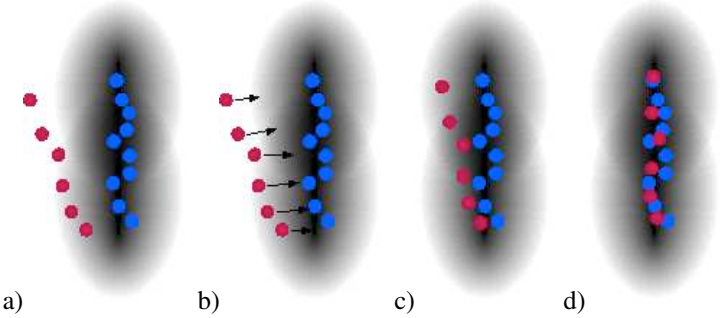


Figure 4.3: Illustration of the scan-to-map matching process. a) Map from previous measures is available (blue dots). New measures (red dots) are put at the initial position estimate ($T(p_0, x_i)$). b) For each point the gradient (and Hessian) is directly computable. c) Scan position p is updated, iterate back to (b) until convergence. d) Matching result after convergence

The objective of scan matching is to search the position $\hat{p}$ of the scan that optimizes the score of (4.5), $\hat{p} = argmax_p s(p)$. The derivatives of the score function $s(p)$ can be written explicitly and are cheap to evaluate thus optimization methods such as gradient descent and Newton's can be applied directly.

Let p_n be the current estimate of $\hat{p}$, then the Newton's method defines p_{n+1} as

$$p_{n+1} = p_n - \gamma \cdot (H\ s(p_n))^{-1} \cdot \nabla s(p_n) \tag{4.3}$$

where γ is the descent step size ($\gamma \approx 1$), $H\ s(p_n)$ is the Hessian matrix containing the second partial derivatives of $s(p)$ evaluated at p_n and $\nabla s(p_n)$ the gradient vector containing the first partial derivatives of $s(p)$ evaluated at p_n. Both $H\ s(p_n)$ and $\nabla s(p_n)$ can be defined explicitly. The figure 4.3 illustrates this iterative procedure.

Computation cost One iteration of the equation 4.3 requires one evaluation of $(\cos p_\phi, \sin p_\phi)$ and $l \times g$ evaluations of $\exp(\cdot)$ plus some multiplications and additions. On current computers, equation 4.3 can be computed at ~1 [KHz] when $l \times g = 1000$, considering that convergence is usually obtained under 20 iterations then the scan matching speed is ~50 [Hz].

We initialize p_n with p_{n-1} composed with the odometry displacement estimate, after matching, we have a new measure of the displacement. This measure in introduced into an UKF filter which uses the vehicle model and the commands input. Finally, we update the map with the new scan using the resulting filtered position.

Displacement estimate uncertainty In order to use the matching position $\hat{p}$ in a sequential Bayesian filter, its uncertainty needs to be estimated. It can be shown [186]

that equation 4.4 provides a conservative estimate of the position $\hat{p}$ uncertainty.

$$covariance(\hat{p}) \approx 2 \cdot \frac{s(\hat{p})}{l \cdot g - 3} \cdot (H\ s(\hat{p}))^{-1} \tag{4.4}$$

Having a direct access to the scan matching uncertainty is one of the comparative advantages of using the Grid of Gaussians representation with respect to plain ICP.

Using moving objects prediction When estimating the displacement we want to use only the laser scan measures of static obstacles. Let $w_i = P(x_i\ is\ static \mid previous\ measures)$ be probability of point x_i being the measure of a static obstacle, then the equation 4.1 can be rewrote as equation 4.5. When considering the a priori on static or moving objects measures the displacement estimation and static obstacles map building becomes robust to the presence of moving objects.

$$score = s(p) = \sum_{i=1}^{l} \sum_{j=1}^{g} w_i \cdot \exp\left(-\frac{1}{2} \cdot (x_i' - q_i^j)^T \Sigma_i^{j-1} (x_i' - q_i^j) \right) \tag{4.5}$$

In the next subsection we will show how to merge the grid of gaussians representation with a moving objects detection method. The detection of moving objects will allow their tracking, prediction, and thus to estimate the a priori $P(x_i\ is\ static \mid previous\ measures)$.

4.3.4 Moving objects detection

In section 3.7.2 we discussed existing methods for moving objects detection from a mobile platform. Here we will discuss how to implement such a function when using a grid of gaussians representation.

The core notion to detect moving objects is the inconsistencies between observed free space and observed occupied space [167]. If free space appears where a static object was observed, then it probably moved. If measures appear in areas previously seen as free, then these measures probably correspond to moving objects.

Let be $P(S_t^x)$ the static obstacles' occupancy probability at the point x and the instant t. Instead of updating the occupancy probability $P(S_t^x)$ using only the last observation value o_t, the update depends both of the observation value o_t and of the last occupancy estimate $P(S_{t-1}^x)$.

The probability of occupancy is divided in three ranges Free, Unknown and Occupied. Then the relation $P(S_t^x \mid S_{t-1}^x,\ o_t)$ enforcing the coherence between free and occupied space observations can be illustrated as shown in table 4.1. The case when the last observation gives no information about the occupancy probability, $P(S^x \mid o_t) = Unknown$, is omitted. The distribution $P(S^x \mid o_t)$ is constructed based on the sensor characteristics, while $P(S_t^x \mid S_{t-1}^x,\ o_t)$ is a design variable.

Table 4.1: Inverse observation model for the static occupancy probability

$P(S^x_{t-1})$	$P(S^x\mid o_t)$	$P(S^x_t\mid S^x_{t-1},\, o_t)$
Free	Free	Low
Unknown	Free	Low
Occupied	Free	Low
Free	Occupied	Low
Unknown	Occupied	High
Occupied	Occupied	High

The occupancy probability update is then written as in equation 4.6.

$$\begin{aligned} &odds(x) = P(x)/(1-P(x)),\\ &odds(S^x_t\mid o_{1\ldots t},\, S^x_{1\ldots t-1}) = \\ &\quad odds(S^x_t\mid o_t,\, S^x_{1\ldots t-1})\cdot odds(S^x)^{-1}\cdot odds(S^x_{t-1}). \end{aligned} \tag{4.6}$$

In order to merge this approach with the grid of gaussians representation we propose to separate the storage of occupancy measures O_{occ} and the free space measures O_{free}, as defined in 4.7.

$$\begin{aligned} O_{occ} &= \{o\mid P(S^x|o) = Occupied \quad and\ o \in o_{1\ldots t}\}\\ O_{free} &= \{o\mid P(S^x|o) = Free \quad\ \ and\ o \in o_{1\ldots t}\} \end{aligned} \tag{4.7}$$

Since $odds(S^x_t|o_{1\ldots t},\, S^x_{1\ldots t-1})$ is estimated from a multiplication series (equation 4.6), this series can be split and reduced in two separate factors, defined at (4.8). The factor $odds^x_{occ}$ accounts the occupancy estimation based in occupied space measures and the second factor $odds^x_{free}$ accounts the occupancy estimation based in free space measures.

$$\begin{aligned} odds^x_{occ} &= odds(S^x_t\mid O_{occ},\, S^x_{1\ldots t-1})\\ odds^x_{free} &= odds(S^x_t\mid O_{free},\, S^x_{1\ldots t-1}) \end{aligned} \tag{4.8}$$

Occupancy probability can be retrieved at any moment multiplying the two values, as shown by the equation 4.9.

$$odds(S^x_t\mid o_{1\ldots t},\, S^x_{1\ldots t-1}) = odds^x_{free}\cdot odds^x_{occ} \tag{4.9}$$

When doing this separation the grid of gaussians can be used directly as part of the detection of static obstacles and moving obstacles.

If points are added to a gaussian only when $P(S^x_{t-1}) = Occupied$ then the Gaussian distribution evaluated at x can be used as an approximation for $odds^x_{occ}$. This means that the grid of gaussians provides an estimate of the static obstacles in the environment. Because of the dynamic nature of the environment, a previously static object

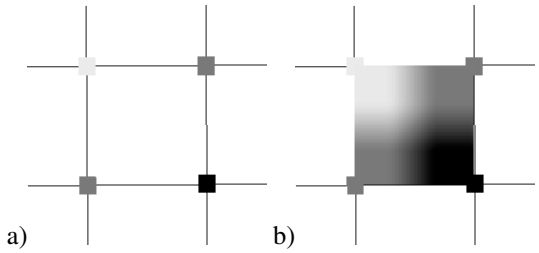
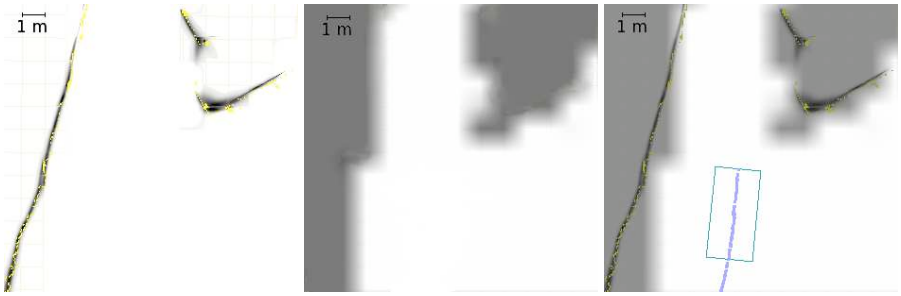


Figure 4.4: Illustration of the odd^x_{free} representation used. a) The values of odd^x_{free} are computed only at the points on the intersection of the grid of gaussians cells (see figure 4.2). b) The interior values are interpolated simply using a bi-linear interpolation



(a) Occupied space evidence (b) Free space evidence (c) Occupancy probability

Figure 4.5: Static occupancy probability scalar field approximation using a grid of gaussians for occupied space (a) and bi-linear interpolation for free space (b). Cells size is 1 [m]. A vehicle and its past trajectory are also shown in the resulting map (c). In (c) white indicates free space, black occupied space and grey areas have non zero occupancy probability. Figure (c) an output of the software described in §4.4

can start moving. In order to clean the gaussians that correspond to space that is no more occupied it is necessary to keep an estimate of the occupancy probability at its mean value q_i (we suppose that the shift of mean point during gaussians parameters updates does not invalidate the occupancy probability estimate). When $P(S_t^{q_i}) = Free$ the corresponding gaussian is erased.

Representing odd^x_{free} The factor odd^x_{free} is a function over space that can be estimated using any representation (including coarse or fine grids). In our implementation we use a bi-linear interpolation between the corners of a cell of the grid of gaussians (the use of the same grid is only for memory usage efficiency, any other grid is possible too). At each corner of this grid an estimate of odd^x_{free} is kept based on the accumulated measures. The values of odd^x_{free} at other locations is obtained by using a bi-linear interpolation between the nearest point, as illustrated in figure 4.4.

An illustration of the resulting occupancy probability scalar field can be seen at figure 4.5. The proposed method still is an approximation (just as grid methods), however separating occupancy and free area factors allows to better control the approximation error. More precise approaches would consider updating the gaussian parameters when portions of it pass into free regions.

Retrieving the moving objects measures At the end of the scan matching, each point x'_i has already been evaluated over its corresponding gaussians, thus $odd_{occ}^{x'_i}$ is available. Computing the $odds_{free}^{x'_i}$ allows to estimate $P(S_t^{x'_i})$. Points where $P(S_t^{x'_i}) = Free$ are considered as moving objects measures.

4.3.5 Moving objects tracking

Once we are able to detect moving objects we need to track them in order to estimate their state and predict their behaviour (since the prediction will be used for the planning stage). Tracking multiple moving objects is a classical problem (see section 3.7.2). In the general case this problem is very hard, however it has been shown experimentally that simple methods are good enough to cope with urban scenarios [173, 17].

Based on these results we use a simplistic approach where moving objects are defined as clusters of nearby points. The clusters are defined by a fixed distance threshold between consecutive laser points (0.3 $[m]$ for instance). Each cluster is then tracked separately and classified according to their size, speed and geometry. The object class is used to adapt the model used for tracking, the prediction model and to define the harm value of a potential collision with it.

4.3.6 Moving objects prediction

In the driverless vehicle context, safety is associated to collision free trajectories. Since the world model provided by the perception module is the only information available for the planning we have to ensure that the trajectories without collisions generated in the predicted world, will remain free of collisions during their realization in the real world. To ensure this the world model needs to do *consistent predictions*: predicted free space has to be effectively free in the real world future.

The future observations of the moving obstacles need to be inside the predicted occupied area. Integrating adequately the model error into the predictions allows to have consistent predictions. However, too loose predictions (large models errors) will generate large banned areas forcing the planning to be too much conservative. Figure 4.6 illustrates the notion of consistent and conservative predictions for one obstacle.

In order to have a consistent prediction, we do not only have to deal with the measured moving obstacles, but also with not yet observed ones. At the unobserved limits of the field of view frontier we have to assume the possible appearance of moving obstacles.

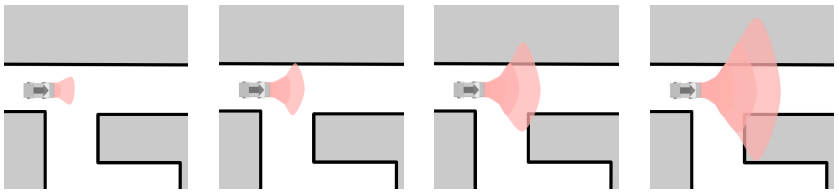


Figure 4.6: Illustration of the conservative prediction of a car position in time. Time grows from left to right. Pink areas indicate the reachable space of the car. White areas are free space

To ensure trajectories free of collisions, we need to suppose the worst case, i.e. the presence of obstacles moving directly toward the current robot position at the maximum expected speed. Creating such virtual obstacles will force the planning module to generate a trajectory conservative enough to deal with the sudden apparition of new obstacles. If the length of the unobserved limit portion allows it the appearance of cars will be supposed, if not only pedestrians will be assumed.

For the pedestrians, we use a Brownian motion model for the prediction. For cars, we use a maximum steering, maximum acceleration prediction model. Unknown objects are supposed to have constant speed (since they usually correspond to spurious measures, see section 4.4).

The uncertainty of the odometry displacement is composed with the prediction of the currently observed moving obstacles is composed in order to estimate the weighting factor $P(x_i \textit{ is static} \mid \textit{previous measures})$ that is fed back into the scan matching method.

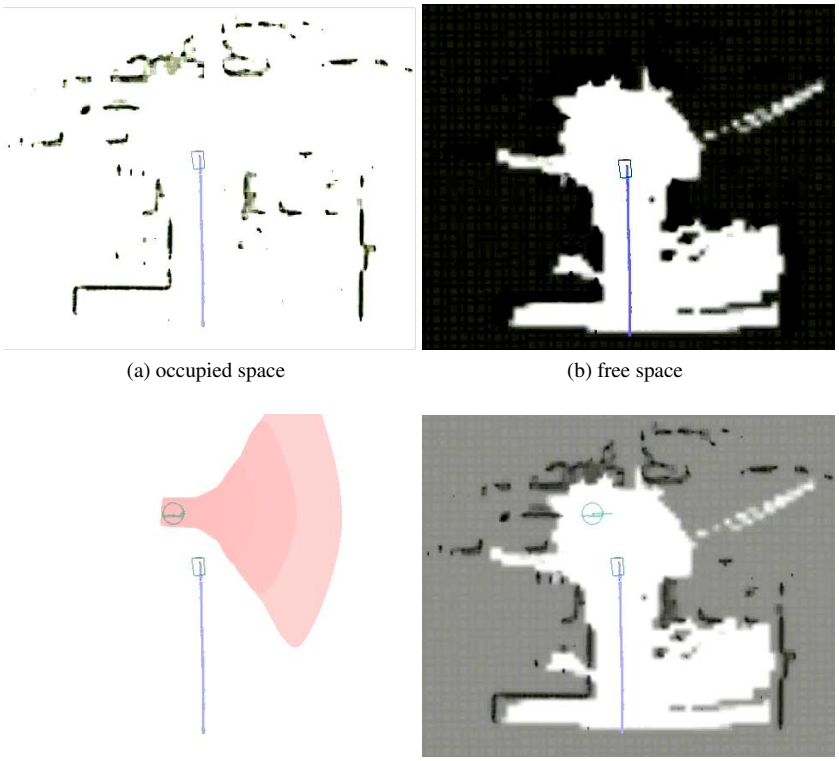
4.3.7 Evaluating the harm function

The world model built up to now (named $w(t_0)$) includes the constructed the map of the free occupied and unobserved space, the detected and tracked the moving objects and their associated conservative prediction models (see figure 4.7). Once these elements are available estimating the harm function $h(x(t), w(t))$ described in chapter 2 is fairly easy.

Depending on the available a priori information, the frontier between free and unobserved space will be considered as a static wall or as a wall of incoming moving objects.

Then, given a state $x(t_1)$ to check, the current world model $w(t_0)$ is extrapolated until t_1. Static objects and unobserved space stay as they are, and a conservative prediction of the moving object occupancy space is used to estimate the traversable space at t_1.

Using this prediction, we check if $x(t_1)$ is in collision, and assign the harm value depending on the kind of object. Ordered from highest to lower harm we have pedestrians, cars, and static obstacles. Unobserved areas should be unreachable without first colliding with a static or moving element in the model. Free space causes zero harm.



(a) occupied space

(b) free space

(c) detected moving object and conservative prediction

(d) SLAMMOT output

Figure 4.7: Illustration of the components of the internal representation of our SLAMMOT algorithm and the actual output of the implemented software (bigger version at 4.10)

A more sophisticated implementation may consider the relative speed at the time of collision.

4.3.8 Algorithm

Our laser scanner based SLAMMOT algorithm (summarized in figure 4.1) can then described using the pseudo code presented in figure 4.8.

The tuple (*vehicle_state, occupied_map, free_map, tracked_moving_objects*) constitutes our current world model $w(t_0)$ and is the information used to predict $h(x(t), w(t))$. Figure 4.7 illustrates the elements used to build the world model.

EstimateStaticMeasuresAPriori and UpdateTrackerMovingObjects are provided by the moving objects tracker described in section 4.3.5. PredictVehiclePosition and Updat-

Recursive function LaserSimpleSLAMMOT

Require: $(vehicle\_state,\ occupied\_map,\ free\_map,\ tracked\_moving\_objects)$

1: $laser\_scan \leftarrow$ AcquireNewScan()
2: $scan\_weights \leftarrow$ EstimateStaticMeasuresAPriori($tracked\_moving\_objects$, $laser\_scan$)
3: $predicted\_vehicle\_position \leftarrow$ PredictVehiclePosition($vehicle\_state$)
4: $corrected\_vehicle\_position \leftarrow$ EstimateDisplacement($predicted\_vehicle\_position$, new_scan, $scan\_weights$)
5: $moving\_objects\_measures$, $static\_objects\_measures \leftarrow$ DetectMovingObjectsMeasures($occupied\_map$, $free\_map$, $laser\_scan$, $corrected\_vehicle\_position$)
6: $tracked\_moving\_objects \leftarrow$ UpdateTrackerMovingObjects($moving\_objects\_measures$, $tracked\_moving\_objects$)
7: $occupied\_map$, $free\_map \leftarrow$ UpdateMaps($corrected\_vehicle\_position$, $laser\_scan$, $occupied\_map$, $free\_map$)
8: $vehicle\_state \leftarrow$ UpdateVehicleState($vehicle\_state$, $corrected\_vehicle\_position$)
9: **return** $(vehicle\_state,\ occupied\_map,\ free\_map,\ tracked\_moving\_objects)$

Figure 4.8: SLAMMOT algorithm pseudo code

eVehicleState are provided by an Unscented Kalman filter (see section 3.4.4). EstimateDisplacement is provided by the scan matching algorithm described in §4.3.3. DetectMovingObjectsMeasures corresponds to the method described in section 4.3.4. Finally UpdateMaps corresponds to the incremental update of the Gaussian distributions used to estimate $odds^x_{occ}$ and of the punctual values used to estimate odd^x_{free} as mentioned in §4.3.1 and §4.3.4.

Bounded online computation The described displacement estimation algorithm is based on an optimization procedure. Fixing an maximum number of optimization steps allows to bound his computation time (currently set to 20 steps). In theory cost of tracking moving obstacles grows quadratically with the number of moving objects, however even for more than ten obstacles the computation time related to this task is negligible compared to the optimization step of the scan matching. Updating the maps and the state of the vehicle is done in constant time (proportional to number of measures and the area they cover).

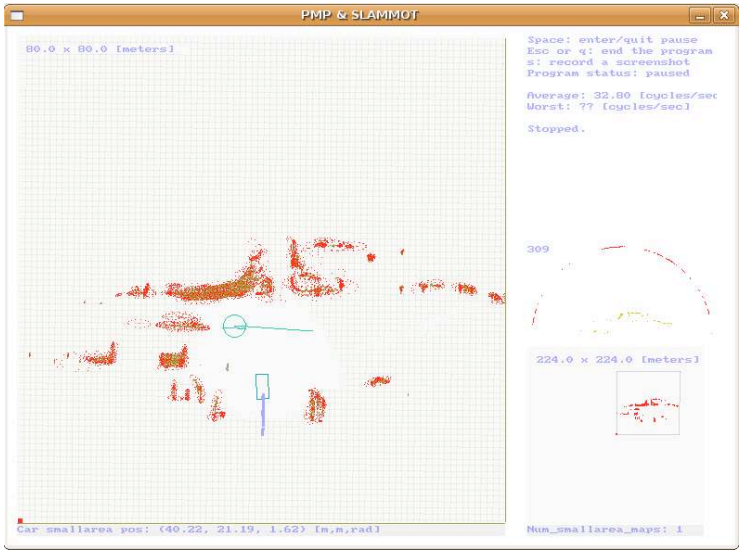


Figure 4.9: SLAMMOT implementation interface. The cross platform application presents to the user the current measure, the vehicle state, its past path, the local map (including free space, static and moving obstacle) and the aggregate of local maps

4.4 SLAMMOT Results

4.4.1 Implementation details

The SLAMMOT algorithms was implemented in C++. Figure 4.9 presents a screenshot of the user interface, showing the current 2D map of the surroundings. In order to reduce the CPU load the interface produces a simplified drawing of the current map, and generates high resolution images (such as figure 4.10) only on demand. The lightweight drawing for the user interface takes an amount of time comparable to the processing time of each new laser measure.

For a local map of 80x80 meters and a grid cell of 1 meters, the memory usage is near 4 Mb per local map, and the computation time to integrate a laser scanner measurements is ~30 [ms] on a modern computer.

4.4.2 Evaluation methodology

In order to evaluate a SLAMMOT algorithm one would like to be able to compare the reconstructed world model with the real world, in order to verify the precision of the reconstructed static elements map, the precision of the vehicle trajectory, the correct detection of moving objects and the precision of their state estimation (speed vector, class, geometric description, etc.).

The problem is that having the ground truth information (of static and moving objects) is costly and difficult to realize in all but very simple or restricted scenarios.

Using a simulator such as the one presented in section 5.5 it would be possible to generate a sequence of measures to feed the algorithm while having access to the ground truth. This would allow to verify the geometric reconstruction and moving objects detection in a set of predefined scenarios.

Even when including the appropriate noise levels on the simulated sensors, this approach provides very limited information since the simulator is based on the same hypothesis than the designed algorithm. In our experience other than validating the implementation and validating the expected behaviour in known situations the simulator is of little use.

Even more, there is a flaw on this evaluation logic. As discussed in chapter 2, for our application we not focus necessarily in precision, but much rather in consistency. Which means that more than having low errors, we want a conservative estimation of such errors.

We used the simulator of section 5.5 to validate the implementation of our SLAMMOT algorithm in simple scenarios, however for evaluation purposes we prefer to present results based on real world measurements.

4.4.3 Recorded data experiment

Setup As a baseline for evaluation we use the data recordings provided by Wang [17, 16]. He uses a fully equiped car (CMU Navlab11) running in the existing traffic flow in the streets nearby the university campus in Pittsburgh, USA.

In order to evaluate the quality and efficiency of the scan matching method no odometry or GPS data is used and no vehicle dynamics model is disregarded (no state filter).

Results A representative example of the results of the algorithm is presented in figure 4.10. It should be noted that on other areas some failure modes are observed. When there is not enough geometric information (such as in corridors) the vehicle trajectory is incorrectly estimated. In these areas the moving objects still correctly detected but their ground speed and trajectory is not. This situation is corrected when using odometry information.

Analysis The result shows that when enough geometric information is available the scan matching alone provides an accurate reconstruction of the vehicle trajectory. It can also be seen that on a horizon of more than 50 meters no noticeable deformation of the incremental map appears despite the 90° turn. This is due partially because of the quality of modern laser scanners and partially because in the proposed method the scan matching is done with the current map instead of just with the previous scan (as commonly done). Figure 4.10 also shows how occupied free space is correctly estimated. It can be also noticed that moving obstacles are detected and tracked.

