Vehicle Perception: Localization, Mapping with Detection, Classification and Tracking of Moving Objects

Trung-Dung Vu

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N^o attribué par la bibliothèque
|_|_|_|_|_|_|_|_|_|_|

THÈSE

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : ***"Imagerie, Vision, Robotique"***

préparée au laboratoire LIG et l'INRIA Rhône-Alpes, dans le cadre de
l'École doctorale *"Mathématiques, Sciences et Technologies de l'Information, Informatique"*

présentée et soutenue publiquement par

Trung-Dung VU

le DATE DE SOUTENANCE: 18 Septembre 2009

Titre :

Vehicle Perception: Localization, Mapping with Detection, Classification and Tracking of Moving Objects

Co-directeurs de thèse :

Christian LAUGIER et Olivier AYCARD

Composition du jury :

M.	Augustin LUX	Président
M.	Michel DEVY	Rapporteur
M.	Fawzi NASHASHIBI	Rapporteur
M.	Cédric PRADALIER	Examinateur
M.	Christian LAUGIER	Directeur de thèse
M.	Olivier AYCARD	Co-directeur de thèse

Abstract

Perceiving or understanding the environment surrounding of a vehicle is a very important step in building driving assistant systems or autonomous vehicles. In this thesis, we study problems of simultaneous localization and mapping (SLAM) with detection, classification and tracking moving objects in context of dynamic outdoor environments focusing on using laser scanner as a main perception sensor. It is believed that if one is able to accomplish these tasks reliably in real time, this will open a vast range of potential automotive applications.

The first contribution of this research is made by a grid-based approach to solve both problems of SLAM with detection of moving objects. To correct vehicle location from odometry we introduce a new fast incremental scan matching method that works reliably in dynamic outdoor environments. After good vehicle location is estimated, the surrounding map is updated incrementally and moving objects are detected without a priori knowledge of the targets. Experimental results on datasets collected from different scenarios demonstrate the efficiency of the method.

The second contribution follows the first result after a good vehicle localization and a reliable map are obtained. We now focus on moving objects and present a method of simultaneous detection, classification and tracking moving objects. A model-based approach is introduced to interpret the laser measurement sequence over a sliding window of time by hypotheses of moving object trajectories. The data-driven Markov chain Monte Carlo (DDMCMC) technique is used to solve the data association in the spatio-temporal space to effectively find the most likely solution. We test the proposed algorithm on real-life data of urban traffic and present promising results.

The third contribution is an integration of our perception module on a real vehicle for a particular safety automotive application, named Pre-Crash. This work has been performed in the framework of the European Project PReVENT-ProFusion[1] in collaboration with Daimler. A comprehensive experimental evaluation based on relevant crash and non-crash scenarios is presented which confirms the robustness and reliability of our proposed method.

[1] http://www.prevent-ip.org/profusion

Table of Contents

1 Introduction **1**

1.1 Context: Intelligent Systems for Vehicles 1
1.2 Vehicle Perception: Problem Statement 3
1.3 Contributions 7
1.4 Thesis Outline 8

2 Vehicle Perception - State of The Art **9**

2.1 Introduction 9
2.2 Simultaneous Localization and Mapping 14
2.2.1 Map Representation 16
2.2.2 Kalman Filter SLAM 19
2.2.3 Maximum Likelihood SLAM 21
2.2.4 FastSLAM 23
2.2.5 Comparison of SLAM techniques 24
2.3 Detection and Tracking Moving Objects 25
2.3.1 Moving Object Detection 26
2.3.2 Tracking of Moving Objects 28
2.4 SLAM with DATMO 34
2.5 Summary 36

3 Grid-based SLAM with Detection of Moving Objects **39**

3.1 Introduction 39
3.2 Grid-based SLAM 40
3.2.1 Grid Mapping with Known Trajectories 40
3.2.2 Grid-based Scan Matching 45
3.2.3 Local Mapping vs. Global Mapping 50
3.3 Moving Object Detection 51

3.4 Experimental Results 54
3.5 Summary 56

4 DATMO using Markov Chain Monte Carlo 59
4.1 Introduction 59
4.2 DATMO Formulation 61
4.2.1 Object models 62
4.2.2 Solution Space 64
4.2.3 Prior Probability 65
4.2.4 Likelihood Probability 67
4.2.5 Posterior Probability 71
4.3 Efficient MAP Computation using MCMC 71
4.3.1 MCMC algorithm 72
4.3.2 Moving object hypothesis generation 74
4.3.3 Neighborhood graph of hypotheses 77
4.3.4 Markov chain dynamics 79
4.3.5 Incremental computation 80
4.4 Experimental Results 81
4.4.1 Performance Evaluation 84
4.5 Summary 85

5 Appplication: PreCrash 87
5.1 Introduction 87
5.2 The Demonstrator Vehicle 89
5.3 PreCrash System 91
5.3.1 Data Fusion for Object Tracking 92
5.3.2 Situation Analysis and Decision 95
5.4 Experimental Results 99
5.5 Summary 104

6 Conclusions and Perspectives 107
6.1 Conclusions 107
6.2 Perspectives 109

Bibliography 111

Chapter 1

Introduction

1.1 Context: Intelligent Systems for Vehicles

In order to make driving experience safer and more convenient, over decades, researchers from different fields of robotics, automotive engineering and allied fields have tried to develop intelligent systems for modern vehicles. These systems are designed to help driving automatically or to monitor a human driver and assist him in navigation. They can warn the driver in case of a developing dangerous situation and can provide capabilities of avoiding collisions or mitigate the consequence if there is an inevitable collision. Moreover, these intelligent vehicle systems should be able to operate in all traffic situations wherever on highways or in crowed urban streets. Here is an example of the ultimate research goal to realize such a comprehensive system:

Imagining that you are driving to an unknown city. During your trip, the driving assistant system acts as an attentive co-driver. You will be warned of bicyclists from the right that you have failed to recognize. At the intersection, it will save you from a possible rear-end collision if you are distracted. On the highway, the car is able to take over control with adaptive cruise control mode. Steering is based on the reliable recognition of lanes, longitudinal control exploits the vision system to take into account the speed limits. If you prefer driving yourself, you still get this information as reminder.

Near the destination you get stuck in a slowly moving tailback. The car offers you automated stop-and-go driving. This means that it is able to follow the vehicle in front of you longitudinally as well as laterally. This behavior is

not purely reactive. Traffic lights and signs are additionally taken into account by your intelligent stop-and-go system. Driving manually, the system is able to warn you if you have overlooked a red traffic light or a stop sign. Crosswalks are detected and pedestrians that intend to cross the road are recognized. Finally you reach your destination. The small parking lot is no longer a problem, since you can leave your car and let it park itself.

To some degree, nowadays these advanced intelligent systems for vehicles have been made possible thanks to latest automotive technologies including various sensors, actuators, processors, communication components and advanced algorithms. And rapidly falling cost for the sensors and processors combined with increasing sensor powers provide the basis for a continuous growth of the vehicle's intelligence. This trend have been seen in the automotive market over the last ten years with the appearance of advanced driving assistant applications to improve comfort and safety driving experiences. In 1998, Toyota became the first to introduce an ACC system on a production vehicle when it unveiled a laser-based system for its Progres compact luxury sedan, which it sold in Japan. Radar-based advance cruise control was commercialized by DaimlerChrysler (DC) in 1999 in their premium class vehicles. A vision-based lane departure warning system for heavy trucks was introduced by DC in 2000.

These comfort and safety applications mentioned above have to rely on the knowledge of the vehicle environment and therefore the environment perception task plays a very important role. The perception task itself is also one of the three fundamental components for any autonomous robotic system to work which includes perception, decision and control (see Figure 1.1). This figure implies that autonomous robots must be able to execute three basic tasks: i) perceiving, modeling of the environment, ii) reasoning, deciding which actions to take and

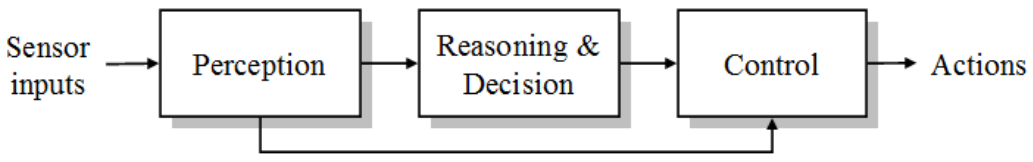


Figure 1.1: Robotics paradigm.

iii) finally, realizing them. For an intelligent vehicle system, such three essential tasks correspond to understanding the scene surrounding of the vehicle (position on the road, presence of obstacles, of road signs), deciding which actions to realize (trajectory and speed over time), and finally controlling the physical vehicle in order to follow the defined actions.

In this dissertation, we focus on studying the first one, the vehicle perception task. Nowadays, existing automotive systems usually consider their applications in simplified scenarios (e.g: on highways) or in simple environments (e.g: in parking places). **Here we propose to tackle this perception problem in a more general way which enables the vehicle to be able to operate in any populated environments such as crowded urban streets**. Emerging issues and state-of-the-art approaches to the problem are investigated. And we will present our approach with the ultimate objective of developing general algorithms to help building a multi-purpose perception system which could be used for a variety of automotive applications.

1.2 Vehicle Perception: Problem Statement

Since the environment around of the vehicle consists of stationary and/or moving objects, establishing the spatial and temporal relationships among the vehicle, stationary objects and moving objects in the scene serves as basic requirements for vehicle perception. This is reflected through three underlying tasks: localization, mapping and moving object tracking. *Localization* is the process of establishing the spatial relationships between the vehicle and stationary objects. *Mapping* is the process of establishing the spatial relationships among stationary objects. And *moving object tracking* is the process of establishing the spatial and temporal relationships between moving objects with the host vehicle and stationary objects. Besides the spatial and temporal information about surrounding objects, we also want to know what kind of moving objects they might be. Distinguishing pedestrians, bicycles or vehicles among other objects, this ability of *classifying objects* of the perception module will help understanding more of the driving situation in order to respond in an appropriate way, especially for safety applications.

In order to get these information from vehicle environment, the perception

module relies on different kinds of perception sensors, such as: camera, radar and laser. Each kind of sensor has advantages and also disadvantages compared with others. Camera has high information content, lower costs and operating power but lower signal-to-noise ratio of acquired images under poor lighting and bad weather making perception using vision alone extremely difficult in outdoor environments. Radars are more robust to weather conditions but give a poor geometric information about objects. Laser scanner provides the distance information to the nearest obstacle with measurements of high reliability and accuracy. With reasonably reduced prices, laser has been rapidly gaining its popularity for mobile robotic applications.

In this dissertation, we address problems of simultaneous localization and mapping (SLAM) with detection and tracking moving objects (DATMO) focusing on using a laser scanner as the main perception sensor of the vehicle. While SLAM provides the vehicle with a map of static parts of the environment as well as vehicle location in the map, DATMO allows the vehicle be aware of dynamic entities around, tracking them and predicting their future behaviors. We believe that a satisfactory solution to both SLAM and DATMO problems will open a vast range of potential for automotive applications.

Localization

When the vehicle is moving on the road, we need to know its relative position to the road as well as which lane it is occupied. A precise localization system is therefore essential. It is known that GPS and DGPS often fail in urban areas because of urban canyon effects, and good inertial measurement systems (IMS) are very expensive. If we can have a stationary object map in advance, the map-based localization techniques such as those proposed by [Olson 2000, Fox *et al.* 1999b] and [Dellaert *et al.* 1999] can be used to increase the accuracy of the pose estimate. Unfortunately, it is difficult to build an usable stationary object map a priori because of temporary stationary objects such as parked cars. Stationary object maps of the same scene built at different times could still be different, which means that we have to do online map building to update the current stationary object map.

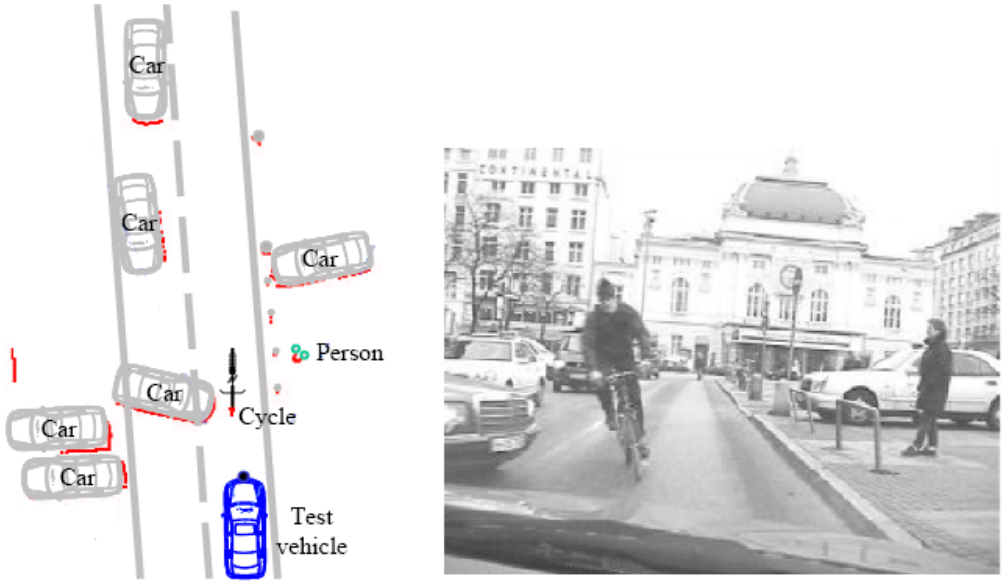


Figure 1.2: The ultimate goal of vehicle perception including fundamental tasks: Localization, mapping with detection, classification and tracking moving objects.

Mapping

In the literature, the mobile robot mapping problem is often referred to as the simultaneous localization and mapping problem. Simultaneous localization and mapping (SLAM) allows robots to operate in an unknown environment and then incrementally build a map of this environment and concurrently use this map to localize robots themselves. Over the last decade, the SLAM problem has attracted immense attention in the mobile robotics literature [Christensen 2002], and SLAM techniques are at the core of many successful robot systems [Thrun 2002]. However, [Wang & Thorpe 2002] have shown that SLAM can perform badly in crowded urban environments because of the static environment assumption. Moving objects have to be detected and filtered out.

Detection and Tracking of Moving Objects

The moving object tracking problem has been originated from radar tracking systems and extensively studied for several decades [Bar-Shalom & Fortman 1988, Blackman & Popoli 1999]. Many tracking works supposes that the measurements

correspond uniquely to moving objects and focus on the data association problem. However most of the real applications include spurious elements in the measures or presence of static objects. Obviously detecting correctly the moving objects is a critical aspect of a moving object tracking system.

Moving object detection in crowded urban environments is not easy because of a wide variety of targets. With cameras, feature-based or appearance-based recognition approaches [Viola & Jones 2001] can be used to detect moving objects. When laser are used, motion-based approaches are usually the preferred solutions since both appearance-based and feature-based methods rely on prior knowledge of the targets. In addition, the shape of moving objects seen by laser scanner can change significantly from scan to scan. As a result, it is hard to define features or appearances for detecting variety of objects with laser data.