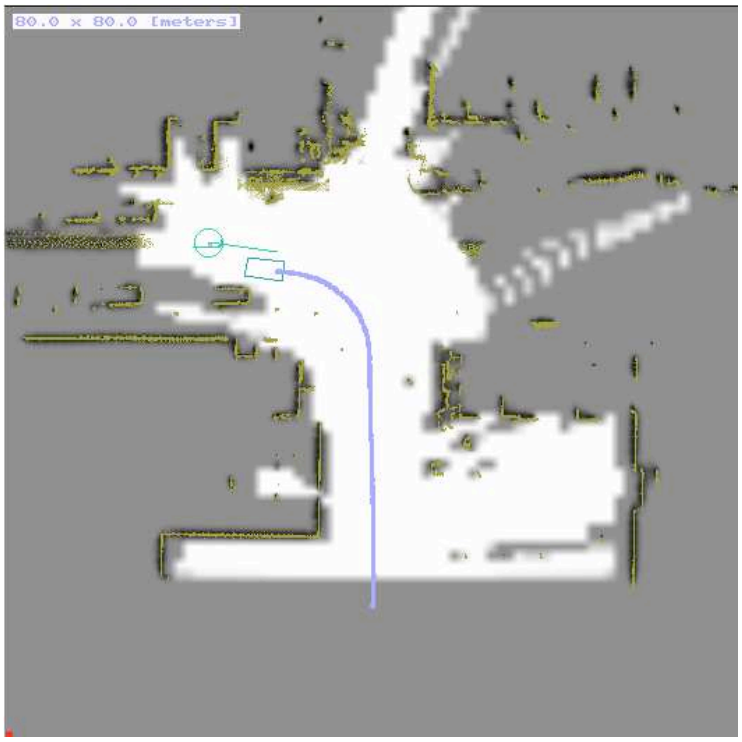


Figure 4.10: SLAMMOT output example. The box represents the current vehicle position, the curved line behind its past path. The circle surrounds a moving obstacle being tracked, the line escaping from it is proportional to the estimated speed vector. White indicates free space, gray unknown, and black occupied by a static obstacles. On top of the static obstacles the accumulation of laser scan measurements is shown

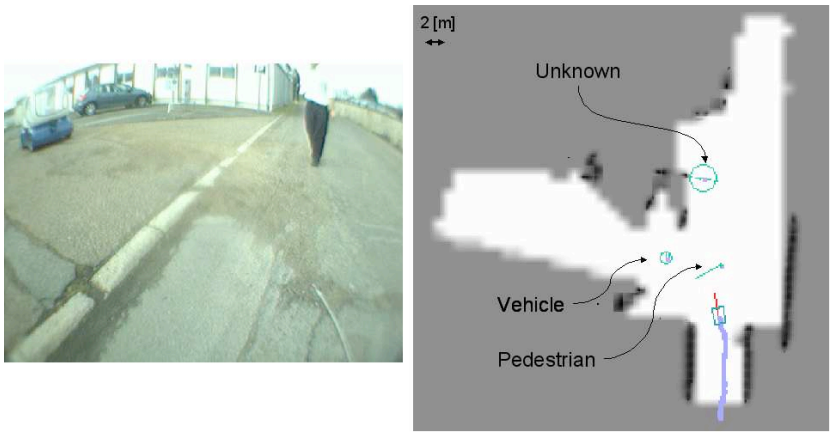


Figure 4.11: SLAMMOT result example. Left: image of the current scene, seen from the vehicle. Right: the current map, including the robot position and past trajectory, the moving pedestrian (and its estimated direction), the stopped vehicle and a spurious measure. See legend of figure 4.10

4.4.4 On board experiment

Setup Using our platform (described in §5.1) and its installed laser scanner we tested our algorithm on our campus. In this case the algorithm is running online using the embedded computing power. Our campus presents a suburban environment with smooth slopes.

When running on our platform, odometry and IMU measurements are available to cope with areas where geometric cues are insufficient to estimate the displacement solely from the scan matching.

Results Outdoor trials on our campus has shown satisfactory results comparable to the ones on recorded data.

In figure 4.11 we present a representative result when running on board the vehicle in our campus. Large, medium and small size obstacles are correctly mapped, the vehicle trajectory reconstructed and the surrounding moving pedestrian and car correctly detected and tracked. Note that this case also presents a failure mode of the laser scanner were one the walls is repeatedly measured at an incorrect distance. This generates a spurious unknown object that has zero speed.

Analysis The only notorious change between the results based on Wang recording and the runs on our vehicle is the presence of obstacles that laser scanner sees as semi transparent. Non modeled, absent on Wang recordings, they are abundant on our campus. From the laser scanner's perspective thin bushes and metallic grids appear as semi

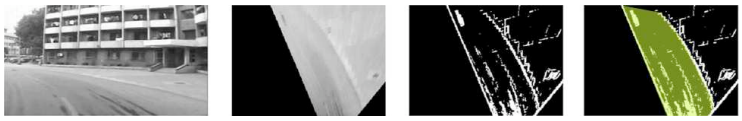


Figure 4.12: Illustration of landmarks detection. From a method similar to [187]

transparent objects. In the proposed SLAMMOT method this objects will be interpreted as moving objects with null speed and thus they do not have a major impact in the system behaviour. Moving objects with null speed are equivalent to static obstacles except they are not used in the movement estimation. Future work on SLAMMOT should explore how to integrate explicitly semi transparent objects in the world model in order to exploit better the available information.

The spurious measurement that appears in figure 4.11 is a rare case (that falls on the very low probability region of the sensor model). In this case the information of traversable ground provided by the vision processing would avoid (see section 4.5) generating unsafe trajectories.

4.5 Traversability estimation using vision

There are four general strategies we can depict for estimating the traversability of a region based on vision:

- Man made landmarks detection (figure 4.12). Urban roads follows specific conventions, usually they are delimited with special painting on the floors. Detecting this marks and supposing that the convention is respected should provide the required traversable area. This approach is straightforward and robust, but brittle since landmarks may not be available everywhere and it requires to manually design the detection method.

- Learning (figure 4.13). Examples of traversable regions of the image are provided to a machine learning algorithm that will then infer the rules that allow do detect traversable regions of an image [188]. This method allows to manage different type of markings and avoid the need of manually designing a detector of landmarks. Its main drawbacks are that it will not necessarily behave correctly in front of situations not similar to the training examples and that it is very difficult to guarantee that the learned examples are “enough”.

- Dense stereo (figure 4.14). In order to get rid of specific landmarks or learned rules, one could imagine to use vision to obtain a full 3D reconstruction of the environment using stereo vision. Then, any large smooth horizontal region and horizontal would be considered as traversable, and any discontinuity or steep slope would be considered non traversable surface. This method fails in three

Figure 4.13: Example of machine learning based ground detection. From [188]

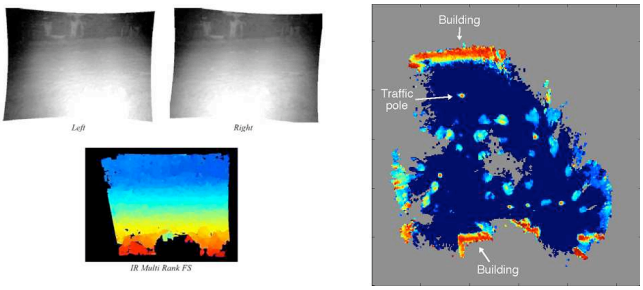


Figure 4.14: Example of dense stereo traversability estimate. Left: an example stereo pair and its depth estimate. Right: height map built from a sequence of dense stereo depth estimates. From [189] and [84]

different ways. First even with a full 3D model it may be not possible to distinguish the traversable space (water, sand, grass). Second the resolution of the 3D reconstruction depends on the resolution of the 2D sensors, the relative distance of the two sensors, and the reconstruction method employed. For standard setups the height curb may be simply not noticed. Finally, despite many years of research the stereo methods have robustness issues when employed in the real world, specially when large homogeneous regions are present.

- Model based (figure 4.15). Instead of using a stereo pair to have a full 3D reconstruction and then search the traversable regions in the 3D space, a model based approach will directly search the traversable areas in the images space [190]. By verifying which pixels match the traversable regions model the computation burden is reduced and the robustness enhanced.

4.5.1 Learning to recognize traversable space

We believe that the model based approaches are the most reliable and versatile ones. In the particular areas where we expect to use our driverless vehicle, the sidewalk is almost at the same level than the road and it is surrounded by grass that is not correctly treated by model or dense stereo methods (due to its repetitive pattern). We need a com-

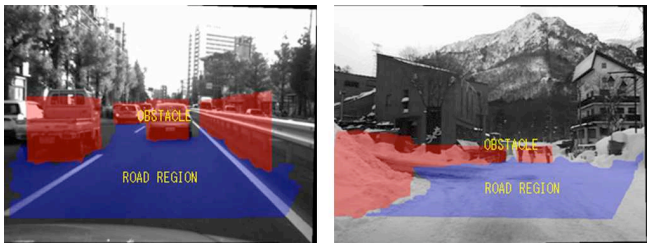


Figure 4.15: Example results of model based ground detection. From [190]

Figure 4.16: Examples of the machine learning train set (used to obtain the results in figure 4.17)

plementary layer to indicate allowed areas and prohibited areas, then we implemented a simple machine learning approach.

Supervised machine learning is a field with a good degree of maturity and multiple methods can be used as black boxes with high effectivity (we suggest [191] as an introduction on the topic). For our application we evaluated using the classic SVM [191] ("support vector machine", based on the statistical learning theory), the IVM [192] ("informative vector machine") and the RVM [193, 194] ("relevance vector machine", both based on sparse Bayesian learning approaches). All three methods can use the so called "kernel trick" to estimate non linear functions, but only IVM and RVM can automatically define the parameters of the kernel (without having to do cross validation as usually done with the SVM).

The a priori information is introduced through some labeled images, such as the one presented in figure 4.16. We solve the problem of segmenting the image into traversable or not regions by transforming it into a classification problem. We use the classical "divide + compute feature vector + classify" setup.

We use a simple rectangular grid in the image space to divide the image into sub-regions (could also be done on the inverse projection of the ground floor, "the top view"). A feature vector is computed for each sub-region. The feature vector to build

is application specific and need to capture discriminative information with respect to the desired classification task. Our strategy consists on first computing a large set of features (~40, based on the features used in [195]) and then use a features selection method to keep only the relevant ones. Choosing only the relevant features allows to accelerate the online execution. We use an "automatic relevance determination" (ARD) kernel with the IVM as a feature selection method. For our road examples, the relevant features are less than ten (the average YCrCb color, and a subset of the Law's masks and the directional gradient masks).

After learning we use the classifier to convert the input image into a binary image and select the largest blob that touches the bottom of the image as the road. Using the calibration of the camera (intrinsic and extrinsic parameters) this blob can be projected to the ground in order to feed an "traversability grid" (like occupancy grid, but estimating the traversability) that will fuse multiple measures in order to build a traversability map. The laser scanner provides the local position at each image acquisition so no additional matching needs to be done between the measure and the "traversability grid". The frontier of the traversable space is fed as a wall of obstacles into the world model (see 4.18). This frontier obstacles will have a low harm value, since it imply only a minor damage to the car without involving other vulnerable beings.

In this work we suppose that the orientation of the camera with respect to the floor stays constant. In applications where the tilt of the vehicle is important, the external parameters of the camera with respect to the floor surface should be estimated online.

4.5.2 Fusion with SLAMMOT

The classification of the regions of the image provides to estimate the traversable space. The SLAMMOT algorithm uses the free space measured by the laser to detect moving objects. The traversable space cannot be integrated into the static part of the SLAMMOT algorithm. The static world output of each algorithm has to be merged into a separate map, using a simple "and" merging criterion (surface is traversable if both methods indicate so). The relative position between the camera and the laser scanner (that allows the transform of coordinates of one map to another) is supposed known and constant.

The traversable regions that touches the frontiers of the observed space with the laser scanner can be used to estimate the possible appearance of vehicles. Cars are unlikely to appear on the frontiers of the observed space frontiers covering a region observed as non traversable.

4.6 Vision results

4.6.1 Setup

The algorithm described in section 4.5 was implemented in C++ and integrated with the SLAMMOT, planning and control software. The learning process in done offline using

different tools (C libraries, Python scripts and R scripts), depending on the selected machine learning and the classification is done online with the mentioned C++ code.

For mono vision task our platform (described in §5.1) is equiped with commercial grade firewire camera (Fire-i from Unibrain). This camera provides 24 bits RGB images with a resolution of 640x480 pixels at 30 frames per second.

The default camera lens provide a field of view that is too narrow to estimate the traversable space. We would like that the images include at least the expected width of a lane a few decimetres in front of the vehicle, and to allow to decide if a turn can be taken or not a few meters away. In order to enlarge the field of view a wide angle lens was installed. The images in figures 4.16 shows clearly the lens distortion (notice the shape of the horizon line). We do not use any kind of calibration or correction of the lens distortion during the classification stage (we use the calibration for the projection from the image to the ground).

As explained in section 4.5.1 the approach is based on decomposing the images in small pieces, selecting the best features and then learn to classify based on these features.

We tested splitting the images in squares with a size between 5x5 pixels to 25x25 pixels. We manually acquired and tagged 25 representative images (all from the desired roads, as dissimilar between them as possible). We used 7 images for the training set (used to learn) and 18 images for the test set (used to evaluate the quality of the method). For a size of 15x15 pixels on 640x480 images this corresponds to ~10000 samples for training and ~25000 samples to testing.

4.6.2 Results

All three methods (SVM, IVM and RVM) provided similar classification results over the test set. their classification error (false positives + false negatives) is inferior to 5% when using a radial basis function kernel with the best configuration of features set and squares size.

The best squares of size was 15x15 pixels, this is dependent of the camera setup. The best features set contained the x,y position of the square, its average YCrCb components, and 6 convolution filters (this means using only 11 of the ~40 features initially provided). The features set is dependent on the training set.

SVM provided a very fast learning stage (a few seconds, thus allowing larger training sets) and resulted in a classifier with a few hundreds of support vectors, good classification results where obtained with the default kernel parameters. The number of support vectors is directly proportional to the classification time and thus to the processing time of each frame.

IVM provided slower learning (a few minutes), but allows to automatically estimate the optimal kernel parameters. It resulted in a classifier with comparable performances with less support vectors (around one hundred).

Finally RVM, while having the slowest learning stage (some tens of minutes) is the method that provided the lighter classifier (a few tens of support vectors) while still

Figure 4.17: Examples of the machine learning method results. The last two images on the bottom-right show some of the failure modes

providing classification errors comparable to the SVM result. With the same classification errors the RVM provides a classifier almost one of magnitude faster than the SVM one.

A selection of the results obtained with the test set are presented in figure 4.17.

For the selected features and patch size, the features computation, RVM classification, polygon detection and projection to the ground plane takes around 1 second on the non optimized C++ code.

4.6.3 Analysis

The general results are quite good (5% of test set error, and given the observations made when running online), even when operating under different light conditions. We observe two failure modes that are intrinsic to the approach:

- When using a small window only local information is available to classify the texture, in some cases local information may not be enough since two similar textures correspond respectively to traversable and non traversable space. This can be countered by the use of larger features vectors using information not only from the local window, but also from its surroundings [196, 195].
- Secondly, even with a high success rate on the test set it is not possible to guarantee the result of the classification when encountered to the real world. New elements may not be correctly classified, even if the machine learning methods offer theoretical guarantees on the generalization capabilities of the learnt classifier. This is a fundamental limitation that can only be countered by the use of large and diverse train sets, by keeping that the vehicle operates in conditions

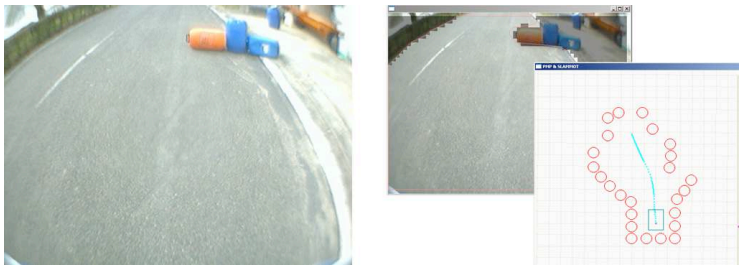


Figure 4.18: Example of obstacles avoidance based on vision. Left: the acquired image. Right: the processed image, resulting obstacles map and safe planning

similar to the training set. Both IVM and RVM are able to provide uncertainty estimates on their classification[1], the class of feature vectors far from the training examples are considered more uncertain. This estimate could be used to provide conservative estimates when confronted to novel situations.

By its own the traversability estimation using vision provides all the required information to safely navigate in a static environment. It is possible then to use the perception-planning-control system described in section 5 only with vision. The displacement estimate is based solely on odometry estimates, the planning includes the growing localization error [197], and the control operates using the odometry estimate as reference. In the figure 4.18 we illustrate how safe planning in a static environment is done based solely on the vision system.

4.7 Conclusion and perspectives

In this chapter, we proposed a perception system capable of satisfying the needs for safe navigation while *imposing zero constraints on the environment* and requiring the strict minimum a priori knowledge.

The perception algorithm described provides a data representation with better uncertainty management, coupled with faster data association and the detection of moving obstacles. Putting aside the locally flat world approximation and examples based learning step, this algorithm makes no strong assumptions on the environment infrastructure or in the moving obstacles' geometry.

It is also worth noticing that the described approach is generic, behind able to directly work with additional sensors, different sensors, or different vehicle model.

Despite the nice properties of the proposed system there still large room for improvements in the quest of a perception system "as good as humans, and better".

[1] although the RVM method has a fundamental flaw causing overconfident predictions

During the chapter we discussed the need to provide conservative predictions. In some situations the lack of additional information forces the system to provide over-conservative predictions, specially for the yet non observed moving obstacles. Humans make a large use of "seeing through the obstacles" to judge the sudden appearance of moving obstacles, for instance, behind a row of parked cars. The current laser based solution is unable to provide such a kind of observation, leading to an over-conservative driving behaviour.

One of the intrinsic limitations of the moving obstacles detection method proposed, is that if an object is never seen moving, it is considered as static; its predicted position will be immutable. However, humans are able to distinguish a parked car from a car waiting behind a semaphore. To manage this situation some level of a priori knowledge is required ("cars contain humans; if on the driver sit, then the car may move", "how do cars look like ?", "how do driving humans look like ?"). Since this issue affects the safety of the system it is worth exploring.

From the used assumptions, the locally flat world assumption is probably the most constraining one (think about cities likes San Francisco or Valparaiso). How to better deal with a non flat world ?

Given the humans example, it is known that driving using only sound and image sensors is possible. Laser scanners being pricey active sensors, there is an interest on exploring an all vision system (see discussion in §7.2.1).

In the next chapter we will show how this perception method is integrated in to a full driverless vehicle system.

Chapter 5

System implementation

> Philosophers have only interpreted the world in various ways, but the real task is to alter it.
> Karl Marx

The purpose of this work is to provide a solution for the perception problem related to driverless vehicle in urban environment. To verify this claim and to explore the interactions between the different components, we integrated the solution described in §4.2 into a complete driverless vehicle solution.

During this chapter we will depict the vehicle employed (section 5.1), describe how the different elements are implemented and interact between them (section 5.2). In section 5.3 we present in more detail the design of the planning and control modules that accompany the perception module. In section 5.5 we present the simulation tool we used to test our system, and finally section 5.6 presents some experimental results.

The author imagined, designed and directly coded or conducted the coding of the software described in this chapter.

5.1 Platform

The vehicle used for our experiments is denominated "Cycab". It is an electric vehicle with two seats designed for experimentation and small scale demonstrations. The vehicle has four motors (one per wheel) and two steering plunger (both front and back wheels can steer). The vehicle has been equipped with diverse sensors, computing and communication capabilities as shown in figure 5.1.

The maximum speed of the vehicle surrounds ~5 [m/s] (~20 [km/h]) however in practice it is manipulated at much lower speeds ~1 [m/s] (comparable to a walking pedestrian).

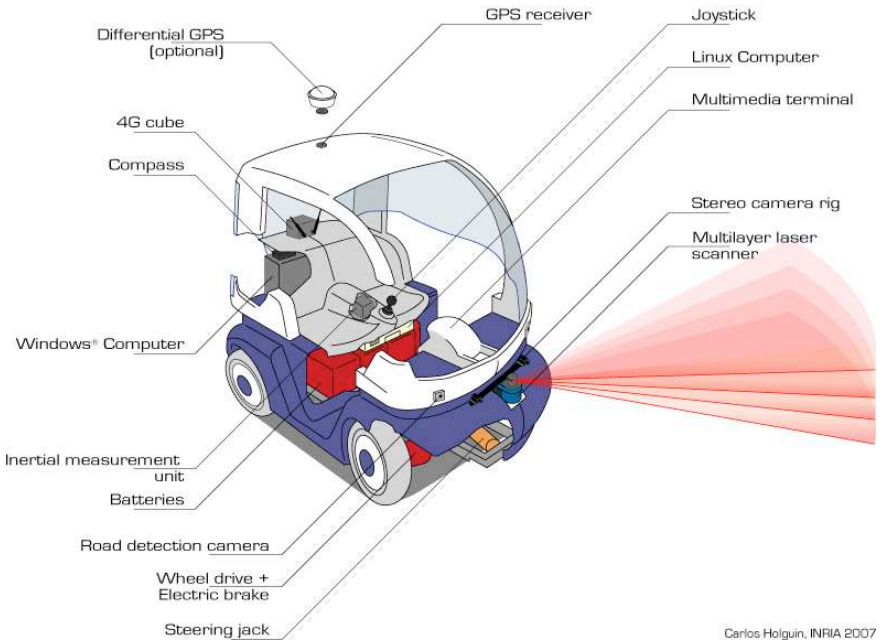


Figure 5.1: Extruded view of the Cycab

5.2 System Integration

We desire to develop the software capable of driving a city vehicle from a point A to a point B automatically as discussed in chapter 1. The general software architecture is presented in figure 5.2. This architecture is a reflection of the six modules presented in the figure 1.10.

The diagram of figure 5.2 includes the main software components, the data exchange between them, and the used vehicle sensors and actuators.

The nucleus perception, planning, control was implemented into a single cross-platform multi-threaded C++ application running in the vehicle on a standard 3.3 [GHz] dual core PC under Windows XP.

5.2.1 Human machine interface

A screenshot of the final user interface is shown on figure 5.3. Using the software Google Earth [198] the user can access geographically referenced content, including the road network, the current vehicle position and its planned route. Through this interface it is possible to request the vehicle to reach one of the specified stations. This interface can be accessed directly inside the vehicle or through the wireless network from a remote location.

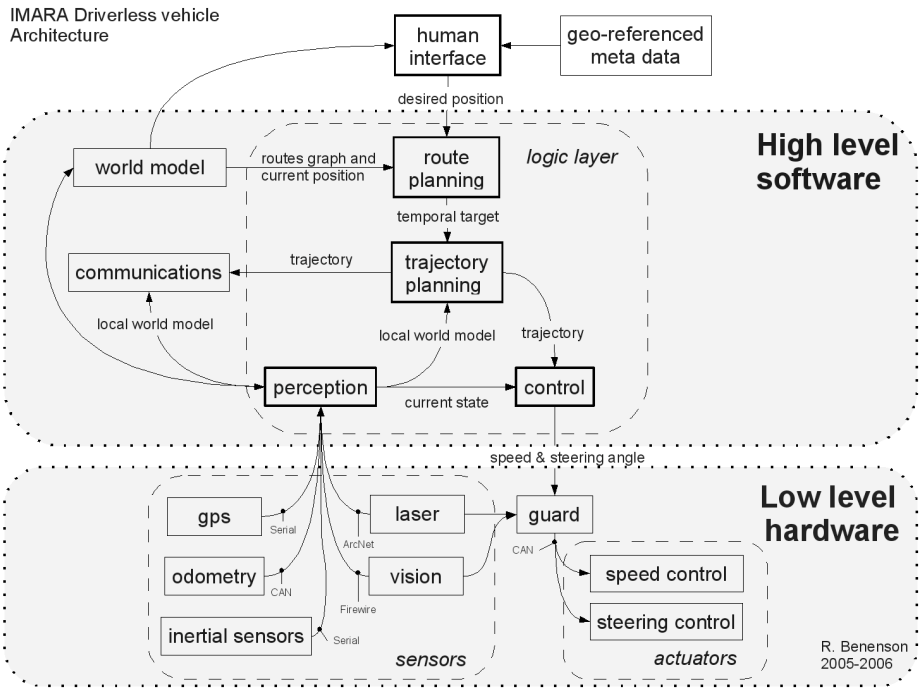


Figure 5.2: Driverless vehicle system architecture

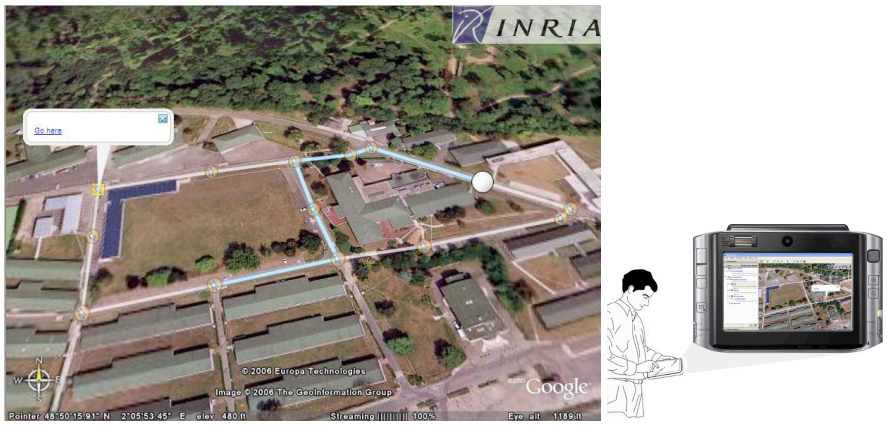


Figure 5.3: Left: route planner interface. Right: illustrating a user doing a request for the vehicle.