After moving objects are identified, the multi-object tracking problem arises in order to estimate dynamic states of each object. It has to deal with the data association problem and maintenance of a list of objects currently present in the environment as well as their dynamics models. In general clutters, occlusions or mis-detections from the detector could cause more challenging situations to the data association step. In addition, changing motion behaviors of moving objects make defining a suitable motion model of the objects being tracked more difficult.

Object Classification

In urban areas, there are many kinds of moving objects such as pedestrians, bicycles, motorcycles, cars, buses, etc... Velocities range from under 3mph (such as a pedestrians movement) to 70mph. In the computer vision literature, algorithms for object classification have been attained in a mature stage where informative features like colors, shapes... could be used. However, so far very few research work have addressed to solve this task with laser sensors. Since all data returned by laser scanner are discrete points of impacts on moving objects, it is hard to define distinguished features to identify different kind of objects. Another clues could be useful with the use of estimated velocities of moving objects but it is not easy to define the appropriated range for each object class.

1.3 Contributions

To deal with SLAM and DATMO, we start by providing a theoretical framework to solve these problems and explain why SLAM and DATMO are mutually beneficial and should be solved together in context of the dynamic environments. Focusing on using laser scanner as the main perception sensor, we explore methods of representing vehicle environment as well as moving objects and their motion models. State-of-the-art approaches to SLAM and DATMO problems are then briefly reviewed in terms of these representation choices.

The first contribution of this research is made by a complete solution to solve both problems of SLAM and detection of moving objects which is based on occupancy grid to represent the vehicle map and free-form objects to represent moving entities. To correct vehicle location from odometry we introduce a new fast incremental scan matching method that works reliably in dynamic outdoor environments. After good vehicle location is estimated, the surrounding map is updated incrementally and moving objects are detected without a priori knowledge of the targets. Experimental results on datasets collected from different scenarios such as: urban streets, country roads and highways demonstrate the efficiency of the method.

The second contribution follows the first results after a good vehicle localization and a reliable map are obtained. We now focus on moving objects and present a method of simultaneous detection, classification and tracking moving objects. Moving objects are represented by pre-defined models of several object classes and this information is used to interpret the laser data sequence over a sliding window of time by hypotheses of moving object trajectories. Knowledge of various aspects including object model, measurement model, motion model are integrated in one theoretically sound Bayesian framework. The data-driven Markov chain Monte Carlo (DDMCMC) technique is used to solve the data association in the spatial-temporal space to effectively find the most likely solution. We test the proposed algorithm on real-life data of urban traffic and present promising results.

The third contribution is an integration of our perception module on a real vehicle for a particular automotive application, named Pre-Crash. This integration has been performed in the framework of the European Project PReVENT-ProFusion in cooperation with Daimler. The application objective is to recognize

unavoidable collisions with obstacles before they take place in order to trigger restraint systems. A comprehensive experimental evaluation based on relevant crash and non-crash scenarios is presented which confirms the robustness and reliability of our proposed method.

1.4 Thesis Outline

The organization of this dissertation is as follows: We review the state-of-the-art approaches to SLAM and DATMO problems in Chapter 2. In Chapter 3, we describe our grid-based solution to solve SLAM with moving object detection. In Chapter 4, we introduce a method of simultaneous detection, classification and tracking moving objects. An application of our perception module implemented on a real vehicle is presented in Chapter 5. Finally, we conclude with a summary of this work and suggest future extensions in Chapter 6.

Chapter 2

Vehicle Perception - State of The Art

2.1 Introduction

In this chapter, we study state-of-the-art approaches to the vehicle perception focusing on problems of localization and mapping with detection and tracking of moving objects. Within the context of perception in dynamic environments, whereas simultaneous localization and mapping (SLAM) deal with modeling static parts, detection and tracking moving objects (DATMO) are responsible for modeling dynamic parts of the environment. These tasks are tightly related to problems of how to represent the static environment and how to represent moving objects together with their dynamics models. The choice of representation methods is a very important step and will eventually decide the approaches to solve SLAM and DATMO problems. These methods are considered both in theoretical aspects (setting assumptions) and practical aspects (defining computational requirements and implementation complexity). In practice the representation choice 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.

In the following sections, approaches to SLAM and DATMO are briefly reviewed in terms of the environment representation choice. We will also provide a theoretical framework to solve these problems and explain why in dynamic environments, SLAM and DATMO are mutually beneficial and they should be solved together. Before going into detail, we introduce some notations and probabilistic backgrounds that will be frequently used later on in the dissertation

to solve the addressed problems.

Notations

The vehicle perception, as shown in Figure 2.1, can be treated as a process of taking inputs from sensor measurements including measurements from perception sensors such as laser scanners or cameras which is denoted by Z and measurements from motion sensors such as odometry or inertial measurement which is denoted by U. The process outputs include the estimated internal vehicle state X, a static map of the surrounding environment M and a list of moving objects in the vicinity of the vehicle O. The vehicle state is comprised of variables regarding the vehicle itself such as speed and its relative pose to the map M. The static map of vehicle environment M contains information about stationary objects as well as their locations in the map. And the moving object list O contains information about dynamic objects, their locations and dynamic states such as velocity and moving direction.

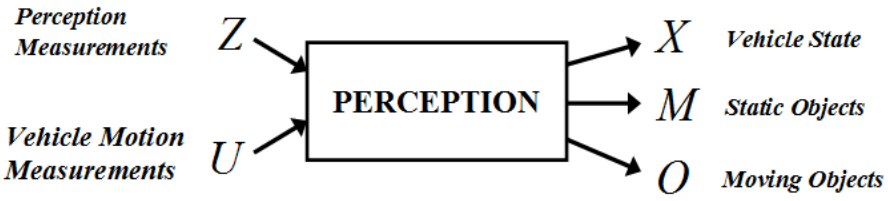


Figure 2.1: The general perception process. Z denotes the perception measurements, U denotes the motion measurements, X is the vehicle state, M is the map of stationary objects and O denotes the states of moving objects.

For states which tend to change over time, we use specific variables to indicate values of each state at certain time. For instance, x_t indicates the true state of the vehicle at time t. This allows to define the trajectory of the vehicle over time:

$$x_{0:t} = \{x_0, x_1, ..., x_t\} \tag{2.1}$$

As the vehicle moves, its state x_t evolves, the motion sensors allow to measure the control u_t of its displacement and the perception sensors allow to collect measurements of the environment z_t. In addition, we define the following set

to refer data leading up to time t:

$$Z_t = z_{0:t} = \{z_0, z_1, ..., z_t\} \tag{2.2}$$

$$U_t = u_{1:t} = \{u_1, u_2, ..., u_t\} \tag{2.3}$$

The static map M is denoted by a list of stationary objects in the environment:

$$M = \{m^1, m^2, ..., m^K\} \tag{2.4}$$

where K is the total number of objects in the environment, and each m^k with $1 \leq k \leq K$ specifies properties and location of the corresponding object.

The moving object list O_t at time t is composed of a finite set of N objects, where each o_t^n with $1 \leq n \leq N$ contains information about locations and dynamic states of each object at time t:

$$O_t = \{o_t^1, o_t^2, ..., o_t^N\} \tag{2.5}$$

Our objective here is to estimate states of the vehicle X, the static map M, the moving objects O given sensor measurements Z and U over time.

In the following we introduce a probabilistic method, the Bayesian Filters, a class of recursive algorithms for state estimation that forms the basis of virtually every techniques presented in this document.

Bayesian Filters

The task of estimating system states from sensor data is at the core of any robotic system. State estimation addresses the problem of estimating quantities from sensor data that are not directly observable, but that can be inferred. In most robotic applications, determining what to do is relatively easy if one only knew certain quantities. For example, moving a mobile robot is relatively easy if the exact location of the robot and all nearby obstacles are known. Unfortunately, these variables are not directly measurable. Instead, a robot has to rely on its sensors to gather this information. Sensors carry only partial information about those quantities, and their measurements are corrupted by noises. Probabilistic state estimation methods seek to recover state variables from the noisy data by computing belief distributions over possible states taking the uncertainty into

account.

Bayesian Filtering (sometimes known as Bayesian Sequential Estimation) [Anderson & Moore 1979] is a widely accepted probabilistic framework to the problem of estimating dynamic states of a system evolving in time given sequential observations or measurements about that system. The idea behind the Bayesian Filtering is that allows to use past and present measures in sequence to enhance the estimation of the actual system state.

Figure 2.2 presents a graphical model (or Bayesian network) that shows the evolution of a discrete system whose state at time t is denoted by s_t and the observations made from it denoted by y_t. The graph describes the relation between state variables following the first order Markov assumption that the current state depends only on the previous one.

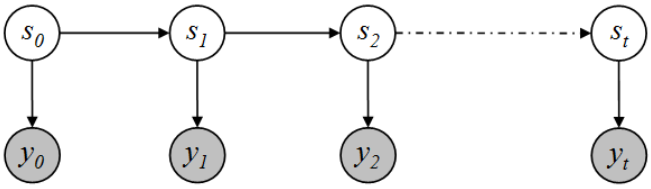


Figure 2.2: Graphical model representation of a simple Bayesian Filtering. Gray circles denotes observed states, clear circles denotes hidden states to be estimated.

The underlying task of the filter is to estimate the posterior probability $P(s_t|y_{0:t})$. An important property of the Bayesian filter is that this probability can be solved recursively using the Bayesian theorem:

$$\begin{aligned} \underbrace{P(s_t|y_{0:t})}_{\text{posterior at } t} &= \eta \; P(y_t|s_t)P(s_t|y_{0:t-1}) \\ &= \eta \; \underbrace{P(y_t|s_t)}_{\text{update}} \underbrace{\int_{s_{t-1}} P(s_t|s_{t-1}) \; \underbrace{P(s_{t-1}|y_{0:t-1})}_{\text{posterior at } t-1}}_{\text{prediction}} \end{aligned} \tag{2.6}$$

where η is the normalization constant. The recursive computation is initialized by the prior distribution $p(s_0|y_0) = p(s_0)$.

This algorithm can be interpreted as transformations over distributions of probability. Using the state transition function $P(s_t|s_{t-1})$ and the previously

estimated probability $P(s_{t-1}|y_{0:t-1})$, we obtain a distribution $P(s_t|y_{0:t-1})$ which is commonly called the prediction step. Then introducing the new measure we update the distribution with the likelihood $P(y_t|s_t)$ to obtain the desired result $P(s_t|y_{0:t})$. The process is illustrated in Figure 2.3.

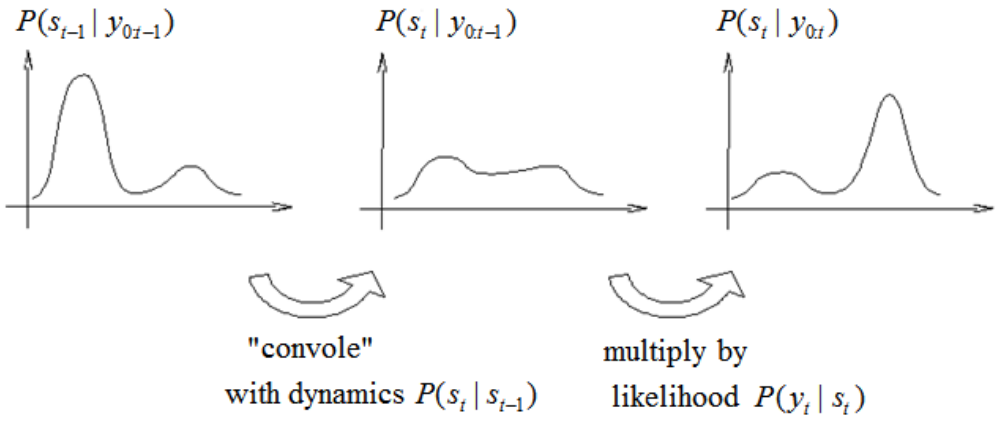


Figure 2.3: Sequential Bayesian Filtering.

It is clear that in order to compute the sequential Bayesian filter, definitions of the likelihood function (or measurement model) $P(y_t|s_t)$ and the state transition function (dynamics model) $P(s_t|s_{t-1})$ are required. And for any implementation also the description of the probability density functions have to bee chosen. Common choices are unimodal Gaussian, mixture of Gaussians and set of particles.

When the sensor model is Gaussian and the dynamics model is linear with Gaussian noise then the sequential Bayesian filtering algorithm leads to the well-known Kalman Filter [Kalman 1960]. The key idea behind the Kalman Filter is the remark that we stay in the ”Gaussian world” as long as we start with Gaussians and perform only linear transformations.

In order to handle the nonlinear and non-Gaussian situations, extensions have been made to the standard Kalman Filter. The Extended Kalman Filter (EKF) [Anderson & Moore 1979] is one of the extensions which uses a first order Taylor series expansion of the nonlinear functions for the approximation of the dynamics and likelihood model.

The Unscented Kalman Filter (UKF) [Julier & Uhlmann 1997] has even

higher accuracy. It approximates the posterior directly instead of approximating the nonlinear dynamics and likelihood functions. In particular, it uses a deterministically selected sample set and pass it through the true nonlinear function to capture the true distribution. UKF gains its popularity because of its capability of handling nonlinear models and its efficiency in computation.

To handle highly nonlinear and non-Gaussian models in Bayesian filtering, Particle filters [Arulampalam *et al.* 2002] are more accurate than Kalman-based methods because of its ability to handle highly nonlinear and non-Gaussian models with a clear and neat numerical approximation. The key idea is to approximate the posterior distribution with a set of randomly sampled particles that have weights associated to them. The more particles are, the better represented a probability function will be. Particle filtering is a very powerful method that can manage any distribution (notably multimodals) and any nonlinear function. Defining a reasonable number of particles and ensuring that sampling correctly the high likelihood regions is in general not well defined.

We note that the system used for the Bayesian filtering example presented above (Figure 2.3) is just a simple case with one state variable and one observation variable. In general, the system can have more than one observation at a time or some applications can have measures depending of two states (as the odometry does) or access to inputs of the system. However, all those cases we can solve with the same methodology to the simple case.

In the following sections, we will present formulations of the SLAM and DATMO problems as Bayesian filter processes.

2.2 Simultaneous Localization and Mapping

Simultaneous localization and mapping is commonly abbreviated as SLAM and is also known as the *concurrent mapping and localization* problem. SLAM is actually a chicken-and-egg problem. Vehicle location and map are both unknown. When the vehicle moves, it accumulates errors in odometry, making it gradually less certain as to where it is. For building an accurate map of the environment, we need to know correct poses of the vehicle. But to estimate the correct vehicle poses, we need to localize the vehicle in the map which requires that an accurate map of the environment is available. The term *simultaneous localization and*

mapping describes the resulting problem: In SLAM, the vehicle acquires a map of the environment while simultaneously localizing itself to this map given all measurements from odometry and perception sensors.

In the probabilistic form, the SLAM problem involves estimating the probability distribution:

$$P(x_t, M|z_{0:t}, u_{1:t}) \tag{2.7}$$

This probability distribution describes the *joint* posterior density of the map and vehicle state at time t given the measurements and control inputs up to time t. In general, since data arrives over time, a recursive solution to SLAM is desirable.

Starting with an estimate for the distribution $P(x_{t-1}, M|z_{0:t-1}, u_{1:t-1})$ at time $(t-1)$, the joint posterior, following a control u_t and measurement z_t, is estimated using a Bayes filtering process:

$$\underbrace{P(x_t, M|z_{0:t}, u_{1:t})}_{\text{postertior at } t} \propto \underbrace{P(z_t|x_t, M)}_{\text{update}} \underbrace{\int_{x_{t-1}} P(x_t|x_{t-1}, u_t) \underbrace{P(x_{t-1}, M|z_{0:t-1}, u_{1:t-1})}_{\text{posterior at } t-1}}_{\text{prediction}} \tag{2.8}$$

For this computation, we need to specify two probabilities: the sensor measurement model $P(z_t \mid x_t, M)$ and the vehicle motion model $P(x_t \mid x_{t-1}, u_t)$. The sensor model describes the probability of making an observation z_t when the vehicle state and a map of environment is known. The motion model describes the probability distribution on vehicle state transition assumed to be a Markov process of first order in which the next state x_t depends only on the immediately proceeding state x_{t-1} and the applied control u_t, and is independent of both the observations and the map. The graphical model in Figure 2.4 explains the dependency structure of variables in the SLAM problem.

In the literature, solutions to the probabilistic SLAM can be roughly classified according to methods to represent the map M and the underlying technique to estimate the posterior (2.8) which involves finding an appropriate representation for the measurement model and motion model. Before describing state-of-the-art SLAM algorithms, we will review some popular methods to represent the map of environment.

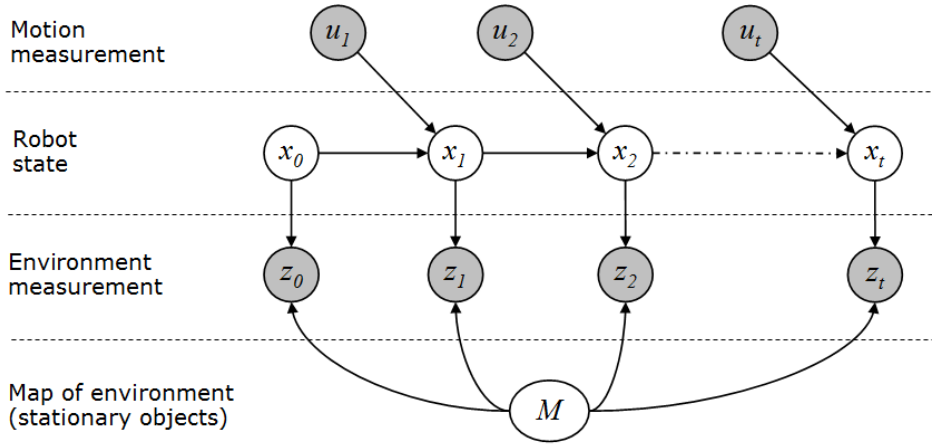


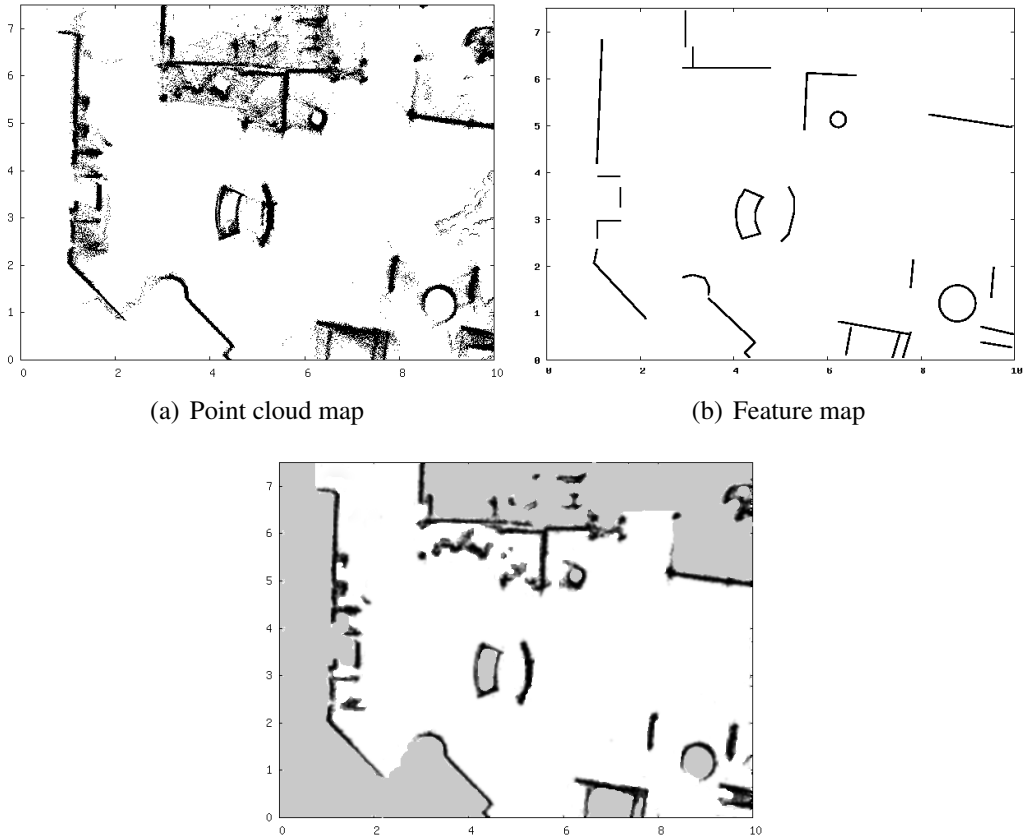
Figure 2.4: Graphical model of the SLAM problem as filtering process. Gray circles denotes observed states, clear circles denotes hidden states to be estimated.

2.2.1 Map Representation

The choice of map representation is an important step when dealing with the perception problem. Popular methods for representing maps of the environment include: feature-based approach [Leonard & Durrant-Whyte 1991], grid-based approach [Elfes 1989], direct approach [Lu & Milios 1997a], and topological approach [Choset & Nagatani 2001]. Because topological maps are usually generated on top of grid-based or feature-based maps by partitioning grid-based or feature-based maps into coherent regions, we will only focus on direct, feature-based and grid-based approaches.

Direct Representation

Direct method is often introduced when range sensors like laser scanners are used. This method is using raw data measurements to represent the physical environment without extracting predefined features. In the laser case, each laser scan measurement is a set of points which are impacts of laser beams with obstacles. A map can be constructed by simply aggregating measured points leading to a *point cloud* map representation (Figure 2.5(a)).



(a) Point cloud map

(b) Feature map

(c) Grid map

Figure 2.5: Example of different representation methods of the map built from the same set of laser data measurements.

Feature-based Representation

Feature-based approaches compress measurement data into predefined features which can be geometric primitives like points, lines, circles, etc,... Mapping then amounts to estimating the parameters of the primitives as to best fit the observations. To detect geometric features, among popular methods we can name some notable ones: the split-and-merge method [Einsele & Farber 1997] for detecting line segments, the Hough-transform [Pfister *et al.* 2003] or RANSAC [Forsyth & Ponce 2002] methods for detecting lines or circles. An example of a two-dimensional geometric feature map is presented in the image Figure 2.5(b).

In terms of spatial information content, feature maps are limited to parametric landmarks or modeled objects. The geometric representation does suffer from

not being able to represent more complex environments, such as the space in between the features, and natural structures. Furthermore they usually are only approximation of natural structures.

Grid-based Representation

Evidence grids or occupancy grids were first introduced by Elfes [Elfes 1989]. In this representation, the environment is subdivided into a regular array or a grid of rectangular cells. The resolution of the environment representation directly depends on the size of the cells. In addition to this discretization of space, a probabilistic measure of occupancy is estimated for each cell of the grid which indicate that cell is occupied by an obstacle or not. To update the state of occupancy for each cell of the grid when data coming, in the literature many methods have been introduced including Bayesian filtering [Elfes 1992, Thrun *et al.* 2005], Dempster-Shafer theory [Gambino *et al.* 1997, Pagac *et al.* 1998] and Fuzzy Logic [Oriolo *et al.* 1997].

An example of an occupancy grid map representation is shown in Figure 2.5(c), where white regions correspond to free cells, and black regions to occupied cells. The evidence grid is an efficient approach for representing uncertainty and for fusing multiple sensor measurements. It is also ideal for incorporating different models of sensor uncertainty.

Comparison between map representations

Direct approach, despite of being the simplest, can represent any kind of environments. However, its disadvantage lies in the important memory usage and the lack of a precise representation of the uncertainty in sensor measurements. Feature-based maps are attractive because of their compactness. However, concerning the environment representability, they are limited to indoor or structured environments where features are easy to define and extract. Whereas grid-based approaches typically require a huge amount of memory, but they are able to represent arbitrary features and provide detailed representations. Regarding sensor characteristics, grid-based approaches are the easiest to implement and the most suitable for range sensors such as sonar and laser. One more advantage of the grid-based approach over the other two is that it takes the sensor characteristics into account so that it can explicitly model the free space which provides useful

information for robot navigation applications. We summarize by a comparison of different methods of map representation in Table 2.1.

Table 2.1: Comparison of map representation methods.

Representations	Raw data	Feature-based	Grid-based
Data compression	-	+	-
Environment representability	+	-	+
Uncertainty management	-	+	+
Sensor characteristics	-	-	+

Because of advantages over others, nowadays, occupancy grid have become the most common choice among map representation methods, particular for applications in outdoor environments [Wang *et al.* 2003, Vu *et al.* 2007, Montemerlo *et al.* 2008]. To overcome its drawback on memory consuming when mapping large environments (i.e. city-sized), we can take an approach similar to that proposed by [Wang 2004]. The idea is that since the sensor range is limited, we only need to construct the grid map locally to avoid mapping non-measured regions. The local maps are then concatenated to represent the entire global map.

2.2.2 Kalman Filter SLAM

Knowing how to represent the map, it now remains to find appropriate representations for the measurement model and the motion model to estimate the posterior in the equation (2.8). By far the most common representation for these models is a linear function with additive Gaussian noise, leading to the use of the Kalman filter based approach to solve the SLAM problem.

Maps in the Kalman filter approach are commonly represented by a set of features. Appropriate features may be landmarks, distinctive objects or geometric shapes in the environment. Denoting the number of features in the map by K and the joint state of vehicle pose and map by $\mathbf{x}_t = (x_t, M)^T$, the posterior in (2.8) is represented by a Gaussian:

$$P(x_t, M|z_{0:t}, u_{1:t}) = P(\mathbf{x}_t|z_{0:t}, u_{1:t}) \sim N(\mu_t, \Sigma_t) \tag{2.9}$$

with a mean μ_t and a covariance matrix Σ_t. We can rewrite (2.8) as follows:

$$P(\mathbf{x}_t|z_{0:t}, u_{1:t}) = P(z_t|\mathbf{x}_t) \int_{\mathbf{x}_{t-1}} P(\mathbf{x}_t|\mathbf{x}_{t-1}, u_t) P(\mathbf{x}_{t-1}|z_{0:t-1}, u_{1:t-1}) \quad (2.10)$$

Kalman filter mapping relies on three basic assumptions. First, the next state function (motion model) must be linear with added Gaussian noise. Second, the same characteristics must also apply to the measurement model. And third, the initial uncertainty must be Gaussian.

A linear state function is one where the vehicle pose x_t and the map M at time t depend *linearly* on the previous pose x_{t-1} and map M at time $(t-1)$, and also linearly on the motion u_t. For the map, this is trivially the case since by assumption, the map does not change. However, the pose x_t is usually governed by a nonlinear trigonometric function that depends nonlinearly on the previous pose x_{t-1} and the control u_t. To accommodate such nonlinearities, Kalman filters approximate the vehicle motion model using a linear function obtained via Taylor series expansion. The resulting Kalman filter is known as *extended Kalman filter* (EKF). The result of the linearization is that the state transition function can be written as a linear function with added Gaussian noise:

$$P(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}, u_t) = A\mathbf{x}_{t-1} + Bu_t + \varepsilon_{motion} \quad (2.11)$$

where A and B are matrices that implement linear mapping from state $\mathbf{x}_{t-1}$ and the motion command u_t to the next state variable $\mathbf{x}_t$, respectively. Noise in motion is modeled via the variable ε_{motion}, which is assumed to be normal distribution with zero mean and the covariance Σ_{motion}.

As with vehicle motion, sensor measurements are usually nonlinear, with non-Gaussian noises. Thus they are also approximated through a first degree Taylor series expansion. Put into equations, Kalman filter method requires that the measurement model $P(z_t \,|\, \mathbf{x}_t)$ is of the following form:

$$P(z_t \,|\, \mathbf{x}_t) = C\mathbf{x}_t + \varepsilon_{measure} \quad (2.12)$$

where C is a matrix (a linear mapping), and $\varepsilon_{measure}$ is normal distributed measurement noise with zero mean and covariance $\Sigma_{measure}$.

Under the linearity and Gaussian noise approximation, the Bayes filter in

(2.10) can be calculated conveniently using the standard Kalman filter equations:

$$\begin{aligned}
\mu_{t-1}^{'} &= \mu_{t-1} + Bu_t \\
\Sigma_{t-1}^{'} &= \Sigma_{t-1} + \Sigma_{motion} \\
K_t &= \Sigma_{t-1}^{'} C^T (C\Sigma_{t-1}^{'} C^T + \Sigma_{measure})^{-1} \\
\mu_t &= \mu_{t-1}^{'} + K_t(z_t - C\mu_{t-1}^{'}) \\
\Sigma_t &= (I - K_t C)\Sigma_{t-1}^{'}
\end{aligned} \tag{2.13}$$

KF-SLAM can be implemented in $O(K^2)$ time, where K is the number of features in the map (the most costly operations in updating the Kalman filter are matrix multiplications). In practice, the number of features is not known a priori. State-of-the-art implementations often grow this list dynamically. To do so, they maintain a list of candidate features, using a separate Kalman filter for these candidates. If a feature is observed sufficiently often, it is permanently added to the list of features in the map. Outliers, that is, measurements that do not correspond to any known feature with sufficient likelihood, are usually ignored. These techniques work particularly well when features are scarce in the environment.

2.2.3 Maximum Likelihood SLAM

Another popular approach to SLAM problem is the *incremental maximum likelihood* method. Unlike Kalman filter methods which try to perform a full posterior estimation over the vehicle pose and map, its idea is to incrementally build a single map as the sensor data arrives without keeping track of any residual uncertainty. The advantage of this paradigm lies in its simplicity which accounts for its popularity.