5.2.2 Route planning

The route planner is implemented as a server (programmed in Python), providing the content for the human machine interface (serving Google Earth KML files and related content) and exchanging data with the vehicle (using XML-RPC) in order to know its current position and provide its next desired position.

The route planner uses the simple Dijkstra's algorithm to find the shortest path between the current position and the requested position.

5.2.3 Trajectory planning

Given the next desired position received from the route planner and world model provided by the perception module, the planning will compute a trajectory that respects the vehicle limitations, avoids collisions and moves towards the goal.

We use a Partial Motion Planning approach to solve this problem. The planner is by nature any time interruptible, which means that it respects the real time constraint, but also means that there no bound on much computation is desirable.

Current C++ code implements different exploration methods, all of them running as partial motion planners. Almost 50% of the CPU is dedicated to this task. Using a standard PC, a retrieving a trajectory at 2 [Hz], the planner is able to provide a trajectory of a few tens of meters in common scenarios.

For more details on the planning problem and solution the reader can consult the section 5.3.1.

5.2.4 Control

Given the trajectory defined by the planner and the constant state updates from the perception, the control commands the actuators in order to track the trajectory.

The electric vehicle already includes low level controllers for each of the controlling motors. At the application level we are only concerned on setting the reference steering and speed values. We suppose that the low level closed loop response is much faster than our desired high level commands change rate.

An initial implementation based on a simple proportional control proved unsatisfactory to ensure a bound on the tracking error. Currently we use a sophisticated control method called "dynamic feedback". The car model defined in 3.2 is a non linear, non holonomic system. By extending the state of the system and using a non linear mapping it is possible to transform it into a linear system and then apply a linear control over it. For more details on the control method the reader can consult the section 5.3.2.

When implemented in C++ the computation time of this method is negligible with respect to the computation cost of perception and planning.

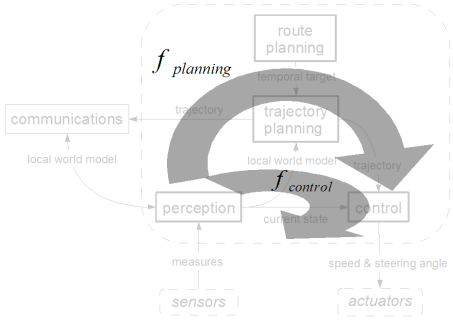


Figure 5.4: Main two logic parallel execution loops

5.2.5 Communication

The vehicle has a wireless networking gateway capable of establishing ad hoc vehicle to vehicle communications between nearby vehicles [199]. The communication module allows them to exchange information related to the perceived world model and the planned trajectory. The issues and solutions related to collaboration between vehicles is detailed in chapter 6.

5.2.6 Execution flow

Both perception and planning algorithms are designed to incrementally and iteratively construct a solution which enables an efficient and simple interweaving. Current implementation profits from the parallel processing of modern processors.

Each of the sensor and processing modules, the communication, the planning modules run in parallel threads. The code has two main parallel logic loops, as illustrated in figure 5.4.

Perception - control

The fastest loop on the software is associated with the control loop. The world model is updated at a frequency $f_{control}$ that can be set up between 10 and 20 $[Hz]$. The fastest this loop runs the easier will be to estimate the displacement of the vehicle and the better will be the tracking of the planned trajectory. At each iteration the map of static and moving obstacles is updated, the vehicle states is updated and transmitted to the control module which will then send a new command to the motors of the vehicle.

Inside the perception module the vision processing runs separately from the laser processing, in an asynchronous manner. This the traversability of the road surface is supposed a static element there is no strong constraint on the update frequency of the image processing. Current implementation process the images in a low priority thread.

Observed output rates are near 1 $[Hz]$. Since this module runs in its own thread, adding one more core to the processor could provide a ten fold increase on the throughput of the images processing.

Perception-Route planning-Planning-Control

This second loops runs at a frequency $f_{planning}$ and is currently setup at 0.5 $[Hz]$. This loop defines the refresh rate of the plan that the control module executes. Depending on the desired trajectory update rate a trade-off is made between reactivity and planning horizon. At each iteration the new trajectory is passed to the control module, the perception provides an updated world model to the planner, the route planner updates the desired goal, and the planner starts a new search for an optimal safe trajectory.

Due to the partial motion planning approach, at the time t_0 the planner provides a plan for $t \in (t_0, t_0 + 1/f_{planning}]$ and starts computing a new plan for $t \in (t_0 + 1/f_{planning}, t_0 + 2/f_{planning}]$. This imply that the world model estimated at t_0 will be used to estimate $h(x(t), w(t))$, $t \in (t_0 + 1/f_{planning}, \infty)$.

5.3 Completing the trinity

In section 5.2 we described how the general systems is implemented and how the components interact. The core components that provide the autonomous function to the vehicle are the perception, planning and control modules. In chapter 4 we provided a detailed description of the perception module. While not a core contribution by themselves, the implemented planning and control algorithms are based on non trivial state of the art solutions. Since their implementation is also a key element for the success of the system implementation we consider meaningful providing a deeper presentation.

Readers not interested in the details of planning or control algorithms may jump directly to the section 5.5.

5.3.1 Motion planning

The purpose of driverless vehicles is to move people and goods, thus motion planning is the main decision module of our mobile robot. Motion planning is a classical problem in robotics and years of work exist on the topic, we invite the reader to consult [200] for a presentation of the classic approaches. While motion planning of industrial robots is considered a problem with good enough solutions, the problem of safe planning for non holonomic robots in unknown dynamic environments remains an active area of research.

Since the purpose of the perception system is to provide information for decision making, it is relevant to have an understanding of the motion planning process in order to define the needs over the perception process.

In this chapter we will present the approach we used to solve the motion problem. It is based on the works on partial motion planning with inevitable collision states verification [15].

Problem definition

The purpose of motion planning is to define a plan that upon execution will safely move the robot from its initial position to its desired position.

As mentioned in chapter 1 for our application this problem can be split into two levels: route planning and trajectory planning.

Route planing Route planning will define the sequence of roads to use to reach the desired position from the current position. If a map of the road network is provided, and the current and desired position in map are known, then the route planning for a single vehicle becomes the classic problem of finding the shortest path on a directed graph. The issues of route planning when considering the current and future traffic, or when the roads map is incomplete or incorrect are important research problems beyond the scope of this dissertation.

Current navigation systems integrated in commercial vehicles show that route planning for a single vehicle can be considered a solved problem. The output of the route planner is a sequence of positions (inflection points on the roads) that the vehicle needs to reach.

For route planning and execution the perception system needs to be able to localize the vehicle in the known roads map (current street segment and bearing).

Trajectory planning Trajectory planning will define the state of the vehicle in space and time in order to transit each of the streets defined by the route planner.

As discussed in previous chapters, in praxis the vehicle has only access to an incomplete and uncertain model of the world. This model is constantly being updated as new measurements are acquired. Since the robot does not have information off all the streets ahead, and that the world model is constantly being updated, only a partial plan can be computed at any instant. This plan needs to be regularly updated to take into account the new information.

The trajectory planning problem can be seen as an *optimization problem with constraints*. Starting from the current state find the sequence of states that approaches the goal, with the following constraints:

- Computation time: the optimization algorithm needs to have a bound on the computation time, in order to provide a solution with a controlled periodicity.
- Feasibility: the computed trajectory needs to be feasible by the robot. Using the a control algorithm the robot should be able to track the sequences of states in time with a known bound on the tracking error.

- Safety: we need to provide guarantees that the robot motion will be harmless.
- Reaching: it is expected that the solution, while respecting the previous constraints, allows the vehicle to approach the final goal.

The solution method should be able to do this online (as the robot moves), and to deal with the non trivial situations encountered in the city (pedestrians, moving cars, incorrectly parked cards, stopped trucks, etc.).

Tractability The space of the solutions is defined by all the possible sequence of states. Finding the optimal sequence that respects the feasibility and safety constraints while reaching the nearest state to the goal is in general intractable through exhaustive approaches. Also the problem is non convex so classical optimization methods are not viable.

Related previous works As discussed in chapter 2, the safety constraint alone rule out the classic methods. Other than our proposal we could mention two relevant different approaches for driverless vehicles. The first is one [201] will try to do a plan discarding the feasibility, safety, and will focus on finding (and updating) efficiently a coarse path. In a second step, the vehicle will use a best effort approach to follow that path and check safety. This two stages approach is unsatisfactory. The best path for a holonomic vehicle is likely not the best path for a non holonomic vehicle, or worse, it could be non feasible at all. Viable in simple situations, this approach will tend to fail on cluttered scenes. Second, in their proposal the actual safety verification are unsatisfactory with respect to the criteria mentioned in chapter 2. A second interesting work is focused on planning for human driver assistance [38]. While the application is different, they propose a full state-time trajectory planning taking into account both the vehicle, the environment (highway situations) and human comfort constraints. In this work the problem is carefully formulated as a very large optimization problem and then special optimization methods are used to find the best solution. While proposed in a non robotic context, this method is apparently fast enough for online execution on robots. It is however unclear how this optimization methods will behave in cluttered or complex environments, if the computation time can be bounded, and if the optimization procedure still find adequate solutions.

Our approach provides good enough solution at any time while respecting all the constraints of the problem, it is similar to [202, 21, 203, 15].

Trajectory planning in dynamic environment

A usual trade off for this kind of large problems is to trade an intractable optimization problem searching for the best solution, for a search problem seeking a good solution. Following the description from [204, 205] a search problem is a structure containing four fields: “states” (describing space of solutions exploration), “operators” (transition

```
Algorithm GenericSearch
given (problem, initial_state, queue_up)
candidates ← MakeCandidatesQueue( problem, initial_state )
while candidates is not empty do
 candidate ← RemoveFront( candidates )
 if GoalTest( candidate ) then return candidate
 new_candidates ← Operators( candidate )
 candidates ← queue_up( problem, candidates, new_candidates )
end while
return failure
```

Figure 5.5: Generic search algorithm

between states), "goal test", and a "path cost" (defines the cost over a sequence of states).

One can define a generic search algorithm, as described in figure 5.5, given a search problem, an initial state and a queuing function this algorithm will either provide a solution that reaches the goal or will exit in a failure mode. This form can be used to describe most of the classic search methods such as depth-first, breadth-first, greedy, beam, hill climbing, A*, among others. In our context each *candidate* contains a sequence of states in time. Operators allows to extend a sequence with a new state, and the *queue_up* function defines the priority on which the candidates should be expanded.

Let us now see how we will adapt this algorithm to manage each of the constraints of our particular problem.

Computation time Clearly the generic search algorithm does not provide any bound on the time used to find a solution. Even worse it may even not provide any solution. In the context of a moving vehicle suddenly not knowing "what do next" could lead to disastrous situations. In order to solve this, the generic search algorithm is modified into an algorithm that can be interrupted at any time, as described in figure 5.6.

Anytime access to best effort solution In general this algorithm may never return, but will provide a solution as soon as it is interrupted. When interrupted it will provide the best solution found up to now. This solution may or may not satisfy the GoalTest condition. We have trade guarantees on reaching the goal for guarantees on computation time. However, as previously discussed in chapter 2, in our setup the reaching the goal is probably not possible anyway. Actually, we trade off finding the best existing solution to finding the best computable solution in the given time slot. Even if the computation time is long enough to find a solution that satisfy GoalTest, the algorithm will

Algorithm AnyTimeSearch
given (*problem, initial_state, previous_candidates, queue_up*)
candidates ← MakeCandidateQueue(*problem, initial_state, previous_candidates*)
do
 best_candidate ← FindBest(*candidates*)
 if Interrupted() **then break**
 candidate ← RemoveFront(*candidates*)
 new_candidates ← Operators(*candidate*)
 candidates ← *queue_up*(*problem, candidates, new_candidates*)
while more that one *candidates*
return *(best_candidate, candidates)*

Figure 5.6: Search algorithm that can be interrupted at any time

continue to explore search for better alternatives to reach the goal. The algorithm will stop by itself if all the alternatives to the best candidate are determined as non worth of being continued (a *candidate* generate an empty *new_candidates* set).

In failure mode this algorithm will return the initial state. Such a failure could occur only if the time slot is suddenly shortened or if the conservative property of the world model are violated. This conditions are considered highly unlikely in our setup. If such a situation would arise an alternative plan should be executed (defined in the past, or as the result of a parallel planner considering collision costs, for instance).

Notice that the algorithm of figure 5.6 takes as input an initial set of candidates, allowing to reuse previously explored options when resuming computation after an interruption. Also notice that since the problem may have been updated ("path cost" related to the world model, "goal test" related to the desired position of the robot), it is likely that previous solution candidate are not viable anymore and MakeCandidateQueue needs to do a purge.

For details on how this computation scheme is integrated in the overall system, the reader may consult chapter 5.

Feasibility In order to ensure the feasibility of the resulting trajectory the search is not done on the vehicle in the states space, but in the vehicle commands space. Then the result of the search algorithm will be a sequence of commands in time. A model of the vehicle dynamics is used to integrate this commands over the initial state in order to retrieve the desired sequence of states in time that will be feed to the vehicle controller. The search algorithm will use this model of the vehicle to predict the future states of each commands set candidate and verify that they respect safety and reaching constraints.

As a side effect the motion planning provides a sequence of command for open loop motion of the vehicle. This information may also be used for by the controller module

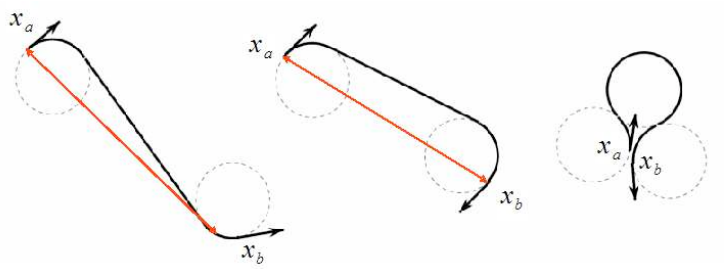


Figure 5.7: Example of different metrics to measure the distance between two vehicle states. In black the continuous curvature metric, in red the 2D euclidean distance. The circles indicate the maximum curvature of the vehicle path

(as initial conditions for an online model based control approach, for instance).

As long as the model adequately describe the vehicle capabilities, this dual space search ensures the feasibility of the resulting states in time sequence.

Safety Safety is at the core of our application. Chapter 2 discuss this issue. We need to ensure that the result will respect the proposed criteria. Since the algorithm can be interrupted at any time, each new candidate could immediately become the returned best trajectory. Consequently we have to ensure that every candidate generated through Operators respects definition 2.9.

In praxis, and given our anytime search in the commands space, this means that we use a conservative prediction to ensure that no harm will be done by action, and that each candidate trajectory finish in state with zero speed. Please consult chapter 2 for a justification of this approach.

Reaching The reaching of the goal is obtained on a best effort basis. The results strongly depends on the specific algorithm employed. The elements *queue_up* and the "path cost"[1](used in FindBest) play a key role. The reader can consult the section 5.3.1 for some instances of such elements.

With respect to reaching we can mention that usually the "path cost" will include a measure of distance between the last state of the trajectory and the desired goal. When measuring the distance between two vehicles it has been shown that the use of an adequate metric have an important influence on the resulting robot motion [206]. Figure 5.7 illustrate some examples, comparing the standard euclidean distance with the continuous curvature metric [207, 208]. The continuous curvature metric evaluates the length of the shortest path between two cars configurations, and thus it is an adequate metric for planning the motion of non holonomic vehicles.

[1]"path" in the search space of the optimization, not in the geometric space of the vehicle

Specific algorithms

Given the presented framework, multiple search algorithms are possible. Actually, most of the existing methods can be adapted to this approach. Once adapted all of them will respect the constraints presented in section 5.3.1. In the next paragraphs we present a few some of the methods explored.

Reactive behaviour The most trivial approach consist on a pure reactive behaviour.

Following the notation of [23], let $p(dt)$ be a *plan*. A plan is sequence of timed commands for the vehicle: $p(dt) = \{(u_1, dt_1), (u_2, dt_2), \ldots, (u_n, dt_n)\}$, where $dt = \sum_i dt_i$. When a plan $p(dt)$ is executed at state $x(t)$ the vehicle will follow the trajectory $\pi(x(t),\ p(dt))$. When a second plan $p'(dt')$ is concatenated to π we will write $\pi'($ $\pi(x(t),\ p(dt)),\ p'(dt')$).

Let $S = \{p_1(\infty), p_2(\infty), \ldots\}$ be a subset of all the possible plans, which guarantee that the vehicle will reach speed zero (stop) in a finite time horizon. For our application we define S as a small group of fixed decelerating commands, keeping the current direction or steering at different rates. Since all of our plans have a constant deceleration, they will stop the vehicle in a finite time.

Now, let $\widetilde{U}$ be a finite set of commands for the vehicle. A reactive planning method will provide at each invocation the next command to execute. In our case this command will be chosen from $\widetilde{U}$[2]. In this particular implementation the function Operators always returns an empty set, at each invocation the AnyTimeSearch will return a *best_candidate* containing a single command.

The best command is selected as indicated in the algorithm of figure 5.8. For each command we verify that one of the stop plans in S allow to stop without collisions, and then for thus who do we select the one which leads nearest to the goal.

When taking into account all the constraints of the system, *reactive planning can be safe*. Its main drawback however, is that since it consider only one step ahead it is prone to get stuck in local minima ("cul de sac" scenario). The logical extension consist then in modifying the algorithm to incrementally build a plan.

Mixed greedy and random exploration The idea is to use the classic greedy and rapid-exploring random tree (RRT) methods to generate new candidates plans. This two methods are well known and described in detail [200].

In a greedy approach the reactive planning previously mentioned is recursively applied to generate a sequence of commands that will go straight to the goal. The method will detect the presence of a dead end if the vehicle stops in front of static obstacles before reaching the goal. When this situation arise, the last command of the plan will be tagged as "leading to a dead end" and an alternative will be explored (the next best

[2] In order to ease the implementation of the safety management, the commands in S should be contained in $\widetilde{U}$

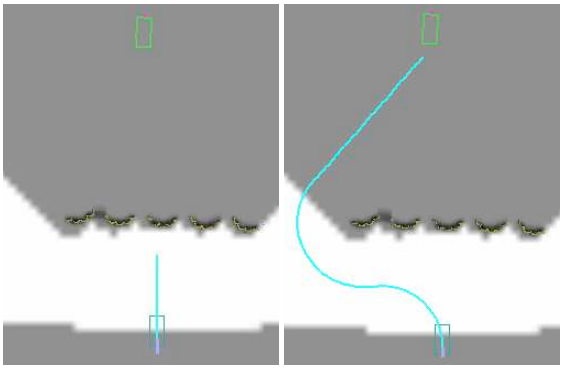
```
FindBest ← ReactiveFindBest
function ReactiveFindBest(candidates)
 safe_candidates ← {c ∈ candidates | IsSafe(c,S) = True}
 if safe_candidates is empty then
  return failure
 else
  return FindNearestToGoal(safe_candidates)
 end if
end function

function IsSafe(candidate_command, plans)
 foreach plan in plans
  trajectory ←π′(π(initial_state, candidate_command), plan)
  if trajectory is collision free then
   return True
 end foreach
 return False
end function

function FindNearestToGoal(candidates)
  return argmin PathCost(π(initial_state, candidate)), candi-
date ∈candidates
end function

Operators ← ReactiveOperators
function ReactiveOperators(candidates)
 return None
end function
```

Figure 5.8: Reactive planning variant of AnyTimeSearch



(a) Greedy planning (b) Mixed greedy and random planning

Figure 5.9: Greedy planning versus mixed greedy and RRT planning. Both cases are given the same computation time. The vehicle at the bottom wants to reach to top of the image, black indicate obstacles

choice). If no path to the goal exists and the algorithm is given infinite computation time, the initial command will be tagged as "leading to a dead end". If a path exists, then it will be found, through a greedy exploration of all possible cases.

The greedy exploration is straightforward and effective, but has low efficiency. A typical failure mode is presented in figure 5.9. When confronted to a wall, the greedy method will explore thousands of plans that lead to a straight collision with the wall. Is only after verifying all of collision plans that an escape plan will be found.

It is well known that random search methods tend to provide surprisingly good results. However in our application it will tend to provide wobbling trajectories, possibly largely suboptimal. Thus we use a mixed approach. Each time a dead end is reached two consecutive binary choices are made: we either select the current best plan or a random plan and we either extend the selected plan towards the goal or to another random point (in an area that could later lead to the goal). Then the selected plan is greedily extended towards the selected goal, until a new dead end is reached. By adjusting the probabilities of this two Bernoulli trials, we control how much the method will diverge from the greedy method. The presence of the random factor avoid the need of exhausting the plans colliding with an obstacle before finding an alternative to surround it. Including a random factor, reduces the computation required to reach the goal. Typical results are presented in figures 5.9 and 5.10.

Another way to reduce the computation is to reuse the set of candidates available at the last interruption of the algorithm. Since the world model may have changed since the last call, the safety of every plan needs to be checked again. However we avoid recomputing the full integration of the plans to obtain the trajectories (required when doing safety checking). Also since the world model has only incremental changes, it is likely that the previous candidates contains already good plans that can be reused

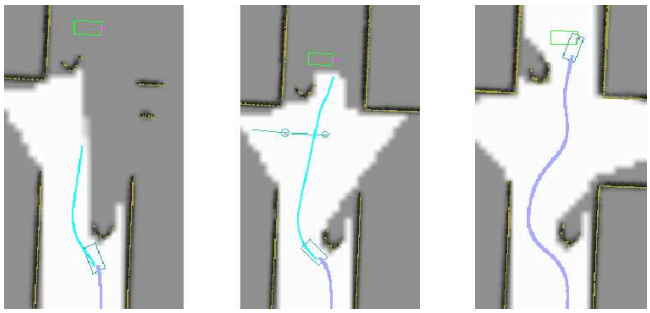


Figure 5.10: Illustrating the planned and executed path resulting from the trajectory planning. Time evolves from left to right. Notice the used model assume that any existing moving obstacle is already observed (academic case)

directly. Then for the same amount of time, more of it is dedicated finding better plans.

5.3.2 Control

As first mentioned in section 1 the automatic control of the vehicle is one of the challenges to be solved. It takes particular importance when taking in consideration the safety constraints, as discussed in chapters 2 and 4, since guarantees on the control are needed to ensure the harmless behaviour of the robot.

In the proposed approach the control problem is casted as a tracking problem. The trajectory planning module computes a sequence of desired states $qd_{t_0 \ldots t_1}$ and the control module is expected to drive the vehicle trying to minimize the error between the actual state q_t and the desired state for that instant qd_t.

The bicycle model presented in the equation 3.1 is the simplest model that captures the movement constraints of the car. A system with two inputs $u_{1,2}$ and a state vector q of four dimensions, as shown in equation 5.1.

$$\dot{q} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ \frac{1}{L} \cdot \tan\phi \\ 0 \end{pmatrix} \cdot u_1 + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \cdot u_2 \tag{5.1}$$

When confronted to such a non linear model multiple control strategies are possible: linearization of the model, model predictive control, sliding mode, feedback linearization; to mention a few.

In [209] a simple non linear control law was proposed. [210] explored the use of sliding mode control for path following. The works of [211, 212] shown that it is possible to separate lateral and longitudinal control through the use of dynamic feedback linearization. Multiple works show dynamic feedback linearization as a theoretically

satisfactory strategy for trajectory tracking [213, 18, 45, 42]. It should be also mentioned that the partial motion planning strategy can be interpreted and used as model predictive control method. We can mention [214] as an application of model predictive control of a car like robot.