Mathematically, the basic idea is to maintain a series of maximum likelihood maps, $\hat{M}_1, \hat{M}_2, ...$ along with a series of maximum likelihood vehicle poses $\hat{x_1}, \hat{x_2}, ...$ The t-th map and pose are constructed from the $(t-1)$-th map and pose via maximization of the marginal likelihood:

$$\langle \hat{x}_t, \hat{M}_t \rangle = \underset{x_t, M_t}{\operatorname{argmax}} \{ P(z_t | x_t, M_t)\, P(x_t, M_t | u_t, \hat{x}_{t-1}, \hat{M}_{t-1}) \} \tag{2.14}$$

This equation directly follows from (2.8) under the assumption that the $(t-1)$-th map and vehicle pose are known. Since the map M_t is usually uniquely determined once the pose x_t is known, in practice, it usually suffices to search in the space of poses x_t. Thus, the incremental Maximum Likelihood method simply requires searching in the space of all poses x_t when a new data item arrives, to determine the pose $\hat{x}_t$ that maximizes:

$$\hat{x}_t = \underset{x_t}{\operatorname{argmax}}\{P(z_t|x_t, \hat{M}_{t-1})\, P(x_t|u_t, \hat{x}_{t-1})\} \tag{2.15}$$

In this equation, the term $P(z_t|x_t, \hat{M}_{t-1})$ is the probability of observing the most recent measurement z_t given the pose x_t and the map $\hat{M}_{t-1}$ constructed so far. The term $P(x_t|u_t, \hat{x}_{t-1})$ represents the probability that the vehicle is at location x_t given that previously it was at position $\hat{x}_{t-1}$ and has carried out (or measured) the motion u_t. The resulting search of $\hat{x}_t$ is then appended to the map along with the corresponding scan z_t:

$$\hat{M}_t = \hat{M}_{t-1} \cup \{\langle \hat{x}_t, z_t \rangle\} \tag{2.16}$$

Maximizing (2.15) is equivalent to finding the vehicle pose x_t satisfying the vehicle motion model under which the measurement z_t is best fit to the given map M_{t-1}. In the literature, we often coin the term *scan matching SLAM* to this maximum likelihood SLAM approach. Depending on the map representations (Section 2.2.1), we have corresponding scan matching methods as seen in the literature: direct [Lu & Milios 1997b], feature-based [Ramos *et al.* 2007] or grid-based [Thrun *et al.* 2000].

The most popular direct scan matching method usually follows the Iterative Closest Point (ICP) algorithm [Lu & Milios 1997b] which compute the transformation (translation, rotation) between two raw scans in an iterative manner until the corresponding condition converges. They often assume an initial estimation of the relative pose of the scans is provided by the vehicle odometry.

Like Kalman filter SLAM, this approach can build maps in real-time, but without maintaining a notion of uncertainty. Moreover, the incremental likelihood maximization is only one-step but not over an entire data set. In particular, once a pose $\hat{x}_t$ and a map $\hat{M}_t$ have been determined, they are frozen once and forever and cannot be revised based on future data, a key feature of the Kalman-based SLAM. This weakness leads to the inability to map cyclic environments where the error in

the poses $\hat{x}_t$ may grow without bounds. Recently, Häehnel [Hähnel *et al.* 2003b] proposed an approach which is able to track several map hypotheses using an association tree. However the necessary expansions of this tree can prevent the approach from being feasible for real time operation.

2.2.4 FastSLAM

An alternative SLAM approach is FastSLAM [Montemerlo *et al.* 2002] which is set to overcome drawbacks of the KF-SLAM methods (linear and Gaussian model assumptions) and the ML-SLAM methods (cyclic mapping inability). To estimate the SLAM posterior (2.7), particle filter is used to represent non-linear models and non-Gaussian distributions. Since the high dimensional state-space of the SLAM problem makes direct application of particle filter computationally infeasible. The key idea of FastSLAM is to reduce the sample space by applying Rao-Blackwell theorem [Murphy 1999], whereby a joint state is partitioned according to the product rule $P(x_1, x_2) = P(x_2|x_1)P(x_1)$ and, if $P(x_2|x_1)$ can be represented analytically, only $P(x_1)$ need to be sampled.

The FastSLAM approach estimates the posterior probability of a joint state over the map M and the vehicle trajectory $x_{0:t}$ rather than the single pose x_t. This probability can be factored as:

$$P(x_{0:t}, M \mid z_{0:t}, u_{1:t}) = \underbrace{P(x_{0:t} \mid z_{0:t}, u_{1:t})}_{\text{estimate of trajectory}} \quad \underbrace{P(M \mid x_{0:t}, z_{0:t})}_{\text{map with known trajectory}} \tag{2.17}$$

Here, a particle filter is used to estimate the probability $P(x_{0:t}|z_{0:t}, u_{1:t})$ about potential trajectories $x_{0:t}$ of the vehicle given its observation $z_{0:t}$ and its odometry measurements $u_{1:t}$. When knowing the knowledge of vehicle trajectory $x_{0:t}$ and observations $z_{0:t}$, the probability $P(M|x_{0:t}, z_{0:t})$ over the map M becomes a problem of mapping with known poses and can be computed analytically. Briefly, in FastSLAM each particle represents a possible vehicle trajectory and a corresponding map making the computation in (2.17) efficiently.

For implementation, scan matching techniques can be used to make the sampling over possible trajectories more efficient [Hähnel *et al.* 2003d]. In this way, FastSLAM can be considered as running multiple ML-SLAMs where each particle is a separate ML-SLAM and basically can be implemented in $O(NK)$

times, where N is the number of particles and K is the map size. Figure 2.6 shows an example of FastSLAM with grid-based mapping. Samples of trajectories and the best map at a time are displayed. A number of potential trajectories are maintained which is reduced until the loop is detected.

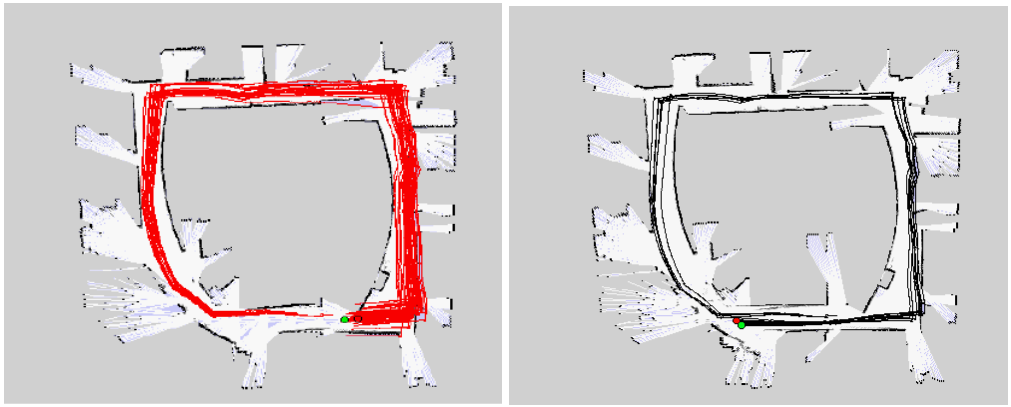


Figure 2.6: Loop closing in grid-based FastSLAM [Hähnel *et al.* 2003d].

2.2.5 Comparison of SLAM techniques

To summarize, we show the comparison of SLAM algorithms in Table 2.2.

Table 2.2: SLAM algorithm comparison

	KF-SLAM	ML-SLAM	FastSLAM
Map representations	- feature map -	raw map feature map grid map	- feature map grid map
Loop closing machenism	yes	no	yes
Computational complexity†	$O(K^2)$	$O(K)$	$O(NK)$‡
Memory complexity†	$O(K^2)$	$O(K)$	$O(NK)$

†For comparison purpose, we estimate the complexity of different SLAM methods with the same feature map representation.

‡With careful implementations, $O(NlogK)$ can be obtained [Montemerlo *et al.* 2002].

The beauty of the KF-SLAM approaches comes from the fact that they estimate a fully correlated posterior over feature maps and robot poses. Their weakness lies in the strong assumptions that have to be made on both the robot motion model and the sensor noise. In addition, EKF SLAM only works with feature maps. And it is not always easy to define and extract features in unstructured and outdoor environments.

Maximum Likelihood SLAM (ML-SLAM) is attractive because of its computational effectiveness and can be applied to any kind of map representations. However, in contrast to KF-SLAM, ML-SLAM only computes the most likely map at each time so that it is unable to close the loop in cyclic environments.

FastSLAM shares the fancy property with KF approach when it maintains the full posterior but is much faster compared to the classical KF-SLAM. FastSLAM can be considered as running multiple ML-SLAM which allows loop closure. It can be applied for feature-based and grid-based mapping so that it is also suitable for outdoor applications.

In practice, for applications where a consistent global map is required and a real-time performance is not necessary (ex: applications focusing on constructing accurate maps), FastSLAM is a better choice. However, for applications where only an instantaneous map is required (ex: obstacle avoidance applications), ML-SLAM is preferred because it can be computed very fast. To overcome its drawback when mapping cyclic environments, Wang [Wang 2004] proposed an interesting method which enables loop closing with ML-SLAM. The idea is that at a time it only needs to construct map locally using ML-SLAM and consider each local map as a feature. Then run a EKF-SLAM over all features to find relative position between local maps to generate the global map. Here we follow this idea to perform mapping and present our implementation latter in Chapter 3 of the thesis.

2.3 Detection and Tracking Moving Objects

While SLAM as described previously are responsible for modeling static part of the environment, DATMO deals with dynamic part of the environment. Its objective is to detect and track moving objects which enables the prediction of their future behaviors. Many tracking works suppose that the measurements

correspond uniquely to moving objects and then focus on data association problems. However most of the real applications include spurious elements in the measurements or presence of static objects. Radar data has ground noise (climatic perturbations, floor of the sea), video images have non stable backgrounds (trees on the wind, changing light conditions, moving camera), laser data includes non moving targets or spurious ground measures. Obviously detecting correctly moving objects is a critical aspect of a moving object tracking system. It is also a very important step for SLAM since separating moving objects from static objects is a key point in order to build accurate maps in highly dynamic environments (e.g. urban streets).

2.3.1 Moving Object Detection

Here we discuss different methods developed to detect moving objects with a particular use of laser sensor. In computer vision domain, moving object detection algorithms can be classified as appearance-based, feature-based, motion-based and model-based methods. Compared with images, laser data has less information so that the appearance-based approaches are not directly applicable.

Schulz in [Schulz *et al.* 2001] presented a method using simple features to detect people in office environments. Indoor people can be recognized by detecting local minimal in the laser range scans. This method can be useful on restricted ambient, but it is clearly not well suited for outdoor conditions where a tree can be similar to a pedestrian. Moreover, it is difficult to extend this method to detect other object classes rather than people.

Wang [Wang *et al.* 2003] presented an algorithm using occupancy grid for moving object detection from a moving ground vehicle. His method borrows idea from background-subtraction methods in computer vision. He proposed to construct an occupancy grid map incrementally from laser measurements that can be considered as a background modeling process. And based on the constructed grid map, we are able to identify moving objects when new measurements arrive: If an object is seen in a location previously observed as free space, the object is moving; if free space is observed in a location previously occupied by an object, then that object was moving. This general algorithm can be applied in any kind of environments and can be used to detect any kind of objects. One drawback of this approach is when an object appears in a previously not observed location, then we

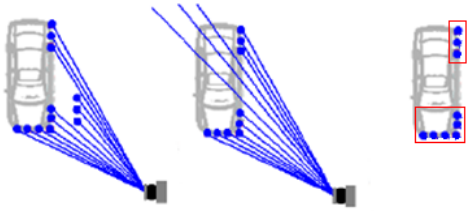
can say if it is moving or not. A priori we can suppose that new objects are static until evidence demonstrates the opposite.

Wolf [Wolf & Sukhatme 2005] proposed a grid-based alternative method. Besides constructing a static object grid map as Wang's method, he maintained a dynamic object grid map which is used to store regions previously identified dynamic objects. The core idea to detect moving objects from a new laser measurement is to put in relation the static objects maps, the dynamic maps, the non observed areas and the new measurement. Such relations are specified in the work of Wolf and permits to see the underlying logic of Wang's method as a particular case. This approach overcomes the drawback of Wang's method which can detect objects in unobserved areas. However, it is unclear how to extract objects from the dynamic grid map.

Häehnel [Hähnel *et al.* 2003c] introduced another approach using likelihood maximization. He defined a likelihood function that includes discrete terms that classify each measures as observing static or moving objects. Thus 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 his application to off-line post-processing. Also this method does not care about identifying specific objects or estimating their trajectories.

All methods mentioned above are model-free approaches which have an advantage that they can be used to detect moving objects of any kind without knowing a prior knowledge about that objects. However, as indicated by Petrovskaya [Petrovskaya & Thrun 2008] and Vu [Vu & Aycard 2009], these mentioned methods pose several problems in particular use with laser sensors. Firstly, due to partial occlusions or laser-absorbed object surfaces (ex: glassy or black surfaces), an object can be divided into several segments (Figure 2.7(a)). Secondly, only parts of the objects currently facing the sensor are visible, as the object moves it comes in different sizes that degrades the tracking results (Figure 2.7(b)). In these figures, we can see the importance of using a geometric vehicle model which allows to naturally handle the disjoint point clusters and the estimation of geometric shape of vehicles leads to more accurate tracking results.

Petrovskaya [Petrovskaya & Thrun 2008] proposed to use flexible models to detect moving vehicles. She introduced a method constructing a virtual grid in



(a) scans from vehicle are often split up into separate clusters due to occlusions or glassy surfaces.

(b) when moving, vehicle comes in different size (visible parts) which degrades the tracking result

Figure 2.7: Vehicle model can help better interpreting laser data.

polar coordinate from laser data and use a scan differencing technique to detect motion evidences. Then flexible rectangular models are fitted to these evidences and vehicle sizes can be learned adaptively after several observations. This method can detect vehicles successfully but does not model and detect pedestrians, bicyclists or motorcyclists which is a prerequisite for driving in populated areas.

In this thesis, we introduce another model-based approach as described in [Vu & Aycard 2009]. We define fixed models to represent several typical moving object classes and introduce a method to perform both moving object detection and tracking which is able to detect and classify buses, cars, motor/bi-cyclists and pedestrians. This method will be detailed in Chapter 4 of the document.

2.3.2 Tracking of Moving Objects

Once moving objects are detected and located, it is desirable to track them in order to estimate their dynamic states. Object tracking allows to aggregate object observations over time in order to enhance the estimation of dynamic states. The state vector can include position, speed, acceleration, geometric description, a classification of object, etc... Usually these state variables can not be observed or

measured directly, but they can be estimated through tracking process.

In general, the problem of tracking multiple objects consists of two parts: *Filtering* and *Data Association* [Bar-Shalom & Fortman 1988]. Filtering methods deal with the problem of tracking one specific object which consists in estimating its state from given observations over time. In the case of tracking multiple objects, data association consists in identifying which observation corresponds to which object being tracked, then filtering techniques are applied to estimate object states with known observations. In the following we will discuss the popular methods of filtering and data association in tracking.

Filtering - Multiple Dynamics Model

When the dynamics of a mobile object can be represented by a single model, we can apply directly Bayesian filtering methods previously mentioned in Section 2.1, such as the Kalman filter, Extended KF, Unscented KF or Particle filter, etc... to estimate object dynamic states.

In practice, however, object can change their dynamics behaviors over time (e.g.: stopped, moving, accelerating, etc...). To adapt to these changing behaviors, a multiple dynamics model is required. At a given time, besides estimating object dynamic states we also have to estimate its corresponding motion modes. While dynamic states are continuous variables, motion modes are discrete variables. These hybrid models are sometimes called switching mode models.

The general formula for the tracking problem of one moving object can be formalized in the probabilistic form as:

$$P(s_t, \mu_t | y_{0:t}) \tag{2.18}$$

where s_t is the true state of the object, and μ_t is the true motion mode of the moving object at time t (defined beforehand which could be of constant velocity, constant acceleration, turning, etc...) Figure 2.8 shows a graphical model representing a multiple model approach for solving object tracking problem.

Unfortunately the state estimate for the hybrid model can not 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.

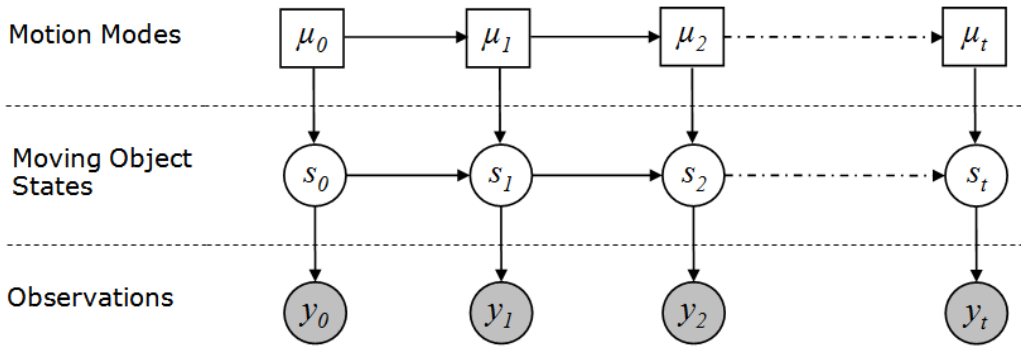


Figure 2.8: Graphical model representation for multiple model object tracking. Clear circles denote continuous states, squares denote discrete states.

If the model supposes linear models and Gaussian noise then the exact result for $P(s_t, \mu_t|y_{0:t})$ 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 [Tugnait 1982]. 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 in the history tail.

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 and so on.

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.

The most commonly used method, named Interacting Multiple Model (IMM) [Blom & Bar-Shalom 1988] which provides a trade-off between GPB1 and GPB2. It only compute k Gaussians as in GPB1 but it still having as output an mixture of k Gaussians as in GPB2. IMM approach performs significantly better than the GPB1 algorithm and almost as well as the GPB2 algorithm in practice.

One important point in designing multiple dynamics model filters that there exists a trade-off between the model complexity and the accuracy of the estimation. The more precise our model is, the better it will describe the behavior of

the moving object and then it will be possible to estimate his future trajectory more accurately. If the model is more complex, it will include more parameters to estimate and take more time to calculate online.

Data Association

Data association arises from the task of multi-target tracking given observations about objects over time (returned by the detector for example). The objective is to work out which observation was generated by which target. Because of the ambiguity of sensor measurements, the data association problem in multi-target tracking becomes more complicated. Actually the number of observations do not necessarily correspond to the number of objects. And the number of objects is difficult to estimate since one object might be temporarily occluded or unobserved simply because objects can enter or go out of ranges of vehicle sensors. Moreover, the perception sensors or the object detection process might generate false alarm measurements.

The data association for multi-target tracking consists in deducing the number of true objects and identifying if each observation corresponds to an already known object being tracked, to a spurious measure or to a new object in the scene that will be started to be tracked. The complexity to solve data association grow exponentially with the number of targets in the scene. Figure 2.9 shows an example of data association given object observations over five time steps. The solution found is comprised of two tracks τ_1, τ_2 and a false alarm observation τ_0.

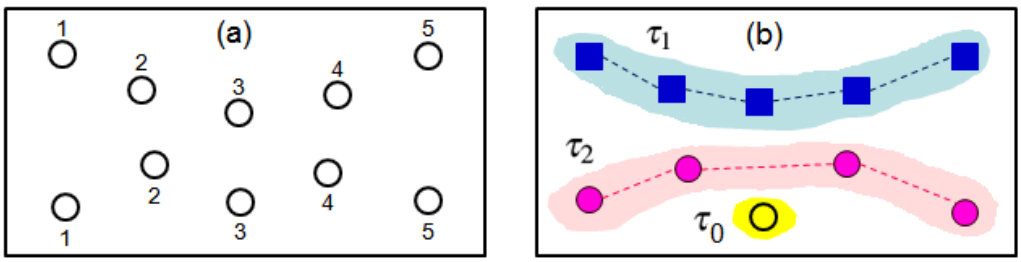


Figure 2.9: Example of data association. a) A set of observations Y (each circle represents an object observation together with numbers representing time steps. b) A solution of data association which is comprised of two tracks and a false alarm.

In the literature, data association algorithms are often categorized according to the objective function that they purport to optimize:

- *Heuristic* approaches typically involve optimizing associations between observations and targets under an explicit objective function.
- *Maximum a posteriori (MAP)* approaches find the most probable association, given all observations returned so far, then estimate tracks with this found association.
- The *Bayesian* approaches generates optimal filtering predictions by summing over all possible associations, weighted by their probabilities.

Data association algorithms can also be categorized by the way in which they process the measurements:

- *Single-scan* algorithms estimate the current states of targets based on their previously computed tracks and the current scan of measurements.
- *Multi-scan* algorithms may revisit past scans when processing each new scan, and can thereby revise previous association decisions in the light of new evidences.

The simplest data association method using a heuristic approach is the Greedy Nearest Neighbor (GNN) [Blackman & Popoli 1999]. It processes the new observations in some order and associates each with the target whose predicted position is closest, thereby selecting a single association after each scan. The method requires very little computation and is extremely fast. One drawback is its inability of correcting error associations at later steps.

MAP approaches includes the well-known multiple hypothesis tracking (MHT) algorithm [Reid 1979]. MHT is a multi-scan association algorithm that maintains multiple hypotheses associating past observations with targets. When a new set of observations arrives, a new set of hypotheses is formed from each previous hypothesis. The algorithm returns a hypothesis with the highest posterior as a solution. MHT is categorized as a ”deferred logic” method in which the decision about forming a new track or removing an existing track is delayed until enough measurements are collected. The main disadvantage of MHT in its pure form is its computational complexity since the number of hypotheses grows exponentially over time. Various heuristic methods have been developped to control this growth

[Blackman 2004] but these methods are applied at the expense of sacrificing the MAP property. However, since the underlying MAP data association problem is NP-hard, so we do not expect to find efficient, exact algorithm.

Exact Bayesian data association is even less tractable than the MAP computation. Several "pseudo-Bayesian" methods have been proposed, of which the best-known is the joint probabilistic data association (JPDA) filter. JPDA is well described in [Bar-Shalom & Fortman 1988] which is a suboptimal single-scan approximation to the optimal Bayesian filter. In its original form, JPDA assume the number of targets is fixed. However, it can be modified to track with varied number of objects [Schulz *et al.* 2001]. At each time step, instead of finding a single best association between observations and tracks, JPDA enumerates all possible associations (NP-hard) and computes association probabilities $\{\beta_{jk}\}$, where β_{jk} is the probability that j-th observation associates with the k-th track. Given an association, the state of a target is estimated by a filtering algorithm and this conditional state estimate is weighted by the association probability. Then the state of a target is estimated by summing over the weighted conditional estimates. JPDA has proved more efficient in cluttered environments compared with GNN [Bar-Shalom & Fortman 1988] but prone to make erroneous decision since only single scan is considered and the association made in the past is not reversible.

Recently, sampling data association methods using Markov chain Monte Carlo (MCMC) have achieved notable success in vision tracking [Song & Nevatia 2005, Yu *et al.* 2006, Zhao *et al.* 2008]. The idea behind these methods is to use MCMC sampling instead of enumerating over all possible associations. Unlike MHT and JPDA, MCMC data association (MCMCDA) is a true approximation scheme for the optimal Bayesian filter; *i.e.*, when run with unlimited resources, it converges to the Bayesian solution. In [Oh *et al.* 2004], Oh showed that a single-scan version MCMCDA can be designed to approximate JPDA in polynomial time and a multi-scan version MCMCDA can be designed to converge to the full Bayesian solution. This promising framework plus increased computational power of machines nowadays makes it possible for real-time tracking applications which also sees its contribution in Chapter 4 of this document.

2.4 SLAM with DATMO

In previous sections, we have briefly reviewed state-of-the-art approaches to solve problems of SLAM and DATMO. As mentioned, both SLAM and DATMO are essential tasks of a perception system for vehicles in dynamic environments. Up to now we have been considered them separately.

We notice that, SLAM methods mentioned in Section 2.2, have implicitly omitted dynamic objects and supposed that all received measurements come from static parts of the environment (see Figure 2.4). Map building is then made without special considerations. However, in dynamic environments, the presence of moving objects will introduce noises during the localization process and adding spurious elements on the constructed maps. In this sense, DATMO with the ability of separating moving objects from static objects during SLAM process allows enhance the mapping results. On the other hand, being able to do SLAM while tracking moving objects allow to better estimate their global speeds and positions and thus to better estimate their trajectories. These relations means that SLAM and DATMO are mutually beneficial and their relationship is depicted in Figure 2.10.

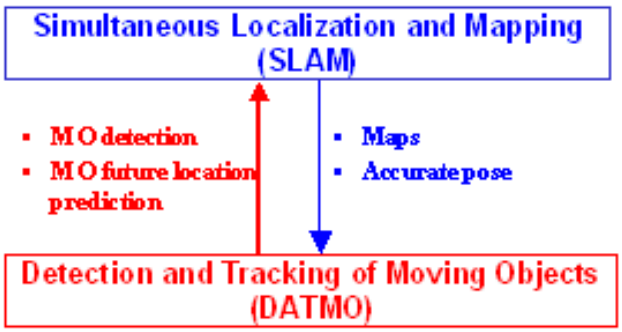


Figure 2.10: SLAM and DATMO are mutually beneficial.

Wang [Wang *et al.* 2003] was one of the first researchers pointing out the mutual relationship of SLAM and DATMO and showed that SLAM and DATMO should be solved together. In his pioneer work [Wang *et al.* 2003], he introduced a mathematical framework to solve SLAMMOT in dynamic environments in which SLAM is integrated with tracking generalized objects (both static and dynamic). The SLAMMOT problem can be represented by a joint posterior over states of

all objects need to estimate (ego-vehicle pose, stationary objects, dynamic objects and their dynamics modes) given all sensor measurements:

$$P(x_t, M, s_t, \mu_t | Z_t, U_t) \tag{2.19}$$

He showed that estimating (2.19) is computationally demanding and generally infeasible because of high dimension of the joint state variable. Overcoming this daunting task, he proposed to solve SLAM with DATMO instead which decomposes the SLAMMOT estimation problem into two separate estimators:

$$\underbrace{P(x_t, M, s_t, \mu_t | Z_t, U_t)}_{\text{SLAMMOT}} = \underbrace{P(x_t, M | Z_t^{(s)}, U_t)}_{\text{SLAM}} \underbrace{P(s_t, \mu_t | Z_t^{(d)})}_{\text{Moving Object Tracking}} \tag{2.20}$$

if in some way we are able to decompose the measurements Z_t into static and dynamic measurements:

$$\underbrace{Z_t = Z_t^{(s)} + Z_t^{(d)}}_{\text{Moving Object Detection}} \tag{2.21}$$

where $Z_t^{(s)}$ and $Z_t^{(d)}$ denote measurements corresponding to static objects and dynamic objects respectively. The decomposition in (2.21) corresponds to the moving object detection step in DATMO.

By such maintenance of separate posteriors for stationary objects and moving objects, the resulting estimation problem of SLAM with DATMO (2.20) are much lower dimensional than problem of direct estimating SLAMMOT in (2.19).

In the robotics literature, SLAM and DATMO have been attracted considerable research works. **As SLAM techniques in static environments are maturating, the research efforts are shifting to solve problems of SLAM in dynamic environments (or SLAM with moving object detection), and growing to deal with a more general SLAMMOT problem.** However, solutions to SLAMMOT are still at their early stage when most of works on SLAMMOT currently focus on indoor environments [Prassler *et al.* 1999, Hähnel *et al.* 2003a, Montesano *et al.* 2005] with simplified assumptions.

Wang [Wang *et al.* 2003] developed the first real-time outdoor perception system performing simultaneously SLAM with DATMO for urban environments from a ground moving vehicle. In his PhD thesis work [Wang 2004] he demonstrated that it is able to perform successfully a city-sized SLAM in urban

environments. However, for DATMO part, he applied a free-form approach for detecting and tracking of moving objects which see its disadvantages as mentioned in Section 2.3.

Recently, in the DARPA Urban Challenge competition [DARPA 2007], we have been witnessed significant advances in efforts of building autonomous vehicles. It is shown that driverless cars, for instance: Boss [Urmson *et al.* 2008] and Junior [Montemerlo *et al.* 2008], are capable of operating autonomously and safely through urban-alike environments. However, testing scenarios for the competition contains only vehicles as moving objects which limits their approaches to be only able to detect and track vehicles. In addition, to obtain a good performance, participant vehicles are equipped with so many precise and expensive sensors, such as 3D laser scanners, 2D laser scanners, radars, vision, precise inertial sensors... In this research, we emphasize the objective of building a reliable vehicle perception system using an affordable 2D laser scanner as the main perception sensor.

2.5 Summary

In this chapter, we have got an overview over state-of-the-art approaches to problems of SLAM and DATMO which are known to be at the core of any vehicle perception system in context of dynamic environments. We have showed that SLAM and DATMO are mutually beneficial and they should be solved together in order to get more robust and accurate results. It is pointed out that the process of moving object detection provides a bridge between SLAM and MOT which helps to reduce the complexity of the general SLAMMOT problem. We also review some notably related works to SLAMMOT problem in the literature.

Inspired by the pioneer work of Wang [Wang 2004], our objective in this dissertation is trying to put forward the state-of-the-art solutions to SLAM with DATMO in order to build a reliable vehicle perception system with affordable sensors (e.g. 2D laser scanner).

We summarize our approach as follows. To deal with SLAM in dynamic environments, we describe in Chapter 3 a grid-based algorithm to perform SLAM with detection of moving objects which is similar to the method proposed by Wang in [Wang 2004]. A maximum likelihood SLAM algorithm is applied for

mapping the vehicle environment. To correct vehicle location from odometry we introduce a new fast incremental scan matching method that works reliably in dynamic outdoor environments. Moving objects are detected as free-form objects using motion-based approach without a priori knowledge of the targets which are then filtered out to help building a more accurate map. After a good vehicle localization and a reliable map are obtained, we focus on moving objects and present a model-based method to perform simultaneous detection and tracking moving objects in Chapter 4. The Markov chain Monte Carlo (MCMC) data association technique is applied to explore in the spatio-temporal space to find the most probable trajectories of moving objects. We test these algorithms on real-life traffic data and demonstrate on a real vehicle to show its robustness and reliability.

The proposed approach will be presented in detail in following chapters.

Chapter 3

Grid-based SLAM with Detection of Moving Objects

3.1 Introduction

In this chapter we will present an algorithm to solve SLAM with moving object detection in dynamic environments from a ground moving vehicle equipped with a 2D laser scanner as the main perception sensor. Our approach here follows the work of Wang in [Wang 2004].