In the next sections we will present the controller proposed by [209] and used in [12], and then the more sophisticated control by dynamic feedback linearization.

Naive proportional controller

For many stable systems with a smooth non linearity a simple proportionally feedback control law may be sufficient to achieve the control objectives. For our vehicle system, [209] proposed control law described in equation 5.2.

$$\begin{array}{rclcl} u_1 & = & v & = & v_{ref} \cdot (1 - k_1 \cdot \Delta_x) \\ u_2 & = & \dot{\phi} & = & \phi - (k_2 \cdot \Delta_y + k_3 \cdot \sin \Delta_\theta) \end{array} \tag{5.2}$$

The value v_{ref} is the reference velocity given by the open loop trajectory calculated by the planning module. The parameters k_1, $k_2$and k_3 are tuning coefficients of the control law. These parameters are tuned through trial and error, during these experiments a bound is defined on the probable tracking error. This bound is then used during the planning stage in order to consider the collision risk associated to tracking errors. During execution the tracking error is monitored and a failure mode is raised if this error exceed the predefined bound.

This simple proportional control method provide asymptotic convergence to the desired state. However, as shows in section 5.3.2, when the state evolves in time, the error does not converge to zero. Since the vehicle moves according to a predefined trajectory that is feasible by design, such a constant tracking error is avoidable. In section 5.3.2 we present how this can be achieved through dynamic feedback linearization and in section 5.3.2 we compare both methods.

Feedback linearization

The feedback linearization method is an approach used to transform non linear system into linear ones. Then a linear control can be applied over a transformed non linear system.

Let us suppose a system with the following form:

$$\begin{array}{rcl} \dot{x} & = & f(x) + g(x) \cdot u \\ y & = & h(x) \end{array} \tag{5.3}$$

Both $x \in \mathbb{R}^n$and y are variables evolving in time, $x = x(t)$ and $y = y(t)$. Thus one can compute

$$\dot{y} = y^{(1)} = \frac{\partial y}{\partial t} = \frac{\partial h(x(t))}{\partial t} = \frac{\partial h(x)}{\partial x} \cdot \frac{\partial x}{\partial t} = \frac{\partial h(x)}{\partial x} f(x) + \frac{\partial h(x)}{\partial x} \cdot g(x) \cdot u \qquad (5.4)$$

Using the Lie derivatives notation one can write

$$L_f h(x) = \frac{\partial h(x)}{\partial x} f(x)$$

then equation 5.4 can be wrote as

$$\dot{y} = y^{(1)} = L_f h(x) + L_g h(x) \cdot u$$

Accordingly we can define the following sequence of equations:

$$\begin{array}{rcl} y^{(0)} & = & h(x) \\ y^{(1)} & = & L_f h(x) + L_g h(x) \cdot u \\ y^{(2)} & = & L_f^2 h(x) + L_g L_f h(x) \cdot u \\ \vdots & & \\ y^{(n)} & = & L_f^n h(x) + L_g L_f^{n-1} h(x) \cdot u \end{array}$$

An important property of the non linear system to be controlled is the so called "relative degree r". This is the lowest degree $r \in \mathbb{N} : 1 \leq r \leq n$ where

$$L_g L_f^k h(x) = 0 \; \forall k \in \mathbb{N} : k < r-1 \quad \text{and} \quad L_g L_f^{r-1} h(x) \neq 0$$

Then we can define a transform

$$T(x) = \begin{bmatrix} z_1(x) \\ z_2(x) \\ \vdots \\ z_r(x) \end{bmatrix} = \begin{bmatrix} y^{(0)}(x) \\ y^{(1)}(x) \\ \vdots \\ y^{(r-1)}(x) \end{bmatrix} = \begin{bmatrix} h(x) \\ L_f h(x) \\ \vdots \\ L_f^{r-1} h(x) \end{bmatrix} = T(y) \qquad (5.5)$$

and obtain a new fully observable dynamic system

$$\begin{array}{rcccl} \dot{z_1} & = & y^{(1)}(x) & = & L_f^1 h(x) \\ \dot{z_2} & = & y^{(2)}(x) & = & L_f^2 h(x) \\ \vdots & & & \vdots & \\ \dot{z_r} & = & y^{(r)}(x) & = & L_f^r h(x) + L_g L_f^{r-1} h(x) \cdot u \end{array}$$

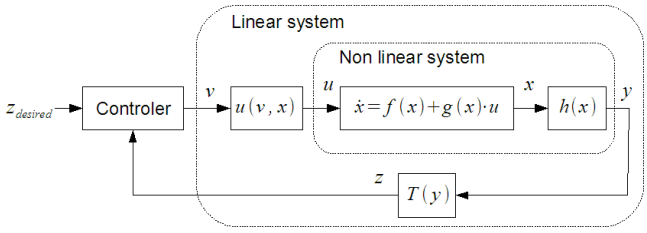


Figure 5.11: Feedback linearization control illustration

Defining a new control variable v such as

$$u(v) = \frac{\left(v - L_f^r h(x)\right)}{L_g L_f^{r-1} h(x)} \tag{5.6}$$

we finally obtain a linearized system based on the transforms 5.5 and 5.6

$$\begin{array}{rcl} \dot{z_1} & = & z_2 \\ \dot{z_2} & = & z_3 \\ \vdots & & \\ \dot{z_r} & = & v \end{array} \tag{5.7}$$

which can be controlled using a simple proportional feedback. An illustration of the linearization is presented in figure 5.11.

Dynamic feedback linearization of the bicycle model Clearly the model 5.1 fits the form 5.3. Given the movement constraints of a car defining a sequence of (x,y) coordinates uniquely defines a sequence of (x,y,θ,ϕ) states.

Let us focus on $x(t)$. From 5.1 we know

$$\dot{x} = x^{(1)} = \frac{\partial x}{\partial t} = \cos\theta \cdot u_1$$

In the definition $\dot{x}$ the input u_1 already appears, however the second component u_2 is not present, thus we need to search for a higher degree.

$$\begin{array}{rcl} x^{(2)} & = & \cos\theta \cdot \dot{u_1} - \sin\theta \cdot \dot{\theta} \cdot u_1 \\ x^{(3)} & = & \cos\theta \cdot \ddot{u_1} - 2 \cdot \sin\theta \cdot \dot{\theta} \cdot \dot{u_1} - \cos\theta \cdot \dot{\theta}^2 \cdot u_1 - \sin\theta \cdot \ddot{\theta} \cdot u_1 \end{array} \tag{5.8}$$

Expanding $\ddot{\theta} = \theta^{(2)}$

$$\begin{array}{rcl} \ddot{\theta} = \theta^{(2)} = & \frac{\partial}{\partial t}\left(\frac{1}{L} \cdot \tan\phi \cdot u_1\right) & \\ = & \frac{1}{L} \cdot \tan\phi \cdot \dot{u_1} + \frac{u_1 \cdot \dot{\phi}}{L \cdot \cos^2\phi} & = \frac{\dot{\theta}}{u_1} \cdot \dot{u_1} + \frac{u_1 \cdot u_2}{L \cdot \cos^2\phi} \end{array} \tag{5.9}$$

Then inserting 5.9 into 5.8 we obtain

$$x^{(3)} = \cos\theta \cdot \ddot{u_1} - 3 \cdot \sin\theta \cdot \dot{\theta} \cdot \dot{u_1} - \cos\theta \cdot \dot{\theta}^2 \cdot u_1 - \sin\theta \cdot u_1^2 \cdot \frac{u_2}{L \cdot \cos^2\phi} \tag{5.10}$$

Notice that now both u_1 and u_2 appear but it is not possible to define a separation of the form $x^{(3)} = f_1(q) + f_2(q) \cdot u$. In order to solve this situation we will redefine the input as $\mu = (\mu_1, \mu_2)^T$ and extend the state of the system using a dynamic compensator such as

$$\dot{u_1} = p1, \qquad \dot{p1} = \ddot{u_1} = \mu_1$$

The extended state vector $\chi = (x, y, \theta, \phi, u_1, p1)$ now includes both the system state and the dynamic compensator state (which is simply a double integrator on the speed command). On its side $\mu_2 = u_2$.

We can now rewrite 5.10 as

$$\begin{aligned} x^{(3)} &= -3 \cdot \sin\theta \cdot \tfrac{1}{L} \cdot \tan\phi \cdot u_1 \cdot p_1 - \cos\theta \cdot \tfrac{1}{L^2} \cdot \tan^2\phi \cdot u_1^3 \\ &\quad + \cos\theta \cdot \mu_1 - \tfrac{\sin\theta \cdot u_1^2}{L \cdot \cos^2\phi} \cdot \mu_2 \\ &= \alpha_x(\chi) + \rho_x(\chi) \cdot \mu \end{aligned}$$

Then following the logic of section 5.3.2 we define the transform

$$T(\chi) = Z(\chi) = \begin{bmatrix} z_1(\chi) \\ z_2(\chi) \\ z_3(\chi) \\ z_4(\chi) \\ z_5(\chi) \\ z_6(\chi) \end{bmatrix} = \begin{bmatrix} x^{(0)}(\chi) \\ x^{(1)}(\chi) \\ x^{(2)}(\chi) \\ y^{(0)}(\chi) \\ y^{(1)}(\chi) \\ y^{(2)}(\chi) \end{bmatrix} = \begin{bmatrix} x \\ \cos\theta \cdot u_1 \\ \cos\theta \cdot p_1 - \sin\theta \cdot \left(\frac{1}{L} \cdot \tan\phi\right) \cdot u_1 \\ y \\ \sin\theta \cdot u_1 \\ \sin\theta \cdot p_1 + \cos\theta \cdot \left(\frac{1}{L} \cdot \tan\phi\right) \cdot u_1 \end{bmatrix}$$

and we can write the linearized system

$$\begin{array}{lllll} \dot{z_1} = z_2 = x^{(1)} & & & & \\ \dot{z_2} = z_3 = x^{(2)} & & & & \\ \dot{z_3} = v_1 = x^{(3)} & = & \alpha_x(\chi) + \rho_x(\chi) \cdot \mu & & \\ \dot{z_4} = z_5 = y^{(1)} & = & & & \\ \dot{z_5} = z_6 = y^{(2)} & = & & & \\ \dot{z_6} = v_2 = y^{(3)} & = & \alpha_y(\chi) + \rho_y(\chi) \cdot \mu & & \end{array}$$

In this particular case the parallel of equation 5.6 is

$$\mu(v) = v = \rho^{-1}(\chi) \cdot \left(\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} - \alpha(\chi) \right) = \rho^{-1}(\chi) \cdot \delta_v$$

with

$$\alpha(\chi) = \begin{pmatrix} \alpha_x(\chi) \\ \alpha_y(\chi) \end{pmatrix} = \begin{pmatrix} -3 \cdot \sin\theta \cdot \frac{1}{L} \cdot \tan\phi \cdot u_1 \cdot p_1 - \cos\theta \cdot \frac{1}{L^2} \cdot \tan^2\phi \cdot u_1^3 \\ 3 \cdot \cos\theta \cdot \frac{1}{L} \cdot \tan\phi \cdot u_1 \cdot p_1 - \sin\theta \cdot \frac{1}{L^2} \cdot \tan^2\phi \cdot u_1^3 \end{pmatrix}$$

$$\rho^{-1}(\chi) = \begin{pmatrix} \rho_x(\chi) \\ \rho_y(\chi) \end{pmatrix}^2 = \begin{pmatrix} \cos\theta & -\frac{\sin\theta \cdot u_1^2}{L \cdot \cos^2\phi} \\ \sin\theta & \frac{\cos\theta \cdot u_1^2}{L \cdot \cos^2\phi} \end{pmatrix}^{-1} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\frac{\sin\theta \cdot L \cdot \cos^2\phi}{u_1^2} & \frac{\cos\theta \cdot L \cdot \cos^2\phi}{u_1^2} \end{pmatrix}$$

Notice that $\rho^{-1}(\chi)$ is ill defined when $u_1 = 0$ making the command μ_2 undefined. This happens since $u_1$is the vehicle speed and μ_2 affects the steering ϕ. When the vehicle does not move ($u_1 = 0$) it does not matter which steering we choose, the error will not decrease. When $u1 = 0$, $\mu_2 = \dot{\phi} = 0$ is a good choice. When $u_1 \approx 0$ the numerical implementation will certainly have problems computing correctly the value for μ_2. Our proposed solution is

$$\mu_2(v) = \begin{cases} 0 & \text{if } u_1 = 0 \\ [0\ 1] \cdot \rho^{-1}(\chi) \cdot \delta_v & \text{if } |u_1| > u_{threshold} \\ k_{\mu_2} \cdot sign\left(\begin{pmatrix} -\sin\theta & \cos\theta \end{pmatrix} \cdot \delta_v \right) & \text{if } 0 < |u_1| < u_{threshold} \end{cases}$$

Where k_{μ_2} and $u_{threshold}$ are suitable values.

In order to obtain an exponentially decreasing error the commands (v_1, v_2) of the linearized system are set

$$\begin{aligned} v_1 &= x_d^{(3)} - k_{x_2} \cdot (x^{(2)} - x_d^{(2)}) - k_{x_1} \cdot (x^{(1)} - x_d^{(1)}) - k_{x_0} \cdot (x^{(0)} - x_d^{(0)}) \\ v_2 &= y_d^{(3)} - k_{y_2} \cdot (y^{(2)} - y_d^{(2)}) - k_{y_1} \cdot (y^{(1)} - y_d^{(1)}) - k_{y_0} \cdot (y^{(0)} - y_d^{(0)}) \end{aligned}$$

Where the coefficients $k_{x_i}, k_{y_i}\ i \in \{2,1,0\}$ are the parameters that define the roots r_x, r_y of the closed loop system. Usually the designer will use $r_x = r_y$, thus the controller has only one free parameter to define.

Reference state interpolation In the proposed scheme the trajectory planning module will generate a sequence of reference state $q_d = (x_d, y_d, \theta_d, \phi_d)^T$. However the controller discussed in section 5.3.2 requires a sequence of reference states z_d.

$$z_d = (x_d^{(0)}, x_d^{(1)}, x_d^{(2)}, x_d^{(3)}, y_d^{(0)}, y_d^{(1)}, y_d^{(2)}, y_d^{(3)})$$

A second implementation issue is that the planning module will generate a discrete reference trajectory with a different sampling rate that the control sampling rate. The planner will use the lowest sampling rate that allows a correct computation of the collisions (low sampling rate mean larger distances between states), while the control

module will use the highest useful sampling rate given the actuators' bandwidth and the state observer update rate.

In order to use the planned reference trajectory for control we need to estimate and interpolate the first and second derivatives of the (x,y). Let us focus on the variable x.

Given δ_c the sampling rate of controller and δ_p the sampling rate of the planner output. At a given time $t_1 = n \cdot \delta_c,\ n \in \mathbb{N}$ we will choose from the reference trajectory four values.

$$x_k = x_d\left(\left\lfloor \frac{t_1}{\delta_p} \right\rfloor + (k-1)\cdot\delta_p\right) \quad k = \{1,2,3,4\}$$

The desired value $x_d(t_1)$ resides between the values x_1 and x_2. In order to estimate $x_d^{(i)}(t_1),\ i \in \{0,1,2,3\}$ we will estimate a third degree polynomial $f(t)$. Then

$$\begin{array}{rcrcl} x_d^{(0)}(t_1) & \approx & f^{(0)}(t_s) & = & a_x \cdot t_s^3 + b_x \cdot t_s^2 + c_x \cdot t_s + d_x \\ x_d^{(1)}(t_1) & \approx & f^{(1)}(t_s) & = & a_x \cdot 3 \cdot t_s^2 + b_x \cdot 2 \cdot t_s + c_x \\ x_d^{(2)}(t_1) & \approx & f^{(2)}(t_s) & = & a_x \cdot 6 \cdot t_s + b_x \cdot 2 \\ x_d^{(3)}(t_1) & \approx & f^{(3)}(t_s) & = & a_x \cdot 6 \end{array} \tag{5.11}$$

The four parameters (a_x, b_x, c_x, d_x) can be estimated using a linear regression. Even more, since we can choose arbitrarily the time position of the samples $x_i$in the $f(t_f)$ function we can define

$$x_i = f(i \cdot \delta_p) \quad i = \{1,2,3,4\} \tag{5.12}$$

and then

$$\begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix} = M^{-1} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

where M is a constant matrix

$$M^{-1} = \begin{bmatrix} (1\cdot\delta_p)^3 & (1\cdot\delta_p)^2 & (1\cdot\delta_p)^1 & (1\cdot\delta_p)^0 \\ (2\cdot\delta_p)^3 & (2\cdot\delta_p)^2 & (2\cdot\delta_p)^1 & (2\cdot\delta_p)^0 \\ (3\cdot\delta_p)^3 & (3\cdot\delta_p)^2 & (3\cdot\delta_p)^1 & (3\cdot\delta_p)^0 \\ (4\cdot\delta_p)^3 & (4\cdot\delta_p)^2 & (4\cdot\delta_p)^1 & (4\cdot\delta_p)^0 \end{bmatrix}^{-1} = \begin{bmatrix} \frac{-1}{6\cdot\delta_p^3} & \frac{1}{2\cdot\delta_p^3} & \frac{-1}{2\cdot\delta_p^3} & \frac{1}{6\cdot\delta_p^3} \\ \frac{3}{2\cdot\delta_p^2} & \frac{-4}{\delta_p^2} & \frac{7}{2\cdot\delta_p^2} & \frac{-1}{\delta_p^2} \\ \frac{-13}{3\cdot\delta_p^1} & \frac{19}{2\cdot\delta_p^1} & \frac{-7}{\delta_p^1} & \frac{11}{6\cdot\delta_p^1} \\ \frac{4}{\delta_p^0} & \frac{-6}{\delta_p^0} & \frac{4}{\delta_p^0} & \frac{-1}{\delta_p^0} \end{bmatrix}$$

Finally given the choice at 5.12 we have

Table 5.1: Controller simulations parameters

Parameter	Value	Units
Vehicle length	1.5	[m]
Max steering	$20 \cdot \frac{\pi}{180}$	[rad]
Max speed	5	[m/s]
Max steering rate	$2 \cdot max\_steering$	[rad/s]
Reference sampling	$1/2$	[Hz]
Control sampling	$1/8$	[Hz]
Vehicle length error	0.1	[m]
Observer noise	$(0.1, 0.1, 0.01, 0.5 \cdot \frac{\pi}{180})/2.0$	[m,m,rad,rad]
Initial position error	0.5	[m]
Reference turning radius	$1.5 \cdot \frac{vehicle\_length}{\tan(max\_steering)}$	[m]
Reference max speed	1.0	[m/s]
$k_1,\ k_2,\ k_3$	$0.5,\ 2.5,\ 1.5$	$\cdot$
$r_x,\ r_y$	$-1+0\cdot i,\ -1+0\cdot i$	$\cdot$

$$t_s = \delta_p + \left(t_1 - \left\lfloor \frac{t_1}{\delta_p} \right\rfloor \cdot \delta_p \right)$$

Using the same procedure we can estimate (a_y, b_y, c_y, d_y) and compute $y_d^{(i)}(t_1),\ i = \{0,1,2,3\}$.

Simulation results

The controllers proposed in sections 5.3.2 and 5.3.2 are evaluated using simulations in Scilab [215]. The parameters of the simulations are summarized in the table 5.1. The parameters of both controllers where tuned by trial an error in order to obtain the lowest error.

In figure 5.12 we compare both controllers when the trajectory describes a speed ramp defined in over a straight line and no noise is present. The trajectory speed starts at zero and linearly increase until reaching the maximum value indicated in table 5.1. As expected both controllers correct the initial error and smoothly converge towards a zero error state.

In figure 5.13 we compare both controllers when following an "S" shaped trajectory, with the same speed ramp as previous example. We can see that when the direction changes in time, the naive controller does not converge to zero error anymore while the dynamic feedback controller does. It can also be appreciated that both controllers present a non exponentially decreasing error (there is a "rebound effect"). This is due to the saturation of the steering angle (that is not considered in the control method),

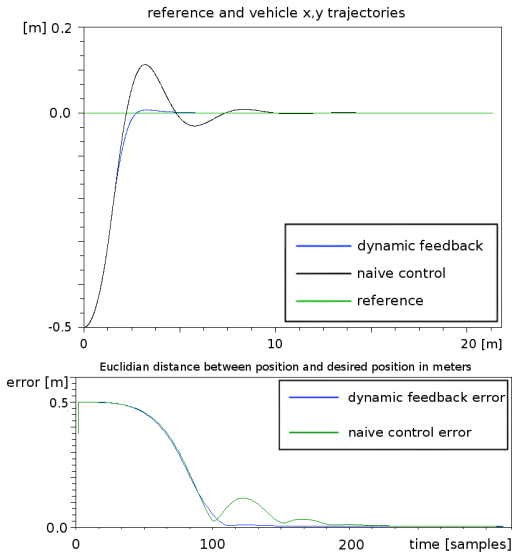


Figure 5.12: Comparison of the two controllers when following a trajectory with linearly increasing speed and constant direction. Top shows the resulting paths, bottom shows tracking error in time

when the error is low enough to avoid saturation (second part of the path) this effect does not appear.

Finally we realize the same simulation but including realistic model error and observation noise (see table 5.1). The overall behaviour is comparable to the perfect case.

Observing the simulation results we can see that the dynamic feedback control method provides a better behaviour when following a curved trajectory under noisy conditions than the naive method. It is also theoretically sounder and much easier to calibrate (only one free parameter).

5.4 Validation methodology

When developing a software that will command a large, expensive and fragile object such as a car, it is very important to be able to validate it without endangering the platform.

Control The control design is first validated using simulations such as the ones presented in section 5.3.2. Then the actual implementation can be validated by interfacing the C++ code with the simulation environment.

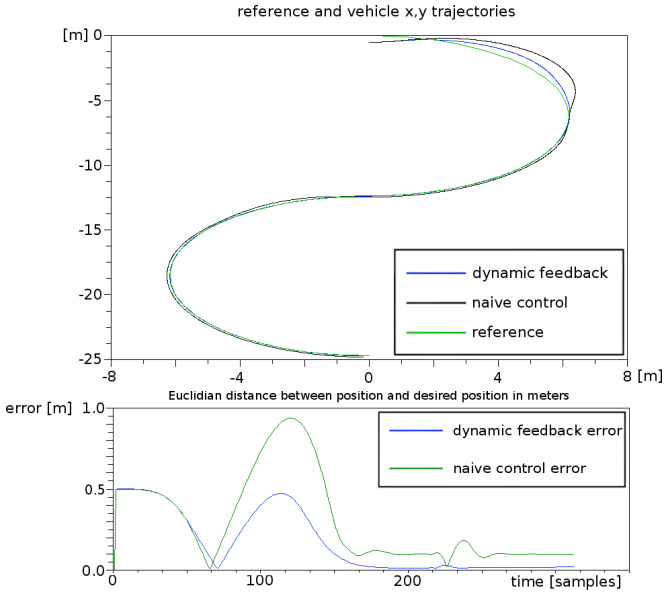


Figure 5.13: Comparison of the two controllers when following an "S" shaped trajectory with linearly increasing speed. Top shows the resulting paths, bottom shows tracking error in time

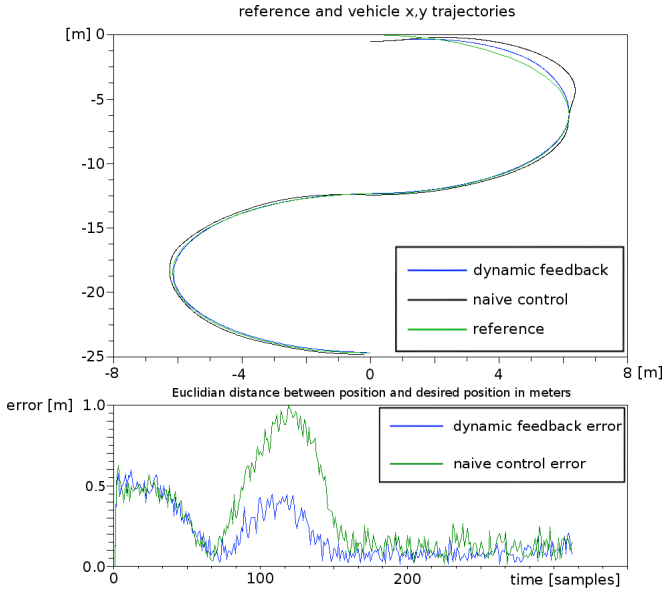


Figure 5.14: Comparison of the two controllers when following an "S" shaped trajectory with linearly increasing speed. Considering model and observation noise.

Perception The perception module can be validated by evaluating the quality of reconstruction of the world model based on a known sequence of measurements. This is discussed in section 4.4.2.

Route planning The implemented route planning is simple enough to be evaluated simply based on human expectations over the output.

Trajectory planning Being the main decision component, this module is fairly critical. All of its decisions are based on an instantaneous estimation of a world model with predictive capabilities (i.e., it includes moving objects). It can then be tested over different predefined scenarios where the input world model is fixed. Given a partial world view it should be verified that the trajectory planner makes a satisfactory choice.

The decisions based on a single partial world view are the one that define the behaviour of the robot when a sequence of measures is used to update online the world model. The portion of world that is observed is dependent of the previously planned trajectory. Previous plans affect future plans. Thus it is desirable to also verify the online behaviour of the vehicle using a simulator such as the one described in section 5.5. For this validation test the perception and control modules should be bypassed, using the simulator to provide a perfectly conservative partial world model and perfect trajectory tracking.

Control & Perception When validating the control on its own we assume that the vehicle state estimation is perfect, or respect a defined noise model. This has to be validated with the actual perception module state estimation. To do so, we execute perception and control modules online in the vehicle, but using a fixed predefined trajectory. The vehicle will move and we are able to check that the perceived control error is lower that a defined bound.

Any error software during this integration phase will lead to erratic movements of the vehicles. It is then desirable to first run this test in a simulated environment (see section5.5) before engaging in the physical vehicle.

Trajectory planning & Perception In order to check that the integration of this two modules is working as expected, the software can be executed in a manual mode. The vehicle will not follow the defined trajectory but the online effectiveness of perception and trajectory planning modules can be assessed.

In order to better emulate the relation between past trajectories and observed world, it is recommended to have the manual driver follow the computed trajectory (playing the role of the control module).

Route planning & Trajectory planning When executing the "Trajectory planning & Perception" validation, enabling the route planning allows to verify it corrects integration with the perception and trajectory modules. As the vehicle moves around the roads, the goal of the trajectory planning should move accordingly to the relative position to the desired destination.

Safety Once the previous steps have been executed, the integration of route planning, trajectory planning, perception and control will provide a vehicle moving autonomously. The key issue becomes then to know if this vehicle is moving safely or not.

Basic tests such as staying in the road, avoiding some obstacles and stopping in front of a road blockage provide an intuition of the correct behaviour. However safety cannot be verified by tests.

Just as software security is not guaranteed by hours of uncorrupted operation, robot safe behaviour is not guaranteed by hours of execution without harm. Safety is a property that is not guaranteed by providing showcases, but rather that by verifying that the design follows the guidelines of chapter 2.

One of the testable properties of a safe driverless vehicle is the fact that it slows down when approaching unobserved areas (in order to cope with the possible appearance of moving objects).

5.5 Simulator

In order to be able to test and debug our software without risking the vehicle we developed an in house simulator [216]. Its use is particularly critical when considering scenarios with multiple automated vehicles, as discussed in chapter 6.

The purpose of the simulator is to be able to test the code that will run embedded in the vehicle. Consequently, the simulator only simulates the vehicle physics and the sensors measurements. All the perception, planning, control and communication processing is done in real time using the code that runs on the vehicle.

From the software point of view, accessing a simulated vehicle or accessing a real world vehicle is transparent. When running in the vehicle the software access the sensors and actuators through multiple data buses (CAN, ArcNet, serial ports, USB, etc...), when running with the simulator, all the sensor data and actuators commands are exchanged with the simulator server through a TCP/IP connection over the network. The simulator server supports having multiple clients connected simultaneously and provides a 3D visualisation of the current situation.

The figure 5.15 illustrates the processing flow when two clients are connected to the simulator. Since all the computation is done in real time, having n clients running in practice means to have n machines running the driverless vehicle software. In practice we would be able to run the software in the vehicles but connected to the simulator

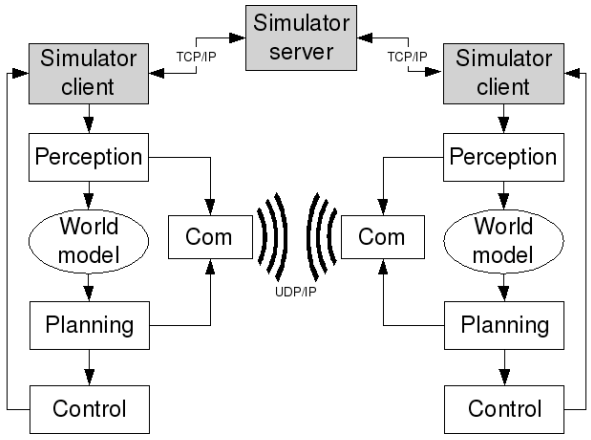


Figure 5.15: Two clients connected to the simulator. The grey boxes are components not used during real world execution. The "com" boxes indicate the communication modules. As depicted, the communication between clients it *not* done through the server, but using separated communication channels. For each connected vehicle the simulator takes as input the vehicle commands, and provides as output updated sensors measurements

server instead of the actual vehicle sensors and actuators, this allowed us to check up the on board computing power.

Current implementation of the simulator provides simulated odometry, GPS, compass and laser scanner sensors, and receives as input the same speed and steering commands that the real vehicle.

Figure 5.16 presents a screenshot of the 3D rendering provided for visualization purposes. Aside from the connected clients (the simulated vehicles), the simulator scenario can include static objects (buildings, trees) and objects moving through simple piecewise linear paths (pedestrians, dummy cars).

5.6 Experimental results

After following the steps pointed out in section 5.4 an integrated system was achieved. As mentioned the system correct behaviour cannot be garuanteed through showcases, however in the next sections we present some experimental results that provide an intuition of the system behaviour.

5.6.1 Simulated experiment

Setup We use the simulator to create a simple 90º intersection scenario, with walls all around the roads. The target of the driverless vehicle is to cross the road. Moving

Figure 5.16: Simulator example scenario

objects traverse the perpendicular road and the intersection area.

In order to make the example less trivial we also added two static obstacles on the driverless vehicle's road.

Results In figure 5.17 we present an example of perception and planning coupling. The figure presents a visualization of the built world model and the currently planned trajectory. Darker areas represent higher occupancy probability of static obstacles. Moving obstacles are represented by a circle, and the protruding line indicates the estimate direction. These results do not include the estimation of unobserved obstacles. The dark rectangle describes the current vehicle pose. The destination goal is at the top of the image. The executed trajectory is behind the vehicle and the planned partial trajectory is represented in front of it.

Analysis It can be seen that the world model is correctly estimated, including the past vehicle trajectory, the crossing area, the static obstacles and the moving ones.

The mismatch between the begining of the planned trajectory and the end of the estimated trajectory indicates that the tracking error is small enough.

The fact that the planned trajectory seems to collide with a moving obstacle indicates that it is correctly considering its predicted movement. Stopping before entering the unobserved area indicates a safe behaviour, as discussed in chapter 2.

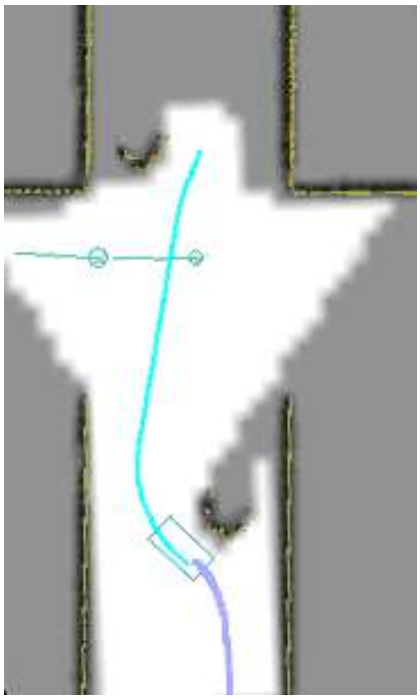


Figure 5.17: Example of safe planning in a perceived environment. See details in §5.6.1

5.6.2 Field experiments

Setup We present two small experiments realized in two different locations, but with a similar setup. The vehicle is asked to move to a nearby goal in an unknown environment. The surroundings of the vehicle are mainly locally planar, some static obstacles are present and a few pedestrians move around crossing themselves in front of the vehicle.

Given the cluttered nature of the environment, the speed of vehicle during the experiments is quite low, around 2 [m/s]. The actual speed is fixed by the planning algorithm that takes into account the "collision free stop" constraint.

Results In figure 5.18, we present the result of an early experiment. The top of the pictures are snapshots of the world model constructed online during the experiment. The bottom of the pictures show the corresponding scenes in the real world. The world model pictures nomenclature is the same as in section 5.6.1 with the addition of the desired position represented as a light green rectangle on the top right of the images.

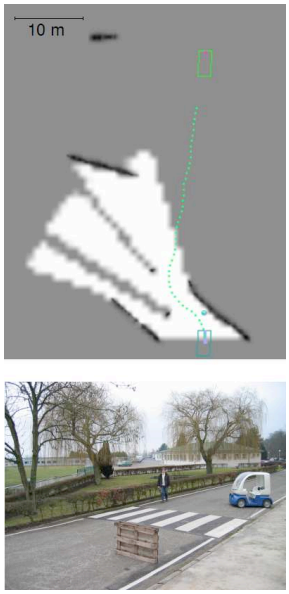
Figure 5.19 presents a sequence of snapshots from a similar experiment. Here the goal is located at the camera position.

Analysis These experiments show that the proposed architecture is functional and provides the expected behaviour. The vehicle is able to incrementally construct a map of approximately 50 square meters, define a safe partial trajectory towards the goal over some tens of meters and follow it, all in real time.

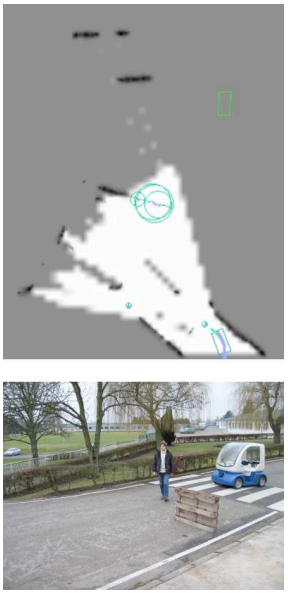
Observed execution times indicate the processing resources are roughly split in 40% for the laser based perception, 40% for planning and 20% for graphical output (control and tracking costs are negligible). Profiting the multi core capability of the on board processor the "perception + control + graphic output" loop runs at 20 [Hz], while the trajectory is computed in parallel and refreshed each 0.5 [s] (as explained in section 5.2.6). Notice that in practice the trajectories found provides a solution for a time horizon much larger than 500 [ms] (around 20 times larger in the presented scenario).

Since the robot is capable of distinguishing moving from static obstacles we use different strategies to avoid them. In the second field experiment, instead of turning to the left to avoid the pedestrian that appears, the planner will slow down until the pedestrian move away. On the other hand, the planner actively search for trajectories that will avoid the static obstacles.

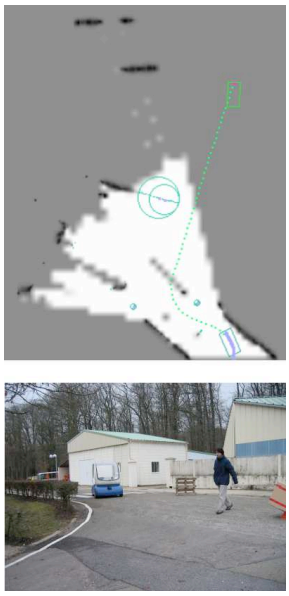
In the figure 5.18 we notice that a wall initially detected as a static obstacle progressively becomes a large set of moving objects. This is due to the presence of semi transparent objects (in this cased case a row of bushes), as discussed in section 4.4.4. As previously mentioned, the bushes are detected as moving objects but with zero speed. As such, they are only a minor inconvenience for the trajectory planner.



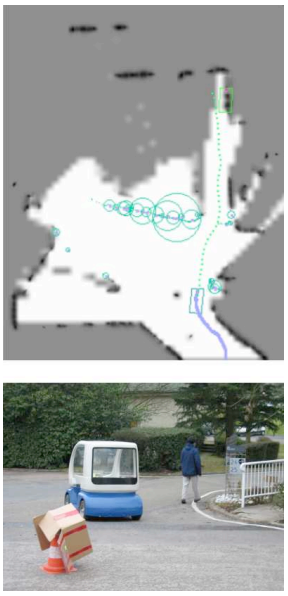
(a) start tracking the first trajectory



(b) braking in front a pedestrian



(c) avoiding obstacle and pedestrian



(d) continuing after avoiding the hedge and another pedestrian

Figure 5.18: Experiment results (using unsafe planning). See §5.6.2 for details

(a) starts exploring an unknown environment

(b) avoids known obstacles and adapts speed to posible obstacles

(c) stops in front of pedestrian

(d) continue moving

Figure 5.19: Experiment results (using safe planning). See §5.6.2 for details

5.7 Conclusion

The integration of the perception module of chapter 4 with the planning and control methods of section 5.3, have provided a first of its kind driverless vehicle system. While most of the previous works have focused the safety concerns on the trajectory planning, we have also discussed how this concerns affect the perception requirements and its integration with the planning algorithm.

The proposed efficiency aware algorithms are able to run all at once on a single computer and still respect the real time constraints. Up to the author knowledge this is the first vehicle capable of driving in urban environment taking into account the dynamics of the vehicle and the dynamic of the environment without the need of modifications of the environment or an initial hand driven visit. This is a distinctive characteristic with respect to previous works such as [217, 164, 11, 218].

We believe that the presented framework provides a generic approach for safe planning in dynamic environments. It is able to manage situations of arbitrary complexity while respecting all the constraints of the problem. The same framework can be applied to different vehicle models (cars, buses, trucks, etc...) and enhanced through more sophisticated (or application specific) methods.

In the next chapter we are going to point out the limitations of the proposed approach in a multiple vehicles' context and discuss how nearby vehicles can benefit from each other.

Chapter 6

Collaborative perception

> Tous pour un, un pour tous, c'est notre devise.
> Alexandre Dumas

In the previous chapter we presented a solution to allow a single driverless vehicle to perceive its environment and based on its observations and previous knowledge safely navigate from a point A to a point B in the city. Logically, in a real world deployment we would like to have many driverless vehicles filling the streets. As the presence of automated vehicles grows, encounters between them will become more frequent, thus some kind of interaction between the driverless vehicles will be established.

The driverless vehicles have the inherent capability of integrating the communication as part of their cognitive process. Automated cars have the possibility of transfering useful information between them, this provides a potentially important opportunity to behave "better than humans". This communication channel open various possibilities at each level of the system described in chapter 1.

Collaboration ▶ Perception

Integrating the data from diverse vehicles allows to precise the relative localization and to have a more complete representation of the near environment. The exchange of global maps or measures of surrounding areas enhance the consistency of the maps and the anticipatory capabilities of the vehicle. Integrating the data from a multitude of vehicles allows to have more vali.e.imates of the four dimensional representation of the city.

Collaboration ▶ Trajectory planning

Since the collaboration between driverless vehicles enhance the quality of their world model, it allows to choose better trajectories. Near vehicles can negotiate the choice

of the trajectories in order to realise coordinated manoeuvres, in order to improve efficiency (avoid unnecessary stops) and efficacy (avoid deadlocks).

The safety aspect was previously discussed in section 2.6.4.

Collaboration ► Control

In chapter 5 we presented a system where trajectory planning defines the reference state for the control. However other strategies can be envisaged, where the desired state is a function of other cars states. A typical case is the car following scenario (so called "platooning"). In such a case instead of only estimating the state of the front vehicle from observations, it is also possible to receive information directly from the front vehicle. More information allows to enhance the state observer and thus enhance the control quality [219].

There is a specific community dedicated to the platooning problem. They use lower level, application specific, solutions. These techniques are considered out of the scope of this dissertation.

Collaboration ► Route planning

Being able to communicate between them, the collection of automated vehicles circulating in the street can be considered as a sensor network. The information of the routes status can be collected to choice the best route. Also, exchanging the planned routes traffic jams can be anticipated and avoided.

This ideas are beginning to be deployed in commercial systems [220].

Collaboration ► Service layer

At the service layer the collaboration between the vehicles open the possibility to distribute the load of the infrastructure and to obtain more responsive and reliable distributed systems.

Supposing the availability of vehicle to vehicle (V2V) and vehicle to infrastructure communications (V2I), it is possible to ideate distributed mechanisms for data collection, traffic jam avoidance, users requests management and anticipation, for instance.

Some work has been made in the recent years to explore these possibilities, never less the topic of multi robot systems continues being a wide area to explore, both on the theoretical and the implementation aspects.

On this dissertation we will focus on the use of data exchange between near vehicles to enhance the perception. Section 6.1 will argue that the core problem behind collaborative perception is a relative positioning problem. In §6.2 we will propose a solution to this particular problem. Section 6.3 presents some experimental results and section 6.4 provides the conclusion and suggest future works.

6.1 Problem definition

Most interactions require communication between the driverless vehicles. During this communication one driverless vehicle will refer to elements in the environment that other driverless vehicles need to recognize in order to take decisions. Such elements can be for instance, objects of the scene, a specific area, one of the vehicles, etc... The problem of data association between the data that one vehicle receives from others and the data that it measures is usually avoided by using special markers or supposing that each vehicle has a high precision localization in a shared reference frame.

One of the advantages of using a probabilistic formulation of the perception problem, such as described in chapters 3 and 4, is that fusing multiple sources of information can be done in a methodical manner.

It has been shown [221, 222] that receiving measures from other robots is, essentially, just like integrating a past measure from our own robot. Measure are in the past due to the communication delays. Given the transformation between the position of the remote vehicle reference frame to the local vehicle reference frame, the fusion of the received information is possible.

The particular problem arises from the fact that we do not want to use special markers on the environment or over the vehicles themselves. Installing special markers on the environment would impose an important cost on the deployment of such robotic systems. Requiring special markers over the vehicles would limit the possibility of having heterogeneous driverless vehicles interacting.

We are also supposing that the vehicles do not have a high precision global localization mechanism. Global localization are usually maps based or satellite based. Using precise maps in an urban environment is usually a brittle approach since by nature it is a changing scenario. The best invariants candidates for localization are the visible parts of the buildings. Constructing such a map still require a considerable amount of effort, and the precision of such a method is not guaranteed to be sub-metric in all areas [183]. Satellite based positioning systems is the most commonly method used for global positioning, unfortunately inside the urban canyon the coverage can be quite low [52] and thus the positioning error will be large (5~10 meters).

Since the global positioning error is high, the relative positioning error obtained when exchanging the global positions is also high. One meter of global positioning error in outdoor environment is usually considered a high precision localization. However the relative positioning error between two vehicles will be of 2 meters, which is the difference between having a driverless car passing by the side or having a front crash.

While global positioning will probably have gross error, relative measures between the vehicle and its surrounding can be much more precise. Typical laser scanners will provide an error of ~10 centimeters at distances of tens of meters. Vision methods will not provide such accurate measures (since the error grows quadratically with the distance) but they can provide accurate bearing measures and sub-metric distance measures on a short range.

When global positioning has high uncertainty, and relative measures have high precision, then the relative positioning problems becomes a data association problem, between the high uncertainty candidates received via the communication channel and the higher precision candidates observed using the driverless vehicle sensors.

6.1.1 What exists?

Previous works on teams of mobile robots have addressed the problem of relative positioning in different ways. We provide here a brief overview of some representative approaches.

The RoboCup championship [223] requires having multiple robots achieving thigh coordination in a dynamic environment. One some categories game rules allow to have an external global view of the scene, rendering the relative positioning issue trivial. More advanced categories require full autonomy of the robots (no third party), while robot to robot communication is allowed. Some teams design coordination strategies able to work with high positioning uncertainty. A more sophisticated approach consists on exchanging the robot measures to enhance the positioning. Goehring [224] proposes to use the observations of known element in the scene in order to allow the tracking of the game ball and of the players robots. This positioning approach is based on the previous knowledge of the game field and the identification of the specific tags on it.

Another well known application where robots relative positioning is needed is the multi-robot mapping task. The usual solution consists on installing externally visible identification tags over the robots, as done in [221] (see figure 6.1). Other works suppose that the robots evolve on the same environment before cooperating [222], thus one robot will localize in the map constructed by a second one, before interacting with it. In order to obtain a non ambiguous localization the robot needs a considerable displacement. Also, we need to verify that the first robot is actually evolving the area mapped by the second one and not in a similar but distant area. More recently, in the context of the Centibots project, another solution was proposed. The robots are able to detect in the environment the presence of other robots. When a robot detects a possible candidate, in order to resolve the data association it requires the remote robot to realize a specific action (e.g. “go to meeting point”). If the observed robot responds to the request as expected, then the data association is resolved, if not, then the candidate is discarded [225].

A third example of robots localization for tasks realization is the coordination between ground and air robots being deployed in urban environments [226]. Here the global view of the air robot and the relative measures of the ground robots is used to enhance the localization of each element and thus to enhance the chances of achieving the exploration task.

Figure 6.1: Example of tagged robots used in collaborative indoor mapping. From [221]

6.1.2 What does not exist?

When two driverless vehicles are near, there seems to be a benefit in being able to exchange data, to enhance the range of perception, to have a better coordination [227] or to enhance the global localization [228], for instance. All of these activities require a precise enough relative localization in order to extract useful information from the exchanged data.

As seen in the previous sections some solutions already exists, however none of them can be considered desirable for our application, since they suppose the use of special tags (on the vehicles or the environment) or the presence of a third party observer (top view of the scene). The idea of using a shared map [222] is not applicable when two driverless vehicles encounter front to front (opposite direction lanes, arrival at an intersection) since their maps cover different areas, however a similar idea can be applied for front to back encounters (the same direction lanes), as discussed in §6.2.2.

Summarizing the proposed formulation, data fusion between data from remote vehicles is a problem of relative positioning. Exchanged data is referenced to a global reference frame with high uncertainty, relative measurement of surrounding cars provides low uncertainty relative positions. Then the problem becomes a data association problem: who is sending which data?

A secondary problem related to collaborative perception is considering a limited bandwidth between the vehicle. Then it is necessary to decide which data to send, in which order, which format?

For the purpose of this work we will suppose that vehicle to vehicle wireless link is available and that the point to point channel has a latency inferior to 100 [ms] and a bandwidth superior to 1 [Mbits/s]. This values are similar to what can be expected in real world experiments.

6.2 Proposed solution

As explained in the previous section, the main problem to solve is the data association between observed vehicles and communicating vehicles. In order to do this we propose a method based on the analysis of the coherence between local observations (from the vehicle) and remote observations (from other vehicles and communicated via a wireless link).

Each vehicle has a world model describing it surroundings. In this world model the objects positions are measured with respect to the vehicle, and the vehicle is able to estimate its position with respect to a global reference frame, shared between the vehicles (e.g. a Global Navigation Satellite System).

The driverless vehicles will transmit information about its view of the world. Then, each vehicle will do individually its decision about the data association problem by verifying the coherence between the observed vehicles, the received data and the possibly present but non observed vehicles.

6.2.1 Front to front encounter

Let W_i be the world model of vehicle i. The world model is composed by a description of the static obstacles surrounding the vehicle, the moving obstacles around the vehicle and a geometric description of the vehicle itself (i.e. the output of a SLAMMOT process, see §3.7 and §4.2). Additionally we are supposing that the vehicle is capable of classifying the moving obstacles as vehicles or not. Let r_j^i be the data received by the vehicle i from the vehicle with identifier j and R_i the set of data received by the vehicle i. $Ov_i = \{ov_1^i, \ldots, ov_n^i\}$ is the set of vehicles observed from vehicle i (and thus part of W_i).

Data association methods in the context of multi-objects tracking or in features based SLAM usually try to estimate the maximum a posteriori probable association. In our application the results of the data association will be used for decision making, thus we do not search for the most probable explanation but for high certainty associations.

For each pair $(ov_k^i,\ r_j^i)$ we are going to estimate

$$\Omega_{kj}^i = P(j \text{ is } ov_k^i |\ W_i,\ R_i)$$

And we are also going to estimate

$$\Omega_{0j}^i = P(j\ \notin Ov_i |\ W_i,\ R_i)$$

Then the association between the observed vehicle ov_k^i and the received data r_j^i is going to be done only if

$$\frac{\Omega_{kj}^i}{\sum_{l=0}^{n} \Omega_{lj}^i} > p_t \tag{6.1}$$

where p_t is a threshold value associated to the confidence level related to the safety requirements of the application. The instantaneous data association result can be modeled as a Markov process and then extended to the time dimension, thus allowing more robust results.