For the SLAM part, similar to Wang's work, we use a grid-based method to represent the vehicle environment. We employ a maximum likelihood SLAM approach (subsection 2.2.3) for mapping process thanks to its computational effectiveness. Due to large scales of the environment (e.g. city-sized), at a given time, only an online grid is maintained representing the local map surrounding of the vehicle. Instead of using an ICP-based method like in [Wang 2004] for vehicle localization, we introduce a new and fast grid-based scan matching method which does not need to find corresponding features and can work reliably in dynamic environments. When good vehicle locations are estimated, we are able to build a consistent local map of the vehicle environment incrementally when new measurements arrive. And then based on the constructed local grid map, moving objects can be detected when they enter object-free regions. This idea originated from the work of Wang which is simple but is shown to work quite well in practice. One important advantage of this approach is the fact that no model assumption is required to separate moving and stationary objects.

The chapter is organized as follows. In the next section 3.2, we describe the grid-based mapping process. Algorithm for detecting moving objects is presented in Section 3.3. Experimental results are reported in Section 3.4 and summary remarks are given in Section 3.5.

3.2 Grid-based SLAM

First introduced by Elfes [Elfes 1989], nowadays occupancy grids are the most popular method to represent maps of the environment. Compared with feature-based approaches, grid maps can represent any environment and are specially suitable for noisy sensors in outdoor environments where features are hard to define and extract. Grid-based approaches also provide an interesting mechanism to integrate different kinds of sensors in the same framework taking the inherent uncertainty of each sensor reading into account.

The process of grid-based mapping is to estimate the state of occupancy for each grid cell when new sensor data arrives. As we mentioned in subsection 2.2.1, many methods have been employed such as Bayesian filtering [Elfes 1992, Thrun *et al.* 2005], Dempster-Shafer theory [Gambino *et al.* 1997, Pagac *et al.* 1998], and Fuzzy Logic [Oriolo *et al.* 1997]. Here we utilize an algorithm to update the occupancy grid map using Bayesian method. In the next subsection, we will describe the mapping process with an assumption that the vehicle trajectories is known and will discuss how to perform simultaneous localization and mapping in the subsection followed.

3.2.1 Grid Mapping with Known Trajectories

We recall that in the grid-based representation (Section 2.2.1), vehicle environment is divided into a two-dimensional lattice M of rectangular cells and each cell is associated with a measure taking a real value in $[0, 1]$ indicating the probability that the cell is occupied by an obstacle or not. A high value of occupancy probability indicates the cell is occupied and a low value means the cell is free. Over time when data is received, we need to estimate the accumulated occupancy probabilities of the grid cells. Assuming that occupancy states of individual grid cells are independent, the objective of a mapping algorithm is to

estimate the posterior probability of occupancy $P(m \mid x_{1:t}, z_{1:t})$ for each cell m of the grid, given data measurements $z_{1:t} = \{z_1, ..., z_t\}$ which are acquired over time at known corresponding vehicle locations $x_{1:t} = \{x_1, ..., x_t\}$.

Using Bayes theorem, this probability is determined by:

$$P(m \mid x_{1:t}, z_{1:t}) = \frac{P(z_t \mid x_{1:t}, z_{1:t-1}, m) \,.\, P(m \mid x_{1:t}, z_{1:t-1})}{P(z_t \mid x_{1:t}, z_{1:t-1})} \tag{3.1}$$

If we assume that current measurement z_t is independent from $x_{1:t-1}$ and $z_{1:t-1}$ given we that know m, $P(z_t \mid x_{1:t}, z_{1:t-1}, m) = P(z_t \mid x_t, m)$. Then after applying Bayes theorem to $P(z_t \mid x_t, m)$, equation (3.1) becomes:

$$P(m \mid x_{1:t}, z_{1:t}) = \frac{P(m \mid x_t, z_t) \,.\, P(z_t \mid x_t) \,.\, P(m \mid x_{1:t}, z_{1:t-1})}{P(m) \,.\, P(z_t \mid x_{1:t}, z_{1:t-1})} \tag{3.2}$$

Equation (3.2) gives the probability for an occupied cell. By analogy, equation (3.3) gives the probability for a free cell:

$$P(\overline{m} \mid x_{1:t}, z_{1:t}) = \frac{P(\overline{m} \mid x_t, z_t) \,.\, P(z_t \mid x_t) \,.\, P(\overline{m} \mid x_{1:t}, z_{1:t-1})}{P(\overline{m}) \,.\, P(z_t \mid x_{1:t}, z_{1:t-1})} \tag{3.3}$$

By dividing equation (3.2) by (3.3), we obtain:

$$\frac{P(m \mid x_{1:t}, z_{1:t})}{P(\overline{m} \mid x_{1:t}, z_{1:t})} = \frac{P(m \mid x_t, z_t)}{P(\overline{m} \mid x_t, z_t)} \cdot \frac{P(\overline{m})}{P(m)} \cdot \frac{P(m \mid x_{1:t-1}, z_{1:t-1})}{P(\overline{m} \mid x_{1:t-1}, z_{1:t-1})} \tag{3.4}$$

If we define:

$$Odds(x) = \frac{P(x)}{P(\overline{x})} = \frac{P(x)}{1 - P(x)} \tag{3.5}$$

equation (3.4) turns into:

$$\begin{aligned} &Odds(m \mid x_{1:t}, z_{1:t}) \\ &\qquad = Odds(m \mid x_t, z_t) \,.\, Odds(m)^{-1} \,.\, Odds(m \mid x_{1:t-1}, z_{1:t-1}) \end{aligned} \tag{3.6}$$

The corresponding *log Odds* representation of equation (3.6) is:

$$\begin{aligned} &log\, Odds(m \mid x_{1:t}, z_{1:t}) \\ &\quad = log\, Odds(m \mid z_t, x_t) - log\, Odds(m) + log\, Odds(m \mid x_{1:t-1}, z_{1:t-1}) \end{aligned} \tag{3.7}$$

To incorporate a new scan into a given map we multiply its *Odds-ratio* with the *Odds-ratio* of a local map constructed from the most recent scan and divide it by the *Odds-ratio* of the prior. Often it is assumed that the prior probability of m is 0.5. In this case the prior can be canceled so that Equation (3.7) simplifies to:

$$\begin{aligned} &log\, Odds(m \,|\, x_{1:t}, z_{1:t}) \\ &\qquad = log\, Odds(m \,|\, z_t, x_t) + log\, Odds(m \,|\, x_{1:t-1}, z_{1:t-1}) \end{aligned} \tag{3.8}$$

To recover the occupancy probability from the Odds representation given in Equation (3.6) we use the following law which can easily be derived from Equation (3.5):

$$P(x) = \frac{Odds(x)}{1 + Odds(x)} \tag{3.9}$$

This leads to:

$$\begin{aligned} &P(m \,|\, x_{1:t}, z_{1:t}) \\ &\quad = \left[1 + \frac{1 - P(m \,|\, x_t, z_t)}{P(m \,|\, x_t, z_t)} \cdot \frac{P(m)}{1 - P(m)} \cdot \frac{1 - P(m \,|\, x_{1:t-1}, z_{1:t-1})}{P(m \,|\, x_{1:t-1}, z_{1:t-1})}\right]^{-1} \end{aligned} \tag{3.10}$$

This equation tells us how to update our belief about the occupancy probability of a grid map given sensor inputs. All we have to do to incorporate a new measurement z_t taken at location x_t is to multiply the *Odds-ratio* of the current belief about m with the *Odds-ratio* of the map constructed from the most recent measurement and divide the result by the prior probability of m.

Computation of the Occupancy Probability

In (3.10), $P(m \,|\, x_{1:t-1}, z_{1:t-1})$ is the occupancy probability value estimated previously from measurements in the past. What we need to know are two probability densities, $P(m \,|\, x_t, z_t)$ and $P(m)$. $P(m)$ is the prior occupancy probability of the grid map cell which is set to 0.5 representing an unknown state, that makes this component disappear. The remaining probability $P(m \,|\, x_t, z_t)$, is called the *inverse sensor model*. This probability is called "inverse" since it reasons from effects to causes: it provides information about the world conditioned on a measurement caused by this world. It specifies the probability

that a grid cell m is occupied based on a single sensor measurement z_t at location x_t.

Now we describe how we compute the inverse sensor model $P(m|x_t, z_t)$. According to Equation (3.10) a measurement z_t has no influence on a cell m if $P(m|x_t, z_t) = P(m)$. Therefore, cells in areas in which a measurement does not change the belief about the state of the world carry the value $P(m)$. Often $P(m) = 0.5$ is assumed to be the prior, but it can be regarded as a special instance of this situation. Our approach assumes that the occupancy probability of a cell m of the grid can be computed independently for all sensor measurements. Although this is a strong assumption, the resulting maps still are of satisfactory quality.

In our case, laser scanner is used as the main perception sensor. This sensor is very common in robotics and currently the state-of-the-art sensor for distance measurements. The signal of a laser-range finder is emitted in a beam and the sensor uses a rotating mirror to combine several distance measurements to a two dimensional scan. At each time t we receive a complete scan z_t, which is a combination of N distance measurements $z_t = \{z_t^n, 1 \leq n \leq N\}$. We assume that these distance measurements are independent and therefore consider the beams individually.

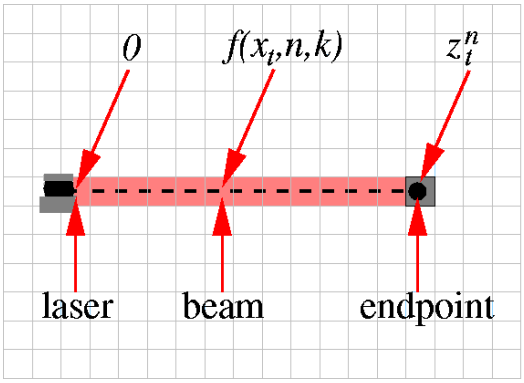
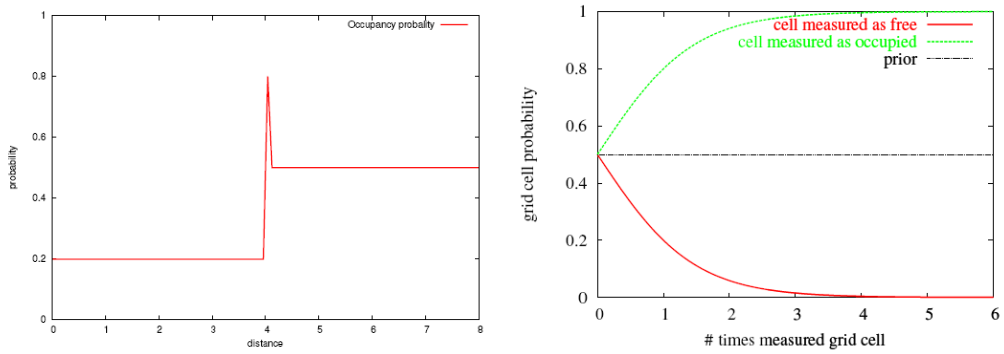


Figure 3.1: Laser beam measured at location x_t covering z_t^n cells of a map. The occupancy of the cell where the beam ends should be increased. On the other side the occupancy of the cells between the robot and the end-point, which are here marked with the color red, should be decreased.

Figure 3.2(a) shows the function we use to compute the occupancy probability of cells related to a laser beam measuring a specific distance (4m in this case). Let f be a function that returns for each pose x_t of the vehicle, each beam number n,



(a) Inverse laser sensor model (b) Updating occupancy probability of grid cell

Figure 3.2: a) shows the occupancy probability of cells along a beam measuring a distance of 4m. Between the sensor and the measured distance it is more likely to be free, at the measured distance we expect the grid cell to be occupied. Please note that because of the small error of the sensor this function has only a narrow peak. b) shows the occupancy probability of a grid cell seen several times as occupied (green) or unoccupied (red). We use the *Odds* model described in equation (3.10) to compute the probabilities. The prior of the map is 0.5.

and each $k \leq z_t^n$ the index $f(x_t, n, k)$ of k-th field covered by that beam in the map (see Figure 3.1). We can describe the function as follow:

$$P(m \mid x_t, z_t) = \begin{cases} P_{free} & : |f(x_t, n, k)| < z_t^n \\ P_{occ} & : |f(x_t, n, k)| = z_t^n \\ P_{prior} & : otherwise \end{cases} \tag{3.11}$$

where P_{prior} the prior of the grid map cells, P_{free} is the occupancy probability for free cells and P_{occ} is the occupancy probability for the occupied cells. In practice, we set $P_{prior} = 0.5$, $P_{free} = 0.2$, $P_{occ} = 0.8$.

We use the inverse sensor model in equation (3.11) to update occupancy probabilities of grid cells in equation (3.10). Figure 3.2(b) shows an example of the probability function that we see the grid cell several times as free or occupied.

In practice, the occupancy probability function can be implemented in an alternative way for a more convenient computation as follows. We count the number of times the cells is updated by P_{free} or P_{occ} and describe this number with c_{free} or respectively c_{occ}. Applying equation (3.10) we compute the probability

directly with:

$$f(c_{occ}, c_{free}) = \frac{exp(c_{occ}P_{occ} + c_{free}P_{free})}{1 + exp(c_{occ}P_{occ} + c_{free}P_{free})} \tag{3.12}$$

It is easy to see that the steepness of the function is related to the values of P_{occ} and P_{free}. Figure 3.2(b) shows, as a practical example, how fast the occupancy probability of a cell approximates to 0 or 1.

3.2.2 Grid-based Scan Matching

In the previous subsection, we described an incremental algorithm to update the occupancy grid map over time given laser scan measurements at known corresponding vehicle locations. In order to build a consistent map of the environment, a good vehicle localization is required. However, using only odometry provided by vehicle internal sensors often results in unsatisfied maps due to its inherent errors (see Figure 3.3 left). With the objective of correcting odometry errors whereby obtaining a good map, we would like to perform simultaneous localization and mapping using one of SLAM algorithms as introduced in Section 2.2. Here we opt for the incremental maximum likelihood SLAM method due to the advantage of its computational and memory complexity over other methods (subsection 2.2.3).

As presented in subsection 2.2.3, the incremental maximum likelihood SLAM approach consists in estimating a sequence of vehicle poses $\hat{x}_1, \hat{x}_2, ...$ and sequentially updated maps $\hat{M}_1, \hat{M}_2, ...$ by maximizing the marginal likelihood of the t-th pose and map relative to the $(t-1)$-th pose and map:

$$\hat{x}_t = \underset{x_t}{\text{argmax}}\{P(z_t|x_t, \hat{M}_{t-1}) \,.\, P(x_t|\hat{x}_{t-1}, u_t)\} \tag{3.13}$$

The resulting search of $\hat{x}_t$ is then used to generate a new map $\hat{M}_t$ along with the corresponding scan z_t via the incremental map updating process which is described in the last subsection (Equation (3.10)):

$$\hat{M}_t = \hat{M}_{t-1} \cup \{\langle \hat{x}_t, z_t \rangle\} \tag{3.14}$$

In the Equation (3.13), the term $P(x_t \,|\, \hat{x}_{t-1}, u_t)$ represents the vehicle motion

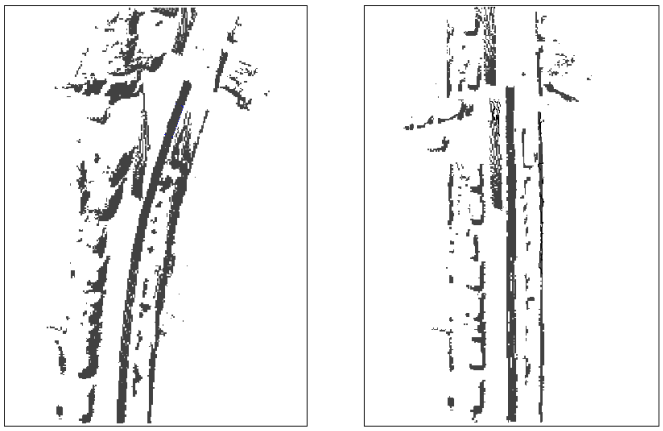


Figure 3.3: Maps built directly from raw laser data collected from a vehicle moving along a straight street: with vehicle localization using odometry (left); and using results of scan matching (right). Note that the scan matching results are not affected by moving objects in the street. See Figure 3.10 for the resulting occupancy grid map.

model which is the probability that the vehicle is at location x_t given that the vehicle was previously at position $\hat{x}_{t-1}$ and executed an action u_t. The term $P(z_t \mid x_t, \hat{M}_{t-1})$ is the measurement model which is the probability of receiving the most recent measurement z_t given the pose x_t and the map $\hat{M}_{t-1}$ constructed so far from measurements $\{z_1, ..., z_{t-1}\}$ at corresponding poses $\{\hat{x}_1, ..., \hat{x}_{t-1}\}$ that were already estimated in the past.

Maximizing (3.13) is equivalent to finding the vehicle pose x_t satisfying the vehicle motion model under which the measurement z_t is best fit to the given map M_{t-1}. In this meaning, we often coin the term *scan matching SLAM* to this maximum likelihood SLAM approach. Depending on the map representations (subsection 2.2.1), we have corresponding scan matching methods: direct, feature-based or grid-based.

In general cases where features in environments are difficult to be defined and extracted, direct scan matching techniques like Iterative Closest Point algorithm (ICP) [Lu & Milios 1997b] can be used. However, in these ICP-style scan matching methods, the measurement uncertainty is not taken into account. Especially, sparse data and dynamic entities in the environment cause problems of correspondence finding which affects the accuracy of matching results. Wang in [Wang 2004] proposed a sampling version of ICP which tried

to model the uncertainty of the matching results by running multiple ICP from different initial search poses but the expensive computation prevents it from a real-time operation. Other grid-based scan matching methods introduced by [Thrun *et al.* 2000, Hähnel *et al.* 2003a] again suffered from getting stuck in local minimum in a gradient search manner.

Here we introduce an alternative grid-based scan matching method to solve (3.13). In our approach, given an underlying vehicle dynamics constraint, the vehicle pose is estimated by correcting the correlation of the current laser scan with the local grid map constructed from all observations in the past instead of only with one previous scan. The advantage of our method is two folds. First, using grid-based correlation, measurement uncertainties are taken into account. Second, a trade-off between the vehicle dynamics model and a matching with the grid map make the localization results more robust. This is because in outdoor environments when measurements are quite sparse making scan matching difficult, we can temporarily rely on the vehicle dynamics until enough measurements are collected to correct the vehicle pose. The proposed method is presented in the following.

At first we describe how we represent the motion model and the measurement model in (3.13).

For the motion model, we adopt a probabilistic velocity motion model similar to that described in [Thrun *et al.* 2005]. The vehicle motion u_t is comprised of two components, the translational velocity v_t and the yaw rate ω_t. Figure 3.4 depicts the probability of being at location x_t given previous location x_{t-1} and control u_t. This distribution is obtained from the kinematic equations, assuming that vehicle motion is noisy along its rotational and translational components.

For the measurement model $P(z_t \mid x_t, M_{t-1})$, mixture beam-based model is widely used in the literature [Fox *et al.* 1999a, Hähnel *et al.* 2003a]. However, the model comes at the expense of high computation since it requires ray casting operation for each beam. This can be a limitation for real time application if we want to estimate a large amount of measurements at the same time. To avoid ray casting, we propose an alternative model that only considers end-points of the beams. Because it is likely that a beam hits an obstacle at its end-point, we focus only on occupied cells in the grid map.

A voting scheme is used to compute the probability of a scan measurement z_t

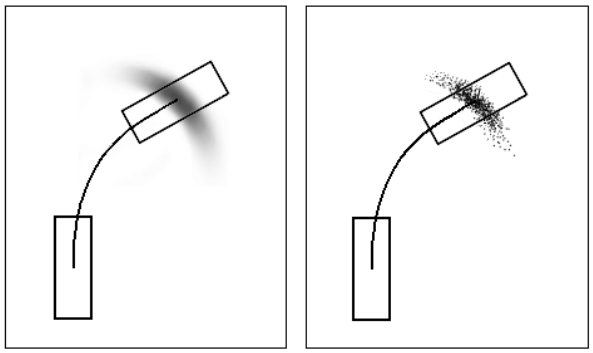


Figure 3.4: The probabilistic velocity motion model $P(x_t \mid x_{t-1}, u_t)$ of the vehicle (left) and its sampling version (right).

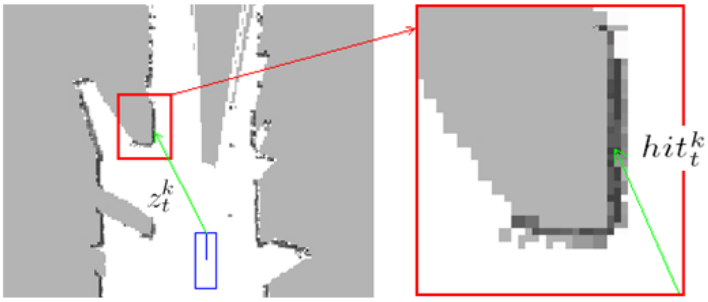


Figure 3.5: The measurement model $P(z_t \mid x_t, M_{t-1})$.

given the vehicle pose x_t and the map M_{t-1} constructed so far. First, from the vehicle location x_t, individual measurement z_t^n is projected into the coordinate space of the map. Call hit_t^n the grid cells corresponding to the projected end-points. If this cell is occupied, a sum proportional to the occupancy value of the cell will be voted. Then the final voted score represents the likelihood of the measurement. Let $P(M_t^i)$ denote the posterior probability of occupancy of the grid cell M^i estimated at time t (following (3.10)), we can write the measurement model under the sum following:

$$P(z_t \mid x_t, M_{t-1}) \propto \sum_{n=1}^{N} \{\, P(M_{t-1}^{hit_t^n}) \text{ such that } M_{t-1}^{hit_t^n} \text{ is occupied} \,\} \tag{3.15}$$

The proposed method is just an approximation to the measurement model because it does not take into account visibility constraints (e.g. a beam can not pass

through an occupied cell), but experimental evidences show that it works well in practice. Furthermore, with a complexity of $O(N)$ where N is the number of beams per scan, the computation can be done rapidly.

It remains to describe how we maximize (3.13) to find the correct pose $\hat{x}_t$. Hill climbing strategy in [Thrun *et al.* 2000, Hähnel *et al.* 2003a] can be used but may suffer from a local maximum. Exploiting the fact that the measurement model can be computed very quickly, we perform an extensive search over the vehicle pose space. A sampling version of the motion model (Figure 3.4 right) is used to generate all possible poses x_t given the previous pose x_{t-1} and the control u_t. The resulting pose will be the pose at which the measurement probability achieves a maximum value. Because of the inherent discretization of the grid, the sampling approach turns out to work very well. In practice, with a grid map resolution of 20cm, it is enough to generate about three or four hundreds of pose samples to obtain a good estimate of the vehicle pose with the measurement likelihood that is nearly unimproved even with more samples. The total computational time needed for such a single scan matching is about 10ms on a conventional PC. An example of scan matching result is shown in Figure 3.6. The most likely vehicle pose is obtained when the laser scan is aligned with the occupied parts of the map and at the same time the vehicle dynamics constraint is satisfied.

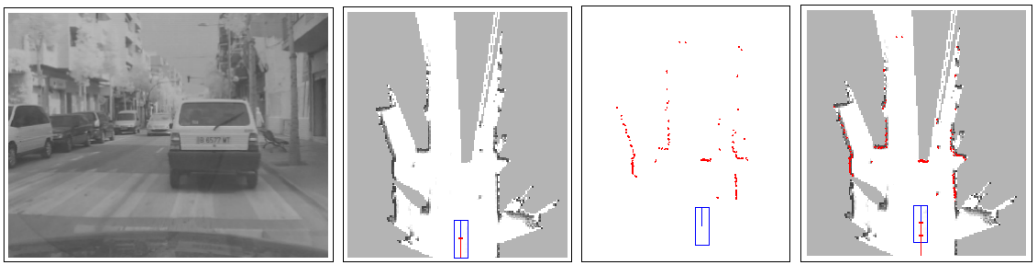


Figure 3.6: Example of scan matching result. From left to right: reference image; local map created so far M_{t-1} and previous vehicle pose x_{t-1}; laser scan at time t; and matching result is obtained by trading off the consistency of the measurement with the map and the previous vehicle pose.

Besides the computational effectiveness, one attraction of our algorithm is that it is not affected by dynamic entities in the environment (see Figure 3.3 right). Since we only consider occupied cells, spurious regions in the occupancy grid

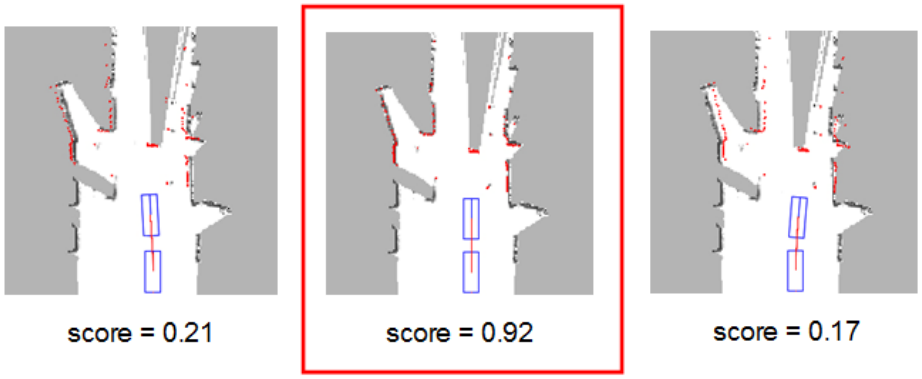


Figure 3.7: Illustration of voting scheme in the scan matching process.

map that might belong to dynamic objects do not contribute to the sum (3.15). The voting scheme ensures that measurement likelihood reach a maximum only when the measurement is aligned with the static parts of the environment. To some meaning, measurements from dynamic entities can be considered as outliers. This property is very useful for moving object detection process that will be described latter in this chapter.

3.2.3 Local Mapping vs. Global Mapping

Up to now, we have not yet considered the loop-closing problem in our mapping process. It is well known that the grid-based maximum likelihood SLAM approach we utilized does not provide a mechanism for loop closing and also is suffered from too much storage and computation load for large scale environments (e.g. mapping city-sized environments). To overcome these limitations, when mapping cyclic environments we can follow the idea proposed by Wang [Wang 2004] to build a consistent global map.

We locally solve SLAM by a maximum likelihood method with occupancy grid maps, and globally solve it by Extended Kalman Filter (EKF) with feature-based maps where each feature is a local grid map with 3-Degree of Freedom (3-DOF) state. (this state can be estimated using visual image registration algorithms from the computer vision literature applied between grid maps). Feature-based EKF algorithm can smoothly solve the loop closing problem, which is a well-known point in the SLAM literature. Also, to maintain local maps is more efficient

than to update a whole global grid map.

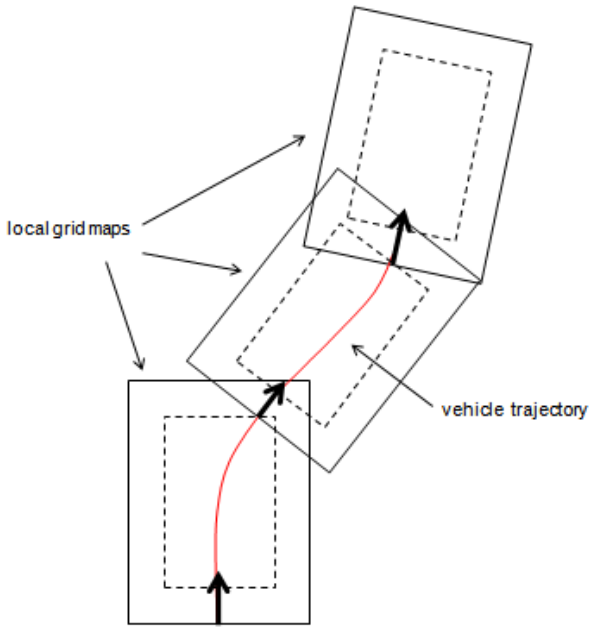


Figure 3.8: A local map is maintained at a time. When the vehicle arrives at the boundary of the grid map, a new grid is created.

Our strategy is that only one local map is maintained at a point in time representing the local environment surrounding of the vehicle. The size of the local map is chosen so that it should not contain loops and the resolution is maintained at a reasonable level. Every time the vehicle arrives near the map boundary, a new grid map is initialized. The pose of the new map is computed according to the vehicle global pose and cells inside the intersection area are copied from the old map (see Figure 3.8).

3.3 Moving Object Detection

We note that, when performing SLAM in dynamic environments, measurements can come from both static and dynamic objects. In the presence of many dynamic entities, the localization technique mentioned in the previous section might be affected. In addition, integrating all measurements from dynamic objects into mapping process leads to an overall decreased quality of the resulting map.

For these reasons, it is necessary to detect moving objects or differentiate between the dynamic and static measurements of the environment. Doing this certainly will considerably contribute to better results of the localization and mapping process.

In the following we describe a motion-based approach based on the grid map to detect moving objects. During the SLAM algorithm described above constructs incrementally a consistent local map of the environment, moving objects can be detected whenever new measurements arrive. The principal idea is based on the inconsistencies between observed free space and occupied space in the local grid map. If an object is detected on a location previously seen as free space, then it is a moving object. If an object is observed on a location previously occupied then it probably is static. However, if an object appears in a previously not observed location, then we can say nothing about that object.

Here we add another important clue which can help to decide an object is dynamic or not using evidences about moving objects detected in the past. For example, if there are many moving objects passing through an area then any object that appears in that area should be recognized as a potential moving object. For this reason, apart from the local static map M as constructed by SLAM described in the previous section, a local dynamic grid map D is created to store information about previously detected moving objects. The pose, size and resolution of the dynamic map is the same as those of the static map. Each dynamic grid cell store a value indicating the number of observations that a moving object has been observed at that cell location.