The core idea to estimate Ω^i_{kj} is illustrated in figure 6.2. When we use only the vehicle's observations and the global position received from the remote vehicle j there is a high uncertainty in the data association (figure 6.2(a)). To reduce this uncertainty we propose to use the vehicle's j measures Ov_j in order to cross-validate the hypothesis Ω^i_{kj}. The figure 6.2(a) shows a simplified one dimensional scenario. The red (left) vehicle has one mobile object observation and received one remote vehicle position with the identifier j. As expected the precision of the local observation is higher than the received remote position (once put on the red vehicle reference frame). Does the local measure corresponds to the remote vehicle? Using only the transmitted vehicle position both situations (b) and (c) of figure 6.2 seem plausible for the blue (right) vehicle. Since both vehicles are facing face to face, and assuming that they are in the range of mutual detection, if vehicle k detects the vehicle j then necessarily the converse is true. Using Ov_j to search a matching measure allows to disambiguate the two situations presented in figure 6.2 and thus to provide a more accurate estimation of Ω^i_{kj}. Please notice that this estimation takes in consideration both the observed obstacles and the observed free space to verify the coherence between both vehicles measures.

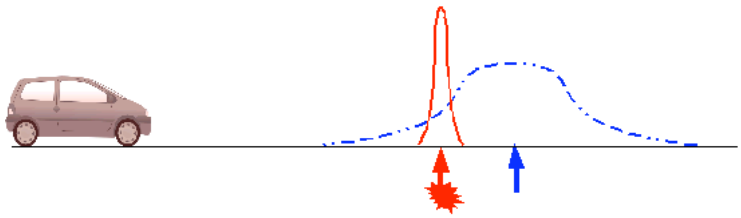
The extension of this idea to the 2D case is simple. The estimation of the relative orientation between the vehicles becomes a relevant parameter since it will impact considerably the uncertainty of the relative measures matching. Fortunately it can be shown that the orientation of the vehicle can be estimated with good precision even when the global position is uncertain[229]. Also cheap sensors are available to directly measure the local magnetic field of the earth, providing a direct measure of the vehicle orientation a global reference frame [230]. In the 2D case each vehicle i needs to transmit its current position on the global reference frame $x_{o\rightarrow i}$, a 2D description of its geometry g_i (in our case, a rectangle) and the set of observed vehicles around it Ov_i, i.e., $r_i = (x_{o\rightarrow i},\ g_i,\ Ov_i)$. The matching between an observed vehicle ov^i and the geometry of a received vehicle candidate g_j is done in a similar manner than [173], where observations can be only a consistent partial observation of the true geometry g_j.

The value of Ω^i_{0j} is estimated using the information about the free space, the occupied space and unobserved space. Defining the function over the space

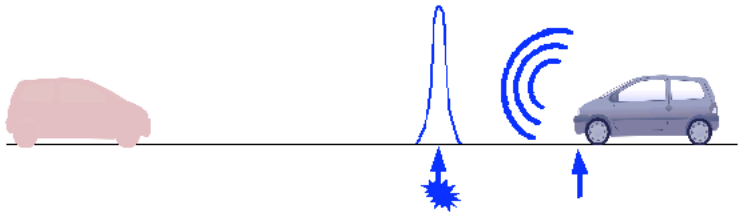
$$unobserved(y) = \begin{cases} 1 & if \quad position\ y\ was\ not\ observed \\ 0 & if \quad position\ y\ was\ observed \end{cases}$$

then we can write.

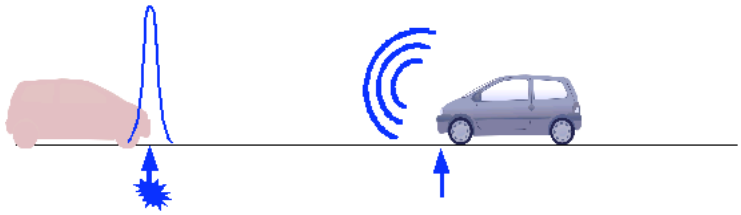
$$\Omega^i_{0j} = \int P(x_{i\rightarrow j} = y) \cdot unobserved(y)\ dy$$



(a) Red car perspective



(b) Blue car perspective, case 1



(c) Blue car perspective, case 2

Figure 6.2: Illustrating disambiguation by cross-validation. Continuous line Gaussian indicate a direct measure, dot line Gaussian indicate a received position, underline arrows indicate the mean value of the Gaussian. All figures drawn in the red car reference frame. Case 1 and 2 are plausible under information of figure (a)

In practice Ω^i_{0j} will be estimated using a discrete approximation of $unobserved(y)$ and doing an integral only in the surroundings of $x_{i\to j}$ (the position of the vehicle j in the reference frame of vehicle i, estimated using the global positioning information).

Using the cross validation approach two vehicles approaching each other will be able to enhance their relative positioning using their relative measures $ov^i$and ov^j(which can be fused) as soon as they are in the mutual detection range and the available information provides a non ambiguous data association. We expect this solution to be useful for vehicles interacting in a double direction lane or arriving at an intersection.

This method does not require special tags, installations or third party observers. The capabilities of detecting and classifying moving obstacles, as well as estimating the observed and unobserved areas, are already required for the basic driverless behaviour [12]. The only additional requirements are the repeated transmission of the data r_i and the ability to relate the observations ov_k with a geometric description g_j. Depending on the available sensors this description can be the shape of the vehicle, some kind of color description, a radar signature or any informative element that could help to narrow the range of possibilities.

It is worth noticing that the mutual detection range requirement is flawed for one typical case: when two cars are moving in the same direction. It is very likely that the driverless vehicle has only long range sensors on its front, the lack of back measures limits the usage of the proposed cross validation method. In the next section we will explain how to deal with this common case.

6.2.2 Front to back encounter

When one car follows the another one, we cannot rely on the mutual observation idea. Instead we propose to match the sequence of observations of static obstacles in order to do the data association between the observed vehicles Ov_i and the received data R_i.

Let denote $ov^i_{k,t}$ a vehicle observation from a vehicle i at current time t. Tracking the moving vehicle in time will provide a track $ov^i_{k,t_a\ldots t}$, i.e., a sequence of observations. This time we search to estimate

$$\Omega^i_{kj} = P(j \text{ is } ov^i_{k,t} |\, W_i,\, R_{i,t_a\ldots t_b})$$

To do so we a going to extend the content of the transmitted data $r_{j,t}$ to include information about the static obstacles surrounding the vehicle, that we will denote as $so_{j,t}$ (part of W_j). Thus we redefine $r_{j,t}$ as $r_{j,t} = (x_{o\to j,t},\, g_j,\, Ov_{j,t},\, so_{j,t})$.

Using its ego-motion estimate the vehicle i is able to retrieve for any time $t_c \in [t_a, t]$ an estimate of the tracked vehicle position ov^i_{k,t_c} with respect to its current position $x_{o\to i,t}$. Conversely, it is able to retrieve for the current position the nearest observed position $ov^i_{k,t_d} \in ov^i_{k,t_a\ldots t}$. Estimating Ω^i_{kj} consists on comparing the current observation of the surrounding static obstacles $so_{i,t}$with the one provided by r_{j,t_d} (received data with the timestamp of ov^i_{k,t_d}). Using ov^i_{k,t_d} the measure so_{j,t_d} can be put in the same reference

frame than $so_{i,t}$ with a precision as good as the sum of the relative measures error and the ego-motion estimation error.

In order to validate the data association we expect to have a sequence of measures passing the criterion of equation 6.1. Comparing the sequence of static measures taken by two vehicles at two different times is essentially equivalent to compare two maps built by the different vehicles [222]. There is however one important condition that the sequence of measures used to do the data association needs to meet. We need to ensure that the sequence is not repetitive. More precisely, we need it to be not repetitive over the length of the $x_{i\to j}$ position uncertainty (composition of $x_{o\to i}$ with $x_{o\to j}$) mapped over the path of the tracked candidate $ov^{i}_{k,t_a\ldots t}$. When the sequence respects this condition we can ensure with high certainty that the data association is correct. This means however that if no discriminative feature appears in the environment or if the vehicles have no view over static elements of the environment, then data association is delayed even if the vehicles have been following each other for a considerable amount of time.

As in §6.2.1 we have proposed a new method for data association between following vehicles that does not require special tags or third party observers. The only special needs are the capabilities of detecting static elements of the environment (that a following car is likely to see again) and matching two observations. These are already the standard requirements for SLAM, that driverless vehicles need to do to achieve a correct planning and ego-motion estimation.

6.3 Results

6.3.1 Platform

The solution described in §6.2.1 was implemented in C++ and integrated into the system described in chapter 5 (code able to run both on the vehicles and in the simulator).

The module "communication" of the figure 5.2 broadcasts periodically the information collected by the world model and receives it. Each time new data is received the equation 6.1 is evaluated and if the data association confidence is high enough, the received data will be integrated into the world model.

The communication is ensured by a WiFi 802.11b/g transceiver installed in the vehicle (see figure 5.1). Data is exchanged using a binary serialization sent into multicast UDP IPv6 packets.

The data exchanged between the vehicles includes the current GPS position and time, the observed free space, the detected moving obstacles and the planned trajectory.

6.3.2 Simulations

The purpose of the initial simulations was to verify the interest of having an enhanced relative positioning when dealing with intersections crossing. We use the simulator described in section 5.5.

We suppose that two vehicles arriving at an intersection are able to communicate their trajectories (states in time). These trajectories are computed repeatedly on real time, considering the current perceived world model and the trajectories received from the other vehicles.

When exchanging trajectories without enhanced positioning one of the vehicle is forced to stop at a considerable distance from the path of the second one. Without communication each vehicle sees the other one as a free moving car. Figure 6.3(a) shows a snapshot of the world model of one the vehicles in such a situation. The blue circles indicate the conservative prediction of the observed car over a finite time horizon and the green line in front of the vehicle its planned trajectory. When no communication is used each car will try to avoid the other one, since they have a symmetric behaviour the situation is not resolved correctly, finishing with both vehicles inter-blocked. When using communication with an enhanced relative positioning each vehicle has a precise information of the others one plan, only the necessary avoidance is done and the situation is correctly resolved. Figure 6.3(b) shows clearly how the vehicle is aware that the second one is planning to avoid it and thus has only a slightly modified its plan.

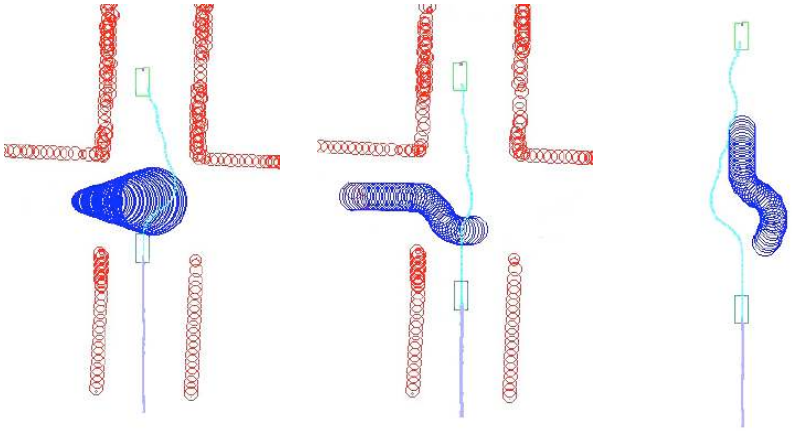
A more obvious situation where enhanced relative positioning is desirable is for front to front encounters between vehicles. Seen as a free moving obstacle, the front vehicle could potentially make a turn at any moment, thus forcing the driverless vehicle to slow down considerably during the encounter. Using communication without enhanced positioning would block both vehicles since the positioning uncertainty will be probably be as large as the width of the road. Figure 6.3(c) shows how the enhanced positioning allows to decide which vehicle passes using which side.

The simulations show that having an enhanced positioning while exchanging vehicles trajectories allows to solve a number of interesting situation without having to impose ad hoc, case per case management logic. The same approach can be extended for more complex situations by adding a generic priorities management system [227].

6.4 Conclusions and perspectives

It is clear that driverless vehicles in urban environment will have frequent interactions with other mobile robots. We have shown that the core issue to allow collaboration with other driverless vehicles is the relative localization between vehicles. This problem is usually trivial when doing small scale indoor robotics, but becomes a non trivial issue in the scenario of urban driverless vehicles. This chapter attempted to point out the related difficulties and proposes a solution under realistic assumptions. The proposed method will allow tight interactions between driverless vehicles without requiring high precision absolute positioning, modifications of the infrastructure or special markings on the vehicles.

An interesting question left to explore is the interest of propagating the data association results between vehicles in order to make use of data coming from vehicles not directly observed. How to do solve this enlarged problem is non trivial. The benefits of solving



(a) Crossing without communication

(b) Crossing with communication

(c) Front encounter with communication

Figure 6.3: Simulation results, world view from one of the vehicles. See §6.3.2 for details. Red circles indicate static obstacles, blue circles indicate the evolution of moving obstacles in time. The dark rectangle indicate the current vehicle position, the green rectangle the desired position of the vehicle. The vehicle is moving from the bottom of the image to the top. The line behind the vehicle is the realized path, the green line in front of it, is the planned trajectory

this problem are not self evident since the time delay affects considerably the relative positioning uncertainty between moving vehicles, it is possible that the increase information obtained by the additional measurements is lots due to increased time delay caused by the multi-hop messages passing. This is question is worth to be explored in more detail.

Chapter 7

Conclusion

> I came to the conclusion that I am not a fiction writer.
> Tim LaHaye

7.1 Present

All along this dissertation we have provided a discussion around perception for ground mobile robots. We have focused on issues such as uncertainty management, tractability, safety requirements, and explored the notion of collaborative perception.

Through these discussions we attempted to provide arguments and experimental evidence to show that "Perception for driverless vehicles in urban environments without specific infrastructure can be done".

In particular we claim the following contributions:

- Chapter 2 presents a novel analysis on safety for driverless vehicles,
- Chapter 4 presents a novel perception method that satisfy the described requirements,
- Chapter 5 describes a practical full scale demonstration of the feasibility of the suggested solution for driverless vehicles,
- Finally, chapter 6 discuss the infant concept of "collaborative perception", points out the insufficiency of existing solutions and proposes a new method.

We believe that this are promising results going one step forwards the deployment of driverless vehicles inside tomorrow's cities.

7.2 Future

The field of robotics has become very active in the latter years. Much of this development has been driven by the advances in electronics, allowing new kind of sensors, lower cost on embedded systems and a remarkable increase on the computing power available. The increase in computing power allow to process more data at higher rates, and to formulate some problem as optimization problems previously considered intractable.

The next years of research on perception should provide more maturity on the design of world models and new methods for their efficient estimation. The latest trends focuses on having larger models taking a more explicit consideration of the generative nature of the measurements (e.g. processing images as 3D to 2D projections instead of analysing the 2D space) and considering more interaction between different cues (e.g. [231, 188]).

The advances in the field of machine learning will also provide important tools for the developments in robotic perception, with expected technical and theoretical progresses on classification, (online) learning, time signal processing and large scale optimization. In the recent years this community has raised a noticeable interest towards robotics related problems and applications.

At the end most of the machine learning problems, and the probabilistic formulations of the perception problem end being instances of optimization problems. Both of them are likely to obtain a net benefit on the recent parallelization trend of the silicon processor.

In the next years we can expect that advances in electronics will allow the popularization of 3D sensors such as the one described in the section 3.3.5. Such sensors will switch the nature of the problem (from 2D to 3D processing) and enable new applications of robotic systems.

7.2.1 Vision

It is worth noticing that even if solid state dense 3D sensors exist, as they use the same underlying technology than passive vision sensors (currently, CMOS), the latter will always be cheaper because of the wider market, and the absence of the active emitter. In some applications active sensors are not desired or effective (e.g. subject sensible to IR emissions, required power beyond eyes safety).

Also, there is the question of feasibility. It is known that human are able to interpret and act in the world using their stereo vision as the main sensor. Since it is known possible, how to reproduce such capabilities in a robot is a relevant question. In the context of this dissertation, the question would be: how to have a robotic driving system solely based on vision is the question?

Currently computing capabilities certainly prohibit an embedded solution respecting the real time constraint. But as long as the solving algorithm is parallelizable, this is a minor issue considering the current trend on computer power increase.

State of the art works on vision show that it possible to learn to classify the main elements of a scene [188], estimate accurately the movement of the vehicle [80, 232], build features based map [233], build occupancy maps [84], detect obstacles and free areas [75, 232, 231]. With such advances it becomes increasingly clear that an all vision perception system for driverless vehicles is around the corner.

An interesting related question is the need or not of a "learning before driving" stage. It could be envisaged that the vehicle is able to judge which parts of the near scene is traversable or not using a 3D geometry reconstruction approach, and then extrapolate the local appearance to far areas of the scene where an accurate 3D geometry reconstruction is not yet possible. This could be integrated into an online learning mechanism allowing a vehicle to drive without requiring to harvest large amounts of a priori learning examples. This approach would enable the vehicle to confront new situations correctly.

7.2.2 Open questions

Through this dissertation we covered a large area of research related to ground mobile robots, and proposed solutions to the specific application of driverless vehicles in urban environment. Yet some interesting questions are left open.

For instance, as discussed in chapter 4, it is necessary to assume the appearance of moving obstacles in the unobserved areas. The probability of appearance depends on space and time. Is there a better way than just assume worst case scenarios? Most solution would involve either learning the relations between the environment and the moving obstacles or creating a space time city map for moving obstacles (this is related to the notions of "city 4D maps" and "4D GIS"). How to learn such relations? How to create and use such maps?

In chapter 6 we discussed how the perception of one vehicle could benefit from data from another one. This also applies to the planning module. Distributed multi robot planning is still a hard problem to be solved. Which would be good a solution for this problem? How to provide deadlock free plans given the perceived world? How planning for multiple vehicles can include or affect the collaborative perception?

Despite the lengthy amount of existing research presented in chapter 3, the basic questions for robot perception (measurements, model, transformation) remain open. New ideas will emerge on world models and measurements to model transforms. At least we can expect a maturation on the equivalences and relations between the different approaches (e.g. SLAM and Structure from Motion) and tighter relations with other research communities (vision, machine learning).

An evolution from generic to specific will try to define which is the best design that will allow a real world deployment of a driverless vehicles system of the kind described in chapter 1.

An evolution from specific to generic will try to define how a robot can learn and use generic physics principles from observations. "If robots were physicist" they would

be able to observe the world, generate models, correct them to explain the new observations, and then use these models to predict the world. Understanding how to build such a system would probably provide answers for most of the robots applications. Being the support for decision making, robot perception is a key element for the diversification of robot's applications domains. The field of machine learning for robotic perception is still in its infancy.

The big picture of the work discussed all along this dissertation is the notion of "model-based computing". Instead of focusing the effort on defining the input output behaviour of the robots, the focus is put on building models and using such models. This idea is related to artificial intelligence ideas such as the memory-prediction theory of J. Hawkins [234] and span over a large domain of information processing applications.

Hopefully the ideas presented here will motivate the next generation of students to explore these questions.

Bibliography

[1] Gapminder, "Gapminder website," 2007. [Online]. Available: http://www.gapminder.org/

[2] WorldBank, "World bank development education program life expectancy website," 2007. [Online]. Available: http://www.worldbank.org/depweb/english/modules/social/life/index.html

[3] NSC, "What are the odds of dying?" 2007. [Online]. Available: http://www.nsc.org/lrs/statinfo/odds.htm

[4] "European road safety observatory website," 2007. [Online]. Available: http://www.erso.eu/

[5] W. H. Organization, "World report on road traffic injury prevention," 2004. [Online]. Available: http://www.who.int/violence_injury_prevention/publications/road_traffic/world_report/en/index.html

[6] M. Lowson, "Energy use and sustainability of transport systems," Advanced Transport Group, University of Bristol, Tech. Rep., 2004, prepared under the Cybercar Contract, Task 3.4 Energy Management. [Online]. Available: www.cybercars.org

[7] UnitedNations, "World population and urbanization prospects website," 2007. [Online]. Available: http://esa.un.org/unpp

[8] K. Rumar, "Transport safety visions, targets and strategies; beyond 2000," 1st European Transport Safety Lecture. European Transport Safety Council, Brussels, Tech. Rep., 1999. [Online]. Available: http://www.etsc.be/documents/etsl1.pdf

[9] I. Paromtchik and C. Laugier, "Autonomous parallel parking of a nonholonomic vehicle," 1996. [Online]. Available: http://citeseer.ist.psu.edu/article/paromtchik97autonomous.html

[10] C. Pradalier, J. Hermosillo, C. Koike, C. Braillon, P. Bessière, and C. Laugier, "Safe and autonomous navigation for a car-like robot among pedestrian," in *IARP Int. Workshop on Service, Assistive and Personal Robots*, Madrid (ES),

October 2003. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2003/PHKBBL03

[11] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the darpa grand challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, September 2006. [Online]. Available: http://robots.stanford.edu/papers/thrun.stanley05.html

[12] R. Benenson, S. Petti, M. Parent, and T. Fraichard, "Integrating perception and planning for autonomous navigation of urban vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006. [Online]. Available: http://hal.inria.fr/inria-00086286

[13] E. Royer, M. Lhuillier, M. Dhome, and J. Lavest, "Monocular vision for mobile robot localization and autonomous navigation," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 237–260, 2007. [Online]. Available: http://www.lasmea.univ-bpclermont.fr/Personnel/Maxime.Lhuillier

[14] A. Awasthi, "Développement d'un système de routage hiérarchique pour les réseaux urbains," Ph.D. dissertation, UNIVERSITE DE METZ, 2004. [Online]. Available: http://tel.ccsd.cnrs.fr/documents/archives0/00/00/77/51/index_fr.html

[15] S. Petti, "Safe navigation within dynamic environments: a partial motion planning approach," Ph.D. dissertation, Ecole des Mines de Paris, 2007.

[16] C.-C. Wang, "Chieh-chih (bob) wang homepage," 2007. [Online]. Available: http://www.csie.ntu.edu.tw/~bobwang/

[17] C. Wang, "Simultaneous localization, mapping and moving object tracking," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 2004. [Online]. Available: http://www.acfr.usyd.edu.au/people/academic/bobwang/Papers/wang_thesis_04.html

[18] J.-P. Laumond, *Robot Motion Planning and Control*, ser. Lectures Notes in Control and Information Sciences, J.-P. Laumond, Ed. Springer, 1998, vol. 229. [Online]. Available: http://www.laas.fr/~jpl/book.html

[19] T. Fraichard, "A short paper about motion safety," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007. [Online]. Available: http://hal.inria.fr/inria-00134467/

[20] T. Fraichard and H. Asama, "Inevitable collision states - a step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2004/FA04

[21] K. E. Bekris and L. E. Kavraki, "Greedy but safe replanning under kinodynamic constraints," in *Proceedings of the Intl. Conf. on Robotics and Automation*. Rome, Italy: IEEE press, April 2007, pp. 704–710. [Online]. Available: http://www.kavrakilab.org/node/401

[22] R. Parthasarathi and T. Fraichard, "An inevitable collision state-checker for a car-like vehicle," in *IEEE Int. Conf. on Robotics and Automation*, 2007. [Online]. Available: http://hal.inria.fr/inria-00134471/en/

[23] K. E. Bekris, K. I. Tsianos, and L. E. Kavraki, "A distributed protocol for safe real-time planning of communicating vehicles with second-order dynamics," in *Proceedings of the First International Conference on Robot Communication and Coordination (ROBOCOMM 07)*, Athens, Greece, Oct. 15-17 2007. [Online]. Available: http://www.kavrakilab.org/node/399

[24] S. Kolski, K. Macek, L. Spinello, and R. Siegwart, "Secure autonomous driving in dynamic environments: From object detection to safe driving." in *Proceedings of the Workshop on Safe Navigation in Open and Dynamic Environments: Applications to Autonomous Vehicles at the IEEE International Conference on Robotics and Systems*, San Diego, USA, 2007. [Online]. Available: http://teamster.usc.edu/~moradi/iros07/iv_iros07.htm

[25] R. Philippsen, B. Jensen, and R. Siegwart, "Toward online probabilistic path replanning in dynamic environments," in *Proceedings of the International Conference on Intelligent Robots and Systems*, 2006. [Online]. Available: http://asl.epfl.ch/index.html?content=member.php\&SCIPER=103090

[26] J. van den Berg, D. Ferguson, and J. Kuffner, "Anytime path planning and replanning in dynamic environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2006, pp. 2366 – 2371. [Online]. Available: http://www.ri.cmu.edu/pubs/pub_5305.html

[27] S. Petti and T. 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, Spain, September 2005. [Online]. Available: http://hal.inria.fr/inria-00182043

[28] Petti and Fraichard, "Safe motion planning in dynamic environments," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Edmonton, AB, Canada, August 2005. [Online]. Available: http://hal.inria.fr/inria-00182046

[29] C. Urmson, "Driving beyond stopping distance constraints," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, October 2006, pp. 1189–1194.

[30] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid," in *Proceedings of IEEE International Conference on Robotics and Automation, ICRA*, Rome, Italy, April 2007. [Online]. Available: http://hal.inria.fr/inria-00211845/en/

[31] A. D. Vasquez, T. Fraichard, O. Aycard, and C. Laugier, "Intentional motion on-line learning and prediction," in *Proc. of the Int. Conf. on Field and Service Robotics*, Port Douglas (AU), July 2005. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2005/VFAL05

[32] J. van den Berg and M. Overmars, "Kinodynamic motion planning on roadmaps in dynamic environments," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems - IROS*, 2007, pp. 4253 – 4258. [Online]. Available: http://www.cs.unc.edu/~berg/

[33] L. Bouraoui, S. Petti, A. Laouiti, T. Fraichard, and M. Parent, "Cybercar cooperation for safe intersections," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2006. [Online]. Available: http://hal.inria.fr/inria-00182006/

[34] C. Thorpe, O. Clatz, D. Duggins, J. Gowdy, R. MacLachlan, J. R. Miller, C. Mertz, M. Siegel, C.-C. Wang, and T. Yata, "Dependable perception for robots," in *Proceedings of International Advanced Robotics Programme IEEE*. Seoul, Korea: Robotics and Automation Society, May 2001. [Online]. Available: http://www.ri.cmu.edu/pubs/pub_3724.html

[35] C. Thorpe, J. D. Carlson, D. Duggins, J. Gowdy, R. MacLachlan, C. Mertz, A. Suppe, and C.-C. Wang, "Safe robot driving in cluttered environments," in *Proceedings of the 11th International Symposium of Robotics Research*, October 2003. [Online]. Available: http://www.ri.cmu.edu/pubs/pub_4471.html

[36] S. Robertson, "Motion safety for an autonomous vehicle race in an urban environment," in *Proceedings of the 2006 Australasian Conference on Robotics & Automation*, December 2006. [Online]. Available: http://www.araa.asn.au/acra/acra2006/papers/paper_5_47.pdf

[37] C. Urmson, "Navigation regimes for off-road autonomy," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2005. [Online]. Available: http://www.ri.cmu.edu/pubs/pub_5096.html

[38] F. Biral, M. Da Lio, and E. Bertolazzi, "Combining safety margins and user preferences into a driving criterion for optimal control-based computation of reference maneuvers for an adas of the next generation," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, June 2005. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1505074

[39] "Wordnet, a lexical database for the english language," 2005. [Online]. Available: http://wordnet.princeton.edu/

[40] "Bosch's controller area network," 2005. [Online]. Available: http://www.can.bosch.com

[41] M. Kais, S. Dauvillier, A. D. L. Fortelle, I. Masaki, and C. Laugier, "Towards outdoor localization using gis, vision system and stochastic error propagation," in *Proceedings of the second International Conference on Autonomous Robots and Agents*, Palmerston North, New Zealand, December 2004.

[42] E. Yang, D. Gu, T. Mita, , and H. Hu, "Nonlinear tracking control of a car-like mobile robot via dynamic feedback linearization," in *Proceedings of Control*, University of Bath, September 2004.

[43] Wikipedia, the free encyclopedia, "Kinematics," 2005. [Online]. Available: http://en.wikipedia.org/wiki/Kinematics

[44] P. Brault, H. Mounier, N. Petit, and P. Rouchon, "Flatness based tracking control of a manoeuvrable vehicle : the picar," in *Proceedings of the Fourteenth International Symposium of Mathematical Theory of Networks and Systems*, Perpignan, France, June 2000.

[45] J. Hermosillo and S. Sekhavat, "Feedback control of a bi-steerable car using flatness; application to trajectory tracking," in *Proc. of the American Control Conference*, Denver, CO (US), June 2003. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2003/HS03

[46] P. Bonnifait, P. Bouron, P. Crubillé, and D. Meizel, "Data fusion of four abs sensors and gps for an enhanced localization of car-like vehicles," in *IEEE International Conference on Robotics and Automation*, Seoul, South Korea, May 2001, pp. 1597–1602.

[47] "Galileo european satellite navigation system," 2005. [Online]. Available: http://europa.eu.int/comm/dgs/energy_transport/galileo/index_en.htm

[48] M. E. E. Najjar and P. Bonnifait, "A roadmap matching method for precise vehicle localization using belief theory and kalman filtering," in *Proceedings of ICAR03 - The 11th International Conference on Advanced Robotics*, Portugal, July 2003, pp. 1677–1682.

[49] "Tomtom, portable gps car navigation systems," 2005. [Online]. Available: http://www.tomtom.com

[50] Wikipedia, the free encyclopedia, "Global positioning system," 2005. [Online]. Available: http://en.wikipedia.org/wiki/GPS

[51] T. N. Limited, "Trimble gps tutorial," 2005. [Online]. Available: http://www.trimble.com/gps/

[52] J. Chao, Y. qi Chen, W. Chen, X. Ding, Z. Li, N. Wong, and M. Yu, "An experimental investigation into the performance of gps-based vehicle positioning in very dense urban areas," *Journal of Geospatial Engineering*, vol. 3, pp. 59–66, 2001. [Online]. Available: http://www.lsgi.polyu.edu.hk/staff/ZL.Li/vol_3_1/06_chao.pdf

[53] D. Simon, "From here to infinity," *Embedded Systems Programming*, vol. 14, no. 11, pp. 20–32, October 2001. [Online]. Available: http://www.embedded.com/story/OEG20010925S0091

[54] P. L. Rawicz, "H [infinity symbol]/h2/kalman filtering of linear dynamical systems via variational techniques with applications to target tracking," Ph.D. dissertation, Drexel University, November 2002. [Online]. Available: http://dspace.library.drexel.edu/handle/1860/79

[55] A. Hoess, "Multifunctional automotive radar network (radarnet)," RadarNet Consortium, Final Report, December 2004. [Online]. Available: http://www.radarnet.org

[56] P. deliverable, ProFusion Subproject, "Sensor data sheets," 2004. [Online]. Available: http://www.prevent-ip.org/en/public_documents/deliverables/

[57] "Sick, sensor intelligence," 2005. [Online]. Available: http://www.sick.com

[58] "Velodyne hd lidar website," 2007. [Online]. Available: http://www.velodyne.com/lidar

[59] "Prevent subproject usercams," 2005. [Online]. Available: http://tinyurl.com/542c2j

[60] J. Diebel, K. Reutersward, S. Thrun, J. Davis, and R. Gupta, "Simultaneous localization and mapping with active stereo vision," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sendai, Japan, September 2004.

[61] J. Tardos, J. Neira, P. Newman, and J. Leonard, "Robust mapping and localization in indoor environments using sonar data," *International Journal of Robotics Research*, vol. 21, no. 4, pp. 311–330, April 2002. [Online]. Available: http://oe.mit.edu/~jleonard/

[62] S. Thrun, D. Fox, and W. Burgard, "A probabilistic approach to concurrent mapping and localization for mobile robots," *Machine Learning*, vol. 31, pp. 29–53, 1998, also appeared in Autonomous Robots 5, 253–271 (joint issue). [Online]. Available: http://www-2.cs.cmu.edu/~thrun/papers/thrun.maploc.html

[63] V. Varveropoulos, "Robot localization and map construction using sonar data," 2005. [Online]. Available: http://rossum.sourceforge.net/papers/Localization/PosPosterv4.pdf

[64] R. Rouveure, M. O. Monod, and P. Faure, "Mapping of the environment with a high resolution ground-based radar imager," in *Proceedings of the 14th IEEE Mediterranean Electrotechnical Conference*, 2008. [Online]. Available: http://impala.cemagref.fr

[65] K. Fürstenberg and T. Kluge, "Laserscanner for automotive applications," in *WIT Workshop for Intelligence Transport*, 2004.

[66] M. Wiedemann, M. Sauer, F. Driewer, and K. Schilling, "Analysis and characterization of the pmd camera for application in mobile robotics," in *Proceedings of the 17th World Congress of The International Federation of Automatic Control*, Seoul, Korea, July 2008.

[67] R. Sheh, M. W. Kadous, and C. Sammut, "On building 3d maps using a range camera: Applications to rescue robotics," UNSW, Sydney, Australia, Tech. Rep. UNSW-CSE-TR-0609, 2006. [Online]. Available: http://rsheh.web.cse.unsw.edu.au

[68] W. Zhou, J. V. Miro, and G. Dissanayake, "6d appearance slam based on ranging vision and information theory," in *Proceedings of the IEEE/RSJ International Conference on Robots and Systems*, Nice, France, September 2008, pp. 2072–2077. [Online]. Available: http://www.cas.edu.au/publications

[69] E. Royer, M. Lhuillier, M. Dhome, and T. Chateau, "Towards an alternative gps sensor in dense urban environment from visual memory," in *Proceedings of the 15th British Machine Vision Conference*, London, United Kingdom, September 2004. [Online]. Available: http://wwwlasmea.univ-bpclermont.fr/Personnel/Maxime.Lhuillier/

[70] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2000. [Online]. Available: http://robots.stanford.edu/papers/thrun.robust-mcl.html

[71] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)*, vol. 1, June 2004, pp. 652–659. [Online]. Available: http://www.eecs.berkeley.edu/~marci/VisualOdometry.pdf

[72] L. Mejias, S. Saripalli, G. Sukhatme, and P. Campoy, "Detection and tracking of external features in an urban environment using an autonomous helicopter," in *IEEE International Conference on Robotics and Automation*, 2005.

[73] M. Yang, Q. Yu, H. Wang, and B. Zhang, "Vision-based real-time obstacles detection and tracking for autonomous vehicle guidance," in *Proceedings of SPIE Photonics West 2002 - Electronic Imaging 2002: Science and Technology*, vol. 4666, San Jose, California USA, January 2002, pp. 65–74. [Online]. Available: http://yang.ming.free.fr/paper/index.htm#journals

[74] M. Bertozzi, A. Broggi, R. Chapuis, F. Chausse, A. Fascioli, and A. Tibaldi, "Shape-based pedestrian detection and localization," in *IEEE International Conference on Intelligent Transportation Systems 2003*, Shangai, China, October 2003, pp. 328–333. [Online]. Available: http://www.ce.unipr.it/people/broggi/publications

[75] A. Wedel, T. Schoenemann, T. Brox, and D. Cremers, "Warpcut - fast obstacle segmentation in monocular video," in *Pattern Recognition (Proc. DAGM)*, ser. LNCS. Heidelberg, Germany: Springer, September 2007.

[76] A. Davison, W. Mayol, and D. Murray, "Real-time localisation and mapping with wearable active vision," in *Proc. IEEE International Symposium on Mixed and Augmented Reality*. IEEE Computer Society Press, October 2003, (To appear). [Online]. Available: http://www.robots.ox.ac.uk/ActiveVision/Papers/davison_etal_ismar2003/davison_etal_ismar2003.html

[77] S. Se, D. Lowe, and J. Little, "Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks," *International Journal of Robotics Research*, vol. 21, no. 8, pp. 735–758, August 2002. [Online]. Available: http://www.cs.ubc.ca/~se/research.html

[78] A. Levin and R. Szeliski, "Visual odometry and map correlation," in *IEEE Conference on Computer Vision and Pattern Recognition*, Washington DC, June 2004. [Online]. Available: http://www.cs.huji.ac.il/~alevin/

[79] N. Simond and P. Rives, "Trajectography of an uncalibrated stereo rig in urban environments," in *IEEE RSJ/International conference on Intelligent Robot and System (IROS'04)*, Sendai, Japan, Sept. 28-Oct. 2 2004, pp. 3381–3386.

[80] A. Comport, E. Malis, and P. Rives, "Accurate quadri-focal tracking for robust 3d visual odometry," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007. [Online]. Available: http://www-sop.inria.fr/icare/personnel/andrew.comport/publications/pdf/2007_icra_comport.pdf

[81] K. Konolige, "Small vision systems: Hardware and implementation." in *Eighth International Symposium on Robotics Research*, Hayama, Japan, October 1997. [Online]. Available: http://www.ai.sri.com/~konolige/pubs.htm

[82] M. Okutomi, K. Nakano, J. Maruyama, and T. Hara, "Robust estimation of planar regions for visual navigation using sequential stereo images," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 4, 2002, pp. 3321–3327. [Online]. Available: http://okutomi-lab.ctrl.titech.ac.jp/res/RSU.html

[83] Q. Yu, H. Araujo, and H. Wang, "Stereo-vision based real-time obstacle detection for urban environments," in *Proceedings of Int. Conf. on Advanced Robotics*, Coimbra, Portugal, July 2003, pp. 1671–1676. [Online]. Available: http://iris.usc.edu/~qianyu/

[84] T. K. Marks, A. Howard, M. Bajracharya, G. W. Cottrell, and L. Matthies, "Gamma-slam: Stereo visual slam in unstructured environments using variance grid maps," in *Proceedings of the IEEE Conference on Intelligent Robots and Systems (IROS)*, 2007. [Online]. Available: http://www.ai.sri.com/conf/iros2007/5.pdf

[85] L. Paz, P. Pinies, J. Tardos, and J. Neira, "Large-scale 6-dof slam with stereo-in-hand," in *IEEE Transactions on Robotics*, vol. 24, no. 5, October 2008, pp. 946–957. [Online]. Available: http://www.ai.sri.com/conf/iros2007

[86] M. Antone and S. Teller, "Scalable, absolute position recovery for omni-directional image networks," in *Proceedings of the IEEE CVPR International Conference on Computer Vision and Pattern Recognition*, December 2001. [Online]. Available: http://graphics.csail.mit.edu/~seth/pubs/

[87] H. Andreasson and T. Duckett, "Topological localization for mobile robots using omni-directional vision and local features," in *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles*, Lisbon, Portugal, 2004. [Online]. Available: http://www.aass.oru.se/~tdt/Papers/hatd_iav2004.html

[88] D. Scaramuzza and R. Siegwart, "Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles," in *IEEE Transactions on Robotics*, vol. 24, no. 5, October 2008.

[89] A. Davison, Y. G. Cid, and N. Kita, "Real-time 3D SLAM with wide-angle vision," in *Proceedings IFAC Symposium on Intelligent Autonomous Vehicles*, Lisbon, July 2004. [Online]. Available: http://www.robots.ox.ac.uk/ActiveVision/Papers/davison_etal_iav2004/davison_etal_iav2004.html

[90] H. Andreasson, T. Duckett, and A. Lilienthal, "A minimalistic approach to appearance-based visual slam," in *IEEE Transactions on Robotics*, vol. 24, no. 5, October 2008, pp. 991–1001.

[91] M. Bertozzi, A. Broggi, A. Fascioli, T. Graf, and M. Meinecke, "Pedestrian Detection for Driver Assistance Using Multiresolution Infrared Vision," *IEEE Transaction on Vehicular Technology*, vol. 53, no. 6, pp. 1666–1678, November 2004, iSSN 0018-9545. [Online]. Available: http://www.ce.unipr.it/people/bertozzi/pap/

[92] "Swissranger, a miniature 3d time of flight camera," 2005. [Online]. Available: http://www.swissranger.ch

[93] P. deliverable, ProFusion Subproject, "Compendium on sensor data fusion," 2004. [Online]. Available: http://www.prevent-ip.org/en/public_documents/deliverables/

[94] E. T. Jaynes, *Probability Theory: the logic of science*. Cambridge University Press, January 2003. [Online]. Available: http://print.google.com/print?q=Probability+Theory+Jaynes

[95] P. Bessière, "Survey: Probabilistic methodology and techniques for artefact conception and development," INRIA-Rhone-Alpes, Tech. Rep. 4730, February 2003. [Online]. Available: http://www.inria.fr/rrrt/rr-4730.html

[96] M. Paskin, "A short course on graphical models," 2003. [Online]. Available: http://www.stanford.edu/~paskin/gm-short-course/index.html

[97] D. Bellot and P. Bessière, "Approximate discrete probability distribution representation using a multi-resolution binary tree," in *IEEE International Conference on Tools with Artificial Intelligence*, 2003. [Online]. Available: http://www.stat.berkeley.edu/~bellot/

[98] T. Minka, "Bayesian inference in dynamic models – an overview," 2005. [Online]. Available: http://research.microsoft.com/~minka/papers/dynamic.html

[99] J. Diard, P. Bessière, and E. Mazer, "A survey of probabilistic models, using the bayesian programming methodology as a unifying framework," in *Proc. of the Int. Conf. on Computational Intelligence, Robotics and Autonomous Systems*, Singapore (SG), December 2003. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2003/DBM03

[100] S. Maskell, "Sequentially structured bayesian solutions," Ph.D. dissertation, Cambridge University Engineering Department, 2004. [Online]. Available: http://www.simonmaskell.com/

[101] R. van der Merwe and E. Wan, "Sigma-point kalman filters for probabilistic inference in dynamic state-space models," in *Proceedings of the Workshop on Advances in Machine Learning*, Montreal, Canada, June 2003. [Online]. Available: http://choosh.ece.ogi.edu/spkf/

[102] I. Rekleitis, "Particle filter tutorial for mobile robots," School of Computer Science, McGill University, Montreal, Quebec, Canada, Tech. Rep., 2003. [Online]. Available: http://www.cim.mcgill.ca/~yiannis/ParticleTutorial.html

[103] I. Abuhadrous, "Système embarqué temps réel de localisation et de modélisation 3d par fusion multi-capteur," Ph.D. dissertation, Centre de Robotique, Mines de Paris, 2005. [Online]. Available: http://pastel.paristech.org/archive/00001118/

[104] R. van der Merwe and E. A. Wan, "Sigma-point kalman filters for integrated navigation," in *Proceedings of the 60th Annual Meeting of The Institute of Navigation*, Dayton, OH, June 2004. [Online]. Available: http://www.cslu.ogi.edu/publications/

[105] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, June 1989. [Online]. Available: http://csdl.computer.org/comp/mags/co/1989/06/r6046abs.htm

[106] G. K. Kraetzschmar, G. Pagès Gassull, and K. Uhl, "Probabilistic quadtrees for variable-resolution mapping of large environments," in *Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, M. I. Ribeiro and J. Santos Victor, Eds., July 2004. [Online]. Available: http://smart.informatik.uni-ulm.de/PUBLICATIONS/

[107] M. Montemerlo and S. Thrun, "A multi-resolution pyramid for outdoor robot terrain perception," in *Proceedings of the AAAI National Conference on Artificial Intelligence*. San Jose, CA: AAAI, 2004. [Online]. Available: http://robots.stanford.edu/papers/Montemerlo04a.html

[108] J. Leal, "Stochastic environment representation," Ph.D. dissertation, The University of Sydney, January 2003. [Online]. Available: http://www.acfr.usyd.edu.au/projects/research/environment-rep/stoch-repres/

[109] J. Leal, S. Scheding, and G. Dissanayake, "Probabilistic 2d mapping in unstructured environments," in *Proceedings of the Australian Conference on Robotics and Automation*, Melbourne, Australia, August 2000, pp. 19–24. [Online]. Available: http://www.acfr.usyd.edu.au/projects/research/environment-rep/stoch-repres/

[110] T. Bailey, "Mobile robot localisation and mapping in extensive outdoor environments," Ph.D. dissertation, Australian Centre for Field Robotics, University of Sydney, Agust 2002. [Online]. Available: http://www.acfr.usyd.edu.au/homepages/postgrads/tbailey/

[111] P. Biber, S. Fleck, and W. Straßer, "A probabilistic framework for robust and accurate matching of point clouds," in *26th Pattern Recognition Symposium (DAGM 04)*, 2004. [Online]. Available: http://www.gris.uni-tuebingen.de/publics/staff/Peter_Biber.html

[112] C.-C. Wang and C. Thorpe, "A hierarchical object based representation for simultaneous localization and mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, September 2004. [Online]. Available: http://www.ri.cmu.edu/pubs/pub_4721.html

[113] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*. Academic Press, 1988.

[114] Y. Bar-Shalom and X. R. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*. YBS Publishing, 1995.

[115] P. Besl and N. McKay, "A method for registration of 3-d shapes," *IEEE Transaction on Pattern Analysis And Machine Intelligence*, vol. 14, no. 2, pp. 239–25, February 1992. [Online]. Available: http://csdl.computer.org/comp/trans/tp/1992/02/i0239abs.htm

[116] Z. Zhang, "Iterative point matching for registration of free-form curves," INRIA-Sophia Antipolis, Tech. Rep. RR-1658, April 1992, equipe : ROBOTVIS. [Online]. Available: http://www.inria.fr/rrrt/rr-1658.html

[117] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Third International Conference on 3D Digital Imaging and Modeling*, 2001. [Online]. Available: http://graphics.stanford.edu/papers/fasticp/

[118] A. Fitzgibbon, "Robust registration of 2d and 3d point sets," in *The British Machine Vision Conference*, 2001. [Online]. Available: http://www.robots.ox.ac.uk/~awf/lmicp/

[119] D. Haehnel, "Mapping with mobile robots," Ph.D. dissertation, Fakultät für Angewandte Wissenschaften, Universität Freiburg, December 2004. [Online]. Available: http://www.informatik.uni-freiburg.de/~haehnel/

[120] J. Nieto, T. Bailey, and E. Nebot, "Scan-slam: Combining ekf-slam and scan correlation," Australian Centre for Field Robotics, Tech. Rep., 2004, presented at the ICRA'2005 - Workshop W-M08 - Simultaneous Localisation and Mapping Monday 18 April, 2005 (Full Day Workshop). [Online]. Available: http://www.acfr.usyd.edu.au/homepages/academic/tbailey/index.html

[121] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," in *IEEE/RJS International Conference on Intelligent Robots and Systems*, 2003. [Online]. Available: http://www.gris.uni-tuebingen.de/publics/staff/Peter_Biber.html

[122] D. Haehnel, S. Thrun, and W. Burgard, "An extension of the icp algorithm for modeling nonrigid objects with mobile robots," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2003. [Online]. Available: http://www.informatik.uni-freiburg.de/~haehnel/

[123] R. Smith, M. Self, and P. Cheeseman, "A stochastic map for uncertain spatial relationships," SRI, Tech. Rep., 1995. [Online]. Available: http://www.riacs.edu/navroot/Research/General.jsp

[124] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, pp. 333–349, 1997. [Online]. Available: http://citeseer.csail.mit.edu/lu97globally.html

[125] A. Stroupe, M. C. Martin, and T. Balch, "Merging probabilistic observations for mobile distributed sensing," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-00-30, December 2000. [Online]. Available: http://www.ri.cmu.edu/pubs/pub_3499.html

[126] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello, "Bayesian filtering for location estimation," *IEEE Pervasive Computing*, vol. 2, no. 3, pp. 24–33, July-September 2003. [Online]. Available: http://seattle.intel-research.net/people/jhightower/pubs.html

[127] D. Fox, J. Hightower, H. Kautz, L. Liao, and D. Patterson, "Bayesian techniques for location estimation," in *Proceedings of The 2003 Workshop on Location-Aware Computing*, October 2003, pp. 16–18, part of the 2003 Ubiquitous Computing Conference. [Online]. Available: http://seattle.intel-research.net/people/jhightower/pubs.html

[128] J. Gutmann, W. Burgard, D. Fox, and K. Konolige, "An experimental comparison of localization methods," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1998. [Online]. Available: http://citeseer.ist.psu.edu/article/gutmann98experimental.html

[129] J. Gutmann and D. S. Fox, "An experimental comparison of localization methods continued," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002. [Online]. Available: http://www.informatik.uni-freiburg.de/~gutmann/

[130] P. Biber and T. Duckett, "Dynamic maps for long-term operation of mobile service robots," in *Robotics: Science and Systems*, June 2005. [Online]. Available: http://www.gris.uni-tuebingen.de/~biber

[131] A. Howard and N. Roy, "The robotics data set repository (radish)," 2005. [Online]. Available: http://radish.sourceforge.net/

[132] "Openslam website," 2007. [Online]. Available: http://openslam.org/

[133] R. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," SRI Robotics Lab, Tech. Rep., 1985.

[134] U. Frese, "A discussion of simultaneous localization and mapping," *Autonomous Robots*, vol. 20, no. 1, pp. 25–42, 2006. [Online]. Available: http://www.informatik.uni-bremen.de/agbkb/publikationen/bibsearch/detail_e.htm?pk_int=1646

[135] M. Montemerlo, "Fastslam: A factored solution to the simultaneous localization and mapping problem with unknown data association," doctoral dissertation, tech. report CMU-RI-TR-03-28, Robotics Institute, Carnegie Mellon University, July 2003. [Online]. Available: http://www.ri.cmu.edu/people/montemerlo_michael.html

[136] J. Castellanos, J. Neira, and J. Tardos, "Limits to the consistency of ekf-based slam," in *5th IFAC Symposium on Intelligent Autonomous Vehicles, IAV'04*, Lisbon, Portugal, July 2004. [Online]. Available: http://webdiis.unizar.es/%7Ejdtardos/

[137] S. Julier and J. Uhlman, "A counter example to the theory of simultaneous localisation and map building," in *IEEE International Conference on Robots and Automation*, Seoul, South Korea, October 2001. [Online]. Available: http://132.250.128.5/vrlab/

[138] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the ekf-slam algorithm," 2005, -. [Online]. Available: http://www.acfr.usyd.edu.au/homepages/academic/tbailey/

[139] A. S. J. Andrade, T. Alejandra Vidal, "Unscented transformation of vehicle states in slam," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005. [Online]. Available: http://www-iri.upc.es/groups/lrobots/publications.html

[140] J. J. Leonard and H. J. S. Feder, "Decoupled stochastic mapping," *IEEE Journal of Oceanic Engineering*, vol. -, pp. 561–571, October 2001. [Online]. Available: http://albacore.mit.edu/~jleonard/pubs/

[141] J. Knight, A. J. Davison, and I. Reid, "Constant time slam using postponement (pdf format),," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001. [Online]. Available: http://www.doc.ic.ac.uk/~ajd/publications.html

[142] W. SB, D. G, and D.-W. HF, "An efficient approach to the simultaneous localisation and mapping problem," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, February 2002, pp. 406–411. [Online]. Available: http://www.acfr.usyd.edu.au/

[143] J. Guivant and E. Nebot, "Improving computational and memory requirements of simultaneous localization and map building algorithms," in *Proceedings of the IEEE International Conference on Robotics & Automation*, Washington DC, USA, May 2002, pp. 2731–2736. [Online]. Available: http://robot.anu.edu.au/~david/slam2004/

[144] S. Julier and J. Uhlmann, "Using multiple slam algorithms," in *International Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, October 2003.

[145] M. C. Bosse, P. M. Newman, J. J. Leonard, and S. Teller, "Slam in large-scale cyclic environments using the atlas framework." *The International Journal of Robotics Research*, vol. 23, no. 12, pp. 1113–1139, 2004. [Online]. Available: http://albacore.mit.edu/~jleonard/pubs/

[146] C. Estrada, J. Neira, and J. Tardos, "Hierarchical slam: real-time accurate mapping of large environments," 2005, to appear on IEEE Transactions on Robotics. [Online]. Available: http://webdiis.unizar.es/~cestrada

[147] R. Eustice, H. Singh, and J. Leonard, "Exactly sparse delayed-state filters," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005, pp. 2428–2435, best Student Paper Award. [Online]. Available: http://www.whoi.edu/hpb/viewPage.do?id=1160\&cl=3

[148] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *International Journal of Robotics Research*, vol. 23, pp. 693–716, 2004, to Appear. [Online]. Available: http://robots.stanford.edu/papers/thrun.seif.html

[149] M. A. Paskin, "Thin junction tree filters for simultaneous localization and mapping," in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*. San Francisco, CA: G. Gottlob and T. Walsh, 2003, pp. 1157–1164, this paper won a Distinguished Paper Award. [Online]. Available: http://paskin.org

[150] Z. Wang, S. Huang, and G. Dissanayake, "Implementation issues and exp erimental evaluation of d-slam," in *The 5th International Conference on Field and Service Robotics*, Port Douglas, Australia, July 2005. [Online]. Available: http://services.eng.uts.edu.au/~sdhuang/

[151] K. Konolige, "Large-scale map-making," in *Proceedings of the National Conference on AI (AAAI)*, San Jose, CA, 2004. [Online]. Available: http://www.ai.sri.com/~konolige/slam.htm#Kon04c

[152] T. Duckett, S. Marsland, and J. Shapiro, "Learning globally consistent maps by relaxation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2000)*, San Francisco, CA, 2000. [Online]. Available: http://aass.oru.se/~tdt/Papers/icra00.html

[153] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localisation and mapping," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 196–207, April 2005. [Online]. Available: http://www.informatik.uni-bremen.de/~ufrese/slammlr_e.html

[154] M. Bray, E. Koller-Meier, N. Schraudolph, and L. V. Gool, "Stochastic meta-descent for tracking articulated structures," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[155] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," in *Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007. [Online]. Available: http://www.informatik.uni-freiburg.de/~stachnis/pdf/grisetti07rss.pdf

[156] E. Olson, J. Leonard, and S. Teller, "Fast iterative optimization of pose graphs with poor initial estimates," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2006, pp. 2262–2269. [Online]. Available: http://rvsn.csail.mit.edu/graphoptim/eolson-graphoptim2006.pdf

[157] E. Olson, J. J. Leonard, and S. J. Teller, "Spatially-adaptive learning rates for online incremental slam." in *Proceedings of Robotics: Science and Systems*, W. Burgard, O. Brock, and C. Stachniss, Eds. Atlanta, GA, USA: The MIT Press, June 2007. [Online]. Available: http://rvsn.csail.mit.edu/eolson/graphincr/eolson_graph_incr_2007.pdf

[158] D. Hähnel, W. Burgard, B. Wegbreit, and S. Thrun, "Towards lazy data association in SLAM," in *Proceedings of the 11th International Symposium of Robotics Research (ISRR'03)*. Sienna, Italy: Springer, 2003. [Online]. Available: http://robots.stanford.edu/papers/Haehnel03c.html

[159] S. Thrun, "A probabilistic online mapping algorithm for teams of mobile robots," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 335–363, 2001. [Online]. Available: http://robots.stanford.edu/papers/thrun.maps-multi.html

[160] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot, "Fastslam: An efficient solution to the simultaneous localization and mapping problem with unknown data association," *Journal of Machine Learning Research*, vol. to appear, p. to appear, 2004, to appear. [Online]. Available: http://robots.stanford.edu/papers/Thrun03g.html

[161] Hähnel, D., Fox, D., W. Burgard, and S. Thrun, "A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements," in *Proceedings of the Conference*

on Intelligent Robots and Systems (IROS), 2003. [Online]. Available: http://robots.stanford.edu/papers/Haehnel03b.html

[162] A. Eliazar and R. Parr, "Dp-slam 2.0," in *IEEE International Conference on Robotics and Automation*, 2004. [Online]. Available: http://www.cs.duke.edu/~parr/dpslam/

[163] D. Schulz, W. Burgard, D. Fox, and A. Cremers, "Tracking multiple moving objects with a mobile robot," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001. [Online]. Available: http://www.informatik.uni-bonn.de/~rhino/publications/abstracts/cvpr01.html

[164] L. Montesano, J. Minguez, and L. Montano, "Modeling the static and the dynamic parts of the environment to improve sensor-based navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, 2005. [Online]. Available: http://webdiis.unizar.es/~jminguez

[165] C.-C. Wang and C. Thorpe, "Simultaneous localization and mapping with detection and tracking of moving objects," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Washington, DC, May 2002. [Online]. Available: http://www.acfr.usyd.edu.au/people/academic/bobwang/Papers/wang_icra02.html

[166] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun., "Map building with mobile robots in dynamic environments." in *In Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003. [Online]. Available: http://www.informatik.uni-freiburg.de/~haehnel/

[167] D. F. Wolf and G. S. Sukhatme, "Mobile robot simultaneous localization and mapping in dynamic environments," *Autonomous Robots*, vol. 19, no. 1, pp. 53–65, July 2005. [Online]. Available: http://www-robotics.usc.edu/~gaurav/Papers/publications.php

[168] N. Kaempchen and K. Dietmayer, "Imm vehicle tracking for traffic jam situations on highways," in *Proceedings of 7th International Conference on Information Fusion*, Stockholm, Sweden, June 2004. [Online]. Available: http://www.argos.uni-ulm.de

[169] H. M., M. H., and J. A., "Imm-jvc and imm-jpdaf algorithms for closely maneuvering targets," in *Proceedings of the 36th Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, USA, November 2001, pp. 1278–1282. [Online]. Available: http://www.cim.mcgill.ca/~melita/

[170] C. Rasmussen and G. D. Hager, "Probabilistic data association methods for tracking complex visual objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 560–576, June 2001. [Online]. Available: http://www.cs.jhu.edu/~hager/Public/Publications/

[171] C. Hue, J.-P. Le Cadre, and P. Pérez, "Tracking multiple objects with particle filtering," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 3, pp. 791–812, July 2002. [Online]. Available: http://www.irisa.fr/vista/Publis/Publi/Hue02d.html

[172] D. Schulz, W. Burgard, D. Fox, and A. Cremers, "People tracking with a mobile robot using sample-based joint probabilistic data association filters," *International Journal of Robotics Research (IJRR)*, vol. 22, no. 2, pp. 99–116, 2003. [Online]. Available: http://www.informatik.uni-freiburg.de/~burgard/abstracts/people-tracking-ijrr-03.abstract.html

[173] K. C. Fuerstenberg, D. T. Linzmeier, and K. C. Dietmayer, "Pedestrian recognition and tracking of vehicles using a vehicle based multilayer laserscanner," in *Proceedings of the 10th World Congress on Intelligent Transport Systems (ITS)*, Madrid, Spain, November 2003. [Online]. Available: http://www.argos.uni-ulm.de/

[174] B. Kluge, C. Koehler, and E. Prassler, "Fast and robust tracking of multiple moving objects with a laser range finder," in *IEEE Proceedings of the International Conference on Robotics and Automation*, Seoul, May 2001. [Online]. Available: http://www.informatik.uni-freiburg.de/~ckoehler/

[175] D. Hähnel, D. Schulz, and W. Burgard, "Map building with mobile robots in populated environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002. [Online]. Available: http://www.informatik.uni-freiburg.de/~burgard/abstracts/haehnel-iros02.html

[176] C.-C. Wang, C. Thorpe, and A. Suppe, "Ladar-based detection and tracking of moving objects from a ground vehicle at high speeds," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, Columbus, OH, June 2003. [Online]. Available: http://www.acfr.usyd.edu.au/people/academic/bobwang/Papers/wang_iv03.html

[177] C. Schlenoff, "Linking sensed images to an ontology of obstacles to aid in autonomous driving," in *Proceedings of the AAAI. Workshop on Ontologies and the Semantic Web*, no. 550, Edmonton, Canada, July - August 2002. [Online]. Available: http://www.isd.mel.nist.gov/documents/publist.htm

[178] A. D. Vasquez Gomea, F. Large, T. Fraichard, and C. Laugier, "Moving obstacles' motion prediction for autonomous navigation," in *Proc. of the Int. Conf. on Control, Automation, Robotics and Vision*, Kunming, China, December 2004. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2004/VLFL04a

[179] Project-team e-motion, "Geometry and probability for motion and action, e-motion research project activity reports," INRIA Rh?ne-Alpes Research Unit, Tech. Rep., 2004. [Online]. Available: http://www.inria.fr/rapportsactivite/RA2004/e-motion2004/e-motion_tf.html

[180] M. Bennewitz, W. Burgard, and S. Thrun, "Using EM to learn motion behaviors of persons with mobile robots," in *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, 2002. [Online]. Available: http://robots.stanford.edu/papers/bennewitz.02iros.html

[181] L. Liao, D. Fox, and H. Kautz, "Learning and inferring transportation routines." in *Proceedings of the National Conference on Artificial Intelligence*, 2004. [Online]. Available: http://www.cs.washington.edu/ai/Mobile_Robotics/abstracts/gps-aaai-04.abstract.html

[182] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*, R. C. Arkin, Ed. MIT Press, September 2005. [Online]. Available: http://mitpress.mit.edu/catalog/item/default.asp?ttype=2\&tid=10668

[183] W. Zhang and J. Kosecka, "Image based localization in urban environments," *International Symposium on 3D Data Processing, Visualization and Transmission*, vol. 0, pp. 33–40, 2006.

[184] B. Ferris, D. Fox, and N. Lawrence, "Wifi-slam using gaussian process latent variable models," in *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI), 2007*, 2007. [Online]. Available: http://www.cs.washington.edu/homes/fox/abstracts/gplvm-wifi-slam-ijcai-07.abstract.html

[185] M. Magnusson, T. Duckett, R. Elsrud, and L.-E. Skagerlund, "3d modelling for underground mining vehicles," in *SimSafe 2005, Proceedings of the Conference on Modeling and Simulation for Public Safety*, 2005. [Online]. Available: http://aass.oru.se/~mmn/

[186] A. Censi, "An accurate closed-form estimate of ICP's covariance," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Rome, Italy, April 2007, pp. 3167–3172. [Online]. Available: http://purl.org/censi/2006/icpcov

[187] A. Broggi, "Robust real-time lane and road detection in critical shadow conditions," in *Proceedings IEEE International Symposium on Computer Vision*. Coral Gables, Florida, USA: IEEE Computer Society, November 1995. [Online]. Available: http://www.ce.unipr.it/people/broggi

[188] D. Hoiem, "Seeing the world behind the image: Spatial layout for 3d scene understanding," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, August 2007. [Online]. Available: http://www.cs.cmu.edu/~dhoiem/

[189] G. Dubbelman, W. van der Mark, J. C. van den Heuvel, and F. C. Groen, "Obstacle detection during day and night conditions using stereo vision," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, USA, October 2007.

[190] M. Okutomi and S. Noguchi, "Extraction of road region using stereo images," in *Proceedings of International Conference on Pattern Recognition*, vol. 1, August 1998, pp. 853–856. [Online]. Available: http://www.ok.ctrl.titech.ac.jp/res/RSU.html

[191] V. N. Vapnik, *The Nature of Statistical Learning Theory*, ser. Computational learning theory. Springer, 2000.

[192] N. Lawrence, J. Platt, and M. Jordan, "Extensions of the informative vector machine," in *Proc. Sheffield Machine Learning Workshop,*, ser. Springer Lecture Notes on Computer Science, 2005.

[193] M. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001. [Online]. Available: http://www.miketipping.com/index.php?page=rvm

[194] R. Herbrich, *Learning Kernel Classifiers*. The MIT Press, 2002. [Online]. Available: http://www.learning-kernel-classifiers.org/

[195] A. Saxena, S. H. Chung, and A. Y. Ng, "3-d depth reconstruction from a single still image," *International Journal of Computer Vision (IJCV)*, August 2007. [Online]. Available: http://make3d.stanford.edu/publications.html

[196] S. Bileschi, "Object detection at multiple scales improves accuracy," in *19th International Conference on Pattern Recognition (ICPR)*, Tampa, FL, USA, 2008, pp. 1–5. [Online]. Available: http://cbcl.mit.edu

[197] S. Petti and T. Fraichard, "Reactive planning under uncertainty among moving obstacles," in *Proc. of the Int. Symp. on Robotics*, 2005. [Online]. Available: http://hal.inria.fr/inria-00182044/

[198] "Google earth website," 2007. [Online]. Available: http://earth.google.com/

[199] O. Mehani, R. Benenson, S. Lemaignan, and T. Ernst, "Networking needs and solutions for road vehicles at imara," in *Proceedings of the 7th International Conference on Intelligent Transport Systems Telecommunications*, 2007. [Online]. Available: http://hal.inria.fr/inria-00147286/

[200] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. [Online]. Available: http://planning.cs.uiuc.edu/

[201] R. Philippsen, S. Kolski, K. Macek, and R. Siegwart, "Path planning, replanning, and execution for autonomous driving in urban and offroad environments," in *Proceedings of the Workshop on Planning, Perception and Navigation for Intelligent Vehicles at the IEEE International Conference on Robotics and Automation*, Rome, Italy, April 2007.

[202] D. Ferguson and A. Stentz, "Anytime, dynamic planning in high-dimensional search spaces," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, Rome, Italy, 2007. [Online]. Available: http://pittsburgh.intel-research.net/~dfergus1/

[203] J. van den Berg, "Path planning in dynamic environments," Ph.D. dissertation, Utrecht University, The Netherlands, 2007. [Online]. Available: http://www.cs.unc.edu/~berg/

[204] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 1st ed. Prentice Hall, 1995.

[205] H. Daumé and D. Marcu, "Learning as search optimization: approximate large margin methods for structured prediction," in *Proceedings of the 22nd international conference on Machine learning*, ser. ACM International Conference Proceeding, vol. 119, Bonn, Germany, 2005, pp. 169 – 176.

[206] P. S. and F. T., "Safe navigation of a car-like robot in a dynamic environment," in *Proc. of the European Conf. on Mobile Robots*, Ancona, Italy, 2005. [Online]. Available: http://hal.inria.fr/inria-00182047

[207] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990. [Online]. Available: http://www.dtc.umn.edu/~reedsj

[208] A. Scheuer and T. Fraichard, "Planning continuous-curvature paths for car-like robots," in *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 3, Osaka, Japan, November 1996, pp. 1304 – 1311. [Online]. Available: http://citeseer.ist.psu.edu/scheuer96planning.html

[209] P. R. S. Benhimane, E. Malis and J. R. Azinheira, "Vision-based control for car platooning using homography decomposition," in *IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005, pp. 2173–2178. [Online]. Available: http://www-sop.inria.fr/icare/personnel/malis/

[210] J. Lu, S. Sakhavat, X. Ming, and C. Laugier, "Sliding mode control for nonholonomic mobile robot," in *Proceedings of the International Conference on Control, Automation, Robotics and Vision, ICARCV*, 2000. [Online]. Available: ftp://ftp.inrialpes.fr/pub/emotion/sharp-publications/

[211] B. Thuilot, J. Bom, F. Marmoiton, and P. Martinet, "Accurate automatic guidance of an urban electric vehicle relying on a kinematic gps sensor," in *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles*, Lisboa, Portugal, July 2004.

[212] J. Bom, B. Thuilot, F. Marmoiton, and P. Martinet, "Nonlinear control for urban vehicles platooning, relying upon a unique kinematic gps," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005.

[213] A. D. Luca, G. Oriolo, and C. Samson, "Feedback control of a nonholonomic car-like robot," in *Planning robot motion*, J.-P. Laumond, Ed. Berlin, DE: Springer-Verlag, 1998. [Online]. Available: http://citeseer.ist.psu.edu/deluca97feedback.html

[214] B. C. C. M. P. Lenain, R. Thuilot, "Model predictive control for vehicle guidance in presence of sliding: Application to farm vehicles path tracking," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA*, April 2005, pp. 885 – 890.

[215] INRIA, "Scilab website," 2008. [Online]. Available: http://www.scilab.org

[216] S. Boissé, R. Benenson, L. Bouraoui, M. Parent, and L. Vlacic, "Cybernetic transportation systems design and development: Simulation software cybercars," in *2007 IEEE International Conference on Robotics and Automation*, Roma, Italy, 2007. [Online]. Available: http://hal.inria.fr/inria-00126677

[217] 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*, vol. 50, no. 1, pp. 51–68, 2005. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2005/PHKBBL05

[218] S. Kolski, D. Ferguson, M. Bellino, and R. Seigwart, "Autonomous driving in structured and unstructured environments," in *IEEE Intelligent Vehicles Symposium*, June 2006, pp. 558–563. [Online]. Available: http://asl.epfl.ch/member.php?SCIPER=164402

[219] P. Martinet, B. Thuilot, and J. Bom, "From autonomous navigation to platooning in urban context," in *Proceedings of the IARP-Workshop on Adaptive and Intelligent Robots : Present and Future*, vol. 1, no. 1, Moscou, Russia, November 2005, pp. 1–9. [Online]. Available: http://wwwlasmea.univ-bpclermont.fr/Control/index.html

[220] "Dash express automotive navigation system website," 2007. [Online]. Available: http://dash.net/product.php

[221] A. Howard, G. S. Sukhatme, and M. J. Mataric, "Multi-robot mapping using manifold representations," in *IEEE International Conference on Robotics and Automation*, New Orleans, Louisiana, April 2004, pp. 4198–4203.

[222] S. Thrun, W. Burgard, and D. Fox, "A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. San Francisco, CA: IEEE, 2000. [Online]. Available: http://robots.stanford.edu/papers/thrun.map3d.html

[223] "Robocup, international robot soccer championship," 2007. [Online]. Available: http://www.robocup.org/

[224] D. Goehring and H.-D. Burkhard, "Multi robot object tracking and self localization using visual percept relations," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006. [Online]. Available: http://www.aiboteamhumboldt.com/

[225] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, "Distributed multirobot exploration and mapping," in *Proceedings of the IEEE*, vol. 94, no. 7, July 2006, pp. 1325– 1339. [Online]. Available: http://www.cs.washington.edu/homes/fox/abstracts/multi-exploration-ieee-06.abstract.html

[226] L. Chaimowicz, A. Cowley, D. Gomez-Ibanez, B. Grocholsky, M. A. Hsieh, H. Hsu, J. F. Keller, V. Kumar, R. Swaminathan, and C. J. Taylor, "Deploying air-ground multirobot teams in urban environments," in *Multirobot Workshop*, Washington DC, 2005. [Online]. Available: http://www.cis.upenn.edu/~kumar/mars2020/

[227] R. Regele and P. Levi, "Cooperative multi-robot path planning by heuristic priority adjustment," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2006, pp. 5954–5959.

[228] N. Karam, F. Chausse, R. Aufrere, and R. Chapuis, "Localization of a group of communicating vehicles by state exchange," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2006, pp. 519–524.

[229] S. Bonnabel, "Left-invariant extended kalman filter and attitude estimation," in *IEEE Conference on Decision and Control*, 2007. [Online]. Available: http://www.silvere-bonnabel.com

[230] J. Ko, Y. Kim, W. Kang, and J. Lee, "Robust electric compass to dynamic magnetic field interference," in *Proceedings of the International Conference on Control, Automation and Systems*, August 2004. [Online]. Available: http://robotics.ee.pusan.ac.kr/Pdf/Robust%20Electric%20Compass%20to%20Dynamic%20Magnetic%20Field%20Interference.pdf

[231] B. Leibe, N. Cornelis, K. Cornelis, and L. V. Gool, "Dynamic 3d scene analysis from a moving vehicle," in *Proceedings of the IEEE CVPR International Conference on Computer Vision and Pattern Recognition*, June 2007. [Online]. Available: http://www.vision.ee.ethz.ch/~bleibe/cvpr07/

[232] A. Seki and M. Okutomi, "Robust obstacle detection in general road environment based on road extraction and pose estimation," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, June 2006, pp. 437–444.

[233] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transaction on Pattern Analysis And Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, June 2007. [Online]. Available: http://www.doc.ic.ac.uk/~ajd/

[234] J. Hawkins, *On Intelligence*. Henry Holt & Company, 2004. [Online]. Available: http://www.onintelligence.org/

Perception pour véhicule urbain sans conducteur: conception et implémentation

Enjeux Le développement de véhicules sans conducteur capables de se déplacer sur des routes urbaines pourrait offrir des avantages importants dans la diminution des accidents, le confort et la réduction des coûts de déplacements. Ce document traite de la manière de créer un système de perception permettant à un robot de conduire sur des routes sans devoir adapter l'infrastructure, sans avoir besoin de visites préalables, et en prenant en compte la présence de piétons et d'autres voitures.

Positionnement du sujet Nous affirmons que le processus de perception est spécifique à l'application visée et que, par nature, il doit être capable de gérer les incertitudes dans la connaissance du monde.

Nous analysons le problème de perception pour une conduite sûre dans les environnements urbains et proposons une solution où le processus de perception est formulé comme un processus d'optimisation.

Résultats Une première contribution de la thèse étudie l'aspect incertain des modèles reconstruits à partir d'un robot et leur relation avec la tâche de planification de mouvement. Ainsi, des principes généraux de sécurité de mouvement sont définis. Cette approche est l'une des premières à prendre explicitement en compte la relation perception-planification pour la sécurité.

Pour résoudre le problème de perception abordé, nous avons conçu un nouvel algorithme de SLAMMOT basé sur laser. Cet algorithme permet la localisation, la création de cartes, la détection et le suivit d'objets en mouvement, de façon simultanée, en respectant les contraintes de temps de calcul et sans faire d'hypothèses contraignantes sur la géométrie ou la nature de l'environnement.

Ce système de perception fut ensuite couplé avec un planificateur sûr et un contrôleur sophistiqué pour réaliser des expérimentations à pleine échelle sur notre véhicule électrique automatisé, le Cycab. Cette réalisation est parmi les premières à mettre en place un véhicule capable de naviguer de façon sûre en milieux dynamiques inconnus.

Transferts des résultats vers l'industrie Les principes étudiés et le système conçu peuvent se décliner dans une multiplicité d'applications. Ces idées peuvent être employées pour la conception de systèmes d'aide à la conduite, de sécurité active, de stationnement automatique, ou directement dans la conception d'autres robots mobiles qui doivent évoluer de façon sûre en milieux dynamiques et inconnus.

Mots clés SLAMMOT, perception, planification, sécurité, véhicule sans conducteur, robotique