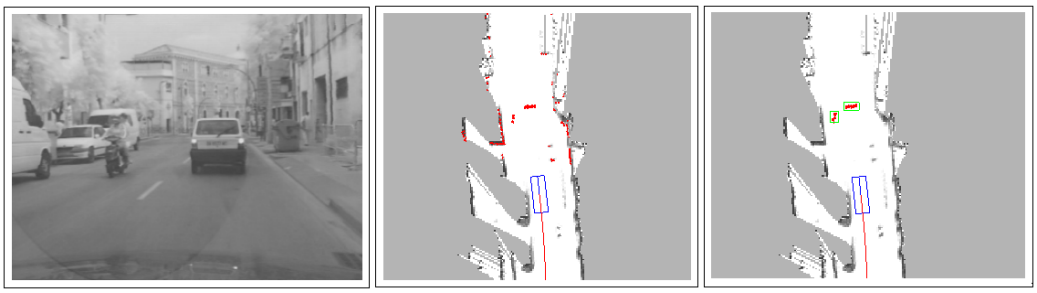


Figure 3.9: Moving object detection example. See text for more details.

From these remarks, our moving object detection process is carried out in two steps as follows. The first step is to detect measurements that might belong to

dynamic objects. Here for simplicity, we will temporarily omit the time index. Given a new laser scan z, the corrected vehicle location and the local static map M computed by SLAM and the dynamic map D containing information about previously detected moving objects, state of a single measurement z^k is classified into one of three types following:

$$state(z^n) = \begin{cases} static & : M_{hit^n} = occupied \\ dynamic & : M_{hit^n} = free \text{ or } D_{hit^n} > \alpha \\ undecided & : M_{hit^n} = unknown \end{cases}$$

where hit^n is the coordinate of the grid cell corresponding to the end-point of the beam z^n and α is a predefined threshold.

The second step is after dynamic measurements are determined, moving objects are then identified by clustering end-points of these beams into separate groups, each group represents a single object. Two points are considered as belonging to the same object if the distance between them is less than 0.3m.

Figure 3.9 illustrates the described steps in moving object detection process. The leftmost image depicts the situation where the vehicle is moving along a street seeing a car moving ahead and a motorbike moving in the opposite direction. The middle image shows the local static map and the vehicle location computed by SLAM and the current laser scan is drawn in red. Measurements which fall into free region in the static map are detected as dynamic and are displayed in the rightmost image. After the clustering step, two moving objects in green boxes are identified and correctly corresponds to the car and the motorbike.

Note that our map updating procedure makes use of results from moving object detection step. Measurements detected as dynamic are not used to update the map in SLAM. For unknown measurements, a priori we will suppose that they are static until latter evidences come. This will help to eliminate spurious objects and result in a better map. Figure 3.10 shows two occupancy grid maps constructed from the same laser data in Figure 3.3 with and without filtering out dynamic measurements. We can see that the left one built without the filtering step results in many fuzzy regions.

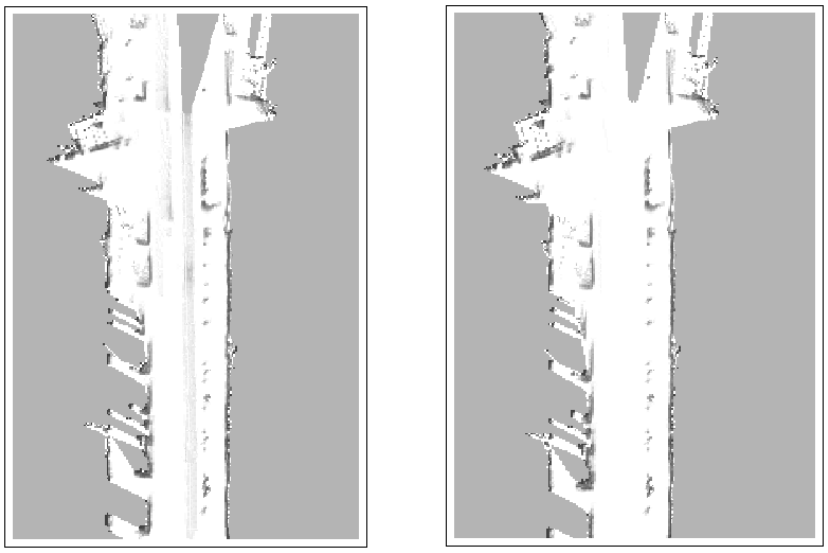


Figure 3.10: Occupancy grid maps built with and without filtering out detected moving objects.

3.4 Experimental Results

We test our approach to SLAM with moving object detection described above using the Navlab dataset [Wang *et al.* 2004]. The dataset was collected using a SICK laser scanner mounted on a moving vehicle (Figure 3.11). The vehicle was driven in real-life traffics. The maximum laser range of the scanner is 80m with the horizontal resolution of 0.5°. The data was collected at 37.5Hz. We only use laser data and odometry vehicle motion information such as translational and rotational velocity (speed and yaw rate) are computed and provided by internal sensors. Images from camera are only for visualization purpose.

In our implementation, the width and height of local grid map are set to 160m and 200m respectively, and the grid resolution is set to 20cm. Every time the vehicle arrives at 40m from the grid border, a new grid map is created. The map is updated and moving objects are detected for every new laser scan.

The results of local SLAM and moving object detection are shown in Figure 3.12. The images in the first row represent online maps and objects moving in the vicinity of the vehicle are all detected and tracked. The current vehicle location is represented by blue box along with its trajectories after corrected from the odometry. The red points are current laser measurements that are identified as

Figure 3.11: Navlab testbed.

belonging to dynamic objects. The green boxes indicate detected moving objects with corresponding tracks shown in dark-yellow. The second row are images for visual references to corresponding situations.

In Figure 3.12, the leftmost column depicts a highway scenario where the testbed car is moving at a very high speed of about 120km/h while other two cars moving in the same direction in front of it are identified. In the middle is the situation where the testbed car is moving at 80km/h on a country road. A car moving ahead and two other cars in the opposite direction are all recognized. Note that the two cars on the left lane are only observed during a very short period of time but both are detected successfully. In the third situation, the testbed moving quite slowly at 25km/h in a crowded city street. Our system detects both the pedestrian moving in front of it and the car moving far ahead. Temporary stationary objects like another standing pedestrian and several other cars parked nearby are considered as static objects.

In all three cases, accurate trajectories of the testbed car are achieved and local maps around the vehicle are constructed consistently. In our implementation, the computational time required to perform both SLAM and moving object detection for each scan is about $20-30$ms on a Pentium IV 3.0GHz, 1Gb RAM PC running Linux. This confirms that our algorithm is able to run synchronously with the data time cycle. The readers can refer to: http://emotion.inrialpes.fr/~tdvu/videos/ for resulting videos.

To demonstrate our localization algorithm can work in cyclic and large

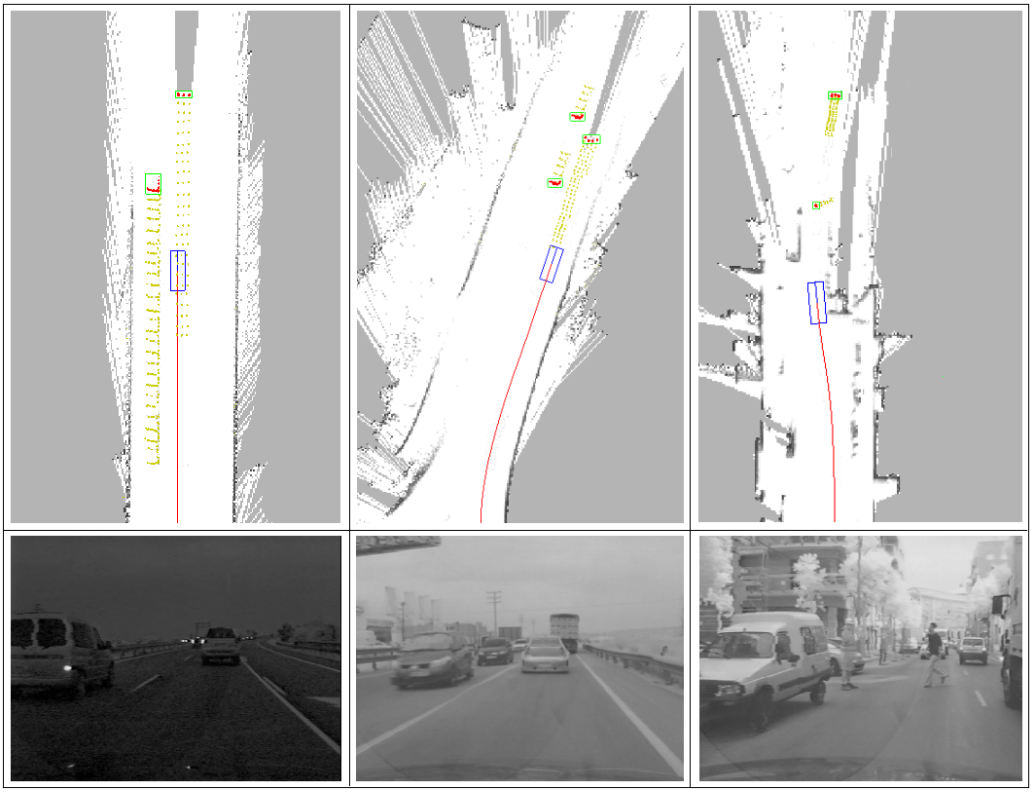


Figure 3.12: Experimental results show that our algorithm can successfully perform both SLAM and moving object detection in real time for different environments.

environments, we show the vehicle trajectory corrected from odometry with data collected through a loop of streets (Figure 3.13). The figure shows that the vehicle trajectory obtained from our localization algorithm is comparable with the ground-truth map.

3.5 Summary

In this chapter, we have presented a method to perform SLAM with detection of moving objects. This is done based on a fast grid-based scan matching algorithm which allows estimating precise vehicle locations and building a consistent map surrounding of the vehicle. After a consistent local vehicle map is

Figure 3.13: The result of localization compared with the ground-truth map.

built, moving objects are detected reliably without knowing a prior knowledge of that objects. The results obtained from moving object detection step help to filter out spurious objects resulting in a better map of the environment. Experiments on real-traffic data have shown that our system can successfully perform a real time mapping with moving object detection from a vehicle moving at high speeds in different dynamic outdoor scenarios.

Chapter 4

DATMO using Markov Chain Monte Carlo

4.1 Introduction

In Chapter 3, we discussed how to model static parts of a dynamic environment, focusing on the SLAM problem with filtering out the presence of dynamic entities. In this chapter, we discuss how to model dynamic parts of the environment, focusing particularly on the problem of detection and tracking of moving objects (DATMO). Inheriting the SLAM results presented in the previous chapter, we will assume that a reasonably precise localization of the ego-vehicle and a map surrounding of the vehicle are always available.

Conventionally, as introduced in Section 2.3, we can separate detection and tracking as two independent procedures: the detector and the tracker. At each time step, the tracker takes a list of observations about moving objects returned by the detector together with observations in the past to solve the data association over the observation space to find correct object trajectories (or tracks) then apply filtering techniques to estimate dynamic states of moving objects. In this way, we can take moving object detection results from the previous chapter and develop an independent tracker. This is similar to the approach described in the PhD theses of Wang [Wang 2004] and Burlet [Burlet 2007] who handle the tracking using the Multiple Hypothesis Tracker (MHT) coupled with a IMM-based filter.

In this conventional approach, since moving object detection at one time instant usually results in ambiguities that makes the data association become

more difficult with missing detections or false alarms. Here we introduce another approach which combines detection and tracking together. We present a probabilistic method for simultaneous detection and tracking of moving objects taking history of measurements that allows the object detection process to make use of temporal information and facilitates a robust tracking of the moving objects.

Moreover, noting that moving objects are detected as free-form in the detection process presented previously. An advantage of this method is that it can be used to detect any kind of objects without knowing a prior knowledge about that objects. However, this suffers from well-known problems with laser-based tracking as explained in the subsection 2.3.1. For example, objects can be divided into several segments or come in different shapes over time leading to an incorrect tracking result. Here we take a model-based approach and introduce predefined models to represent typical moving object classes.

Our algorithm to solve DATMO is summarized as follows. We formulate the detection and tracking problem as finding the most likely trajectories of moving objects given data measurements over a sliding window of time (Figure 4.1a)). A trajectory (track) is regarded as a sequence of object shapes (models) produced over time by an object which must be satisfied the constraint of both an underlying object motion and the consistency with measurements observed from frame to frame. In this way, our approach can be seen as a batch method searching for the global optimum solution in the spatio-temporal space. Due to the high computational complexity of such a scheme, we employ a Markov chain Monte Carlo (MCMC) technique that enables traversing efficiently in the solution space. We employ the detection results from the previous chapter as a *coarse detector* to generate potential moving object hypotheses with predefined models that helps to drive the search more efficiently. This technique earns its name data-driven MCMC (DDMCMC) in the literature [Zhu *et al.* 2000].

The key contribution in this chapter is to design a general framework to perform object detection and tracking at the same time with an explicit integration of various aspects including prior information, object model, measurement model, motion model in a theoretically sound formulation. We test the proposed algorithm on real-life urban traffic datasets and present preliminary results.

The remaining of the chapter is organized as follows. In the following section, we introduce a general formulation of the moving object detection and tracking

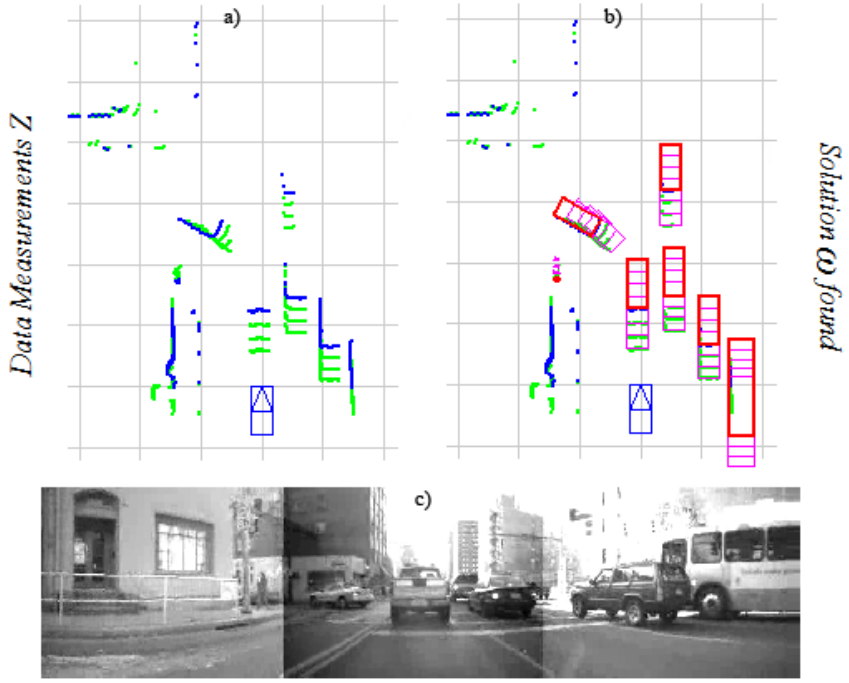


Figure 4.1: Example of an interpretation of moving object trajectories from a laser data sequence. (a) Data comprised of four scans consecutive: in blue is current scan and in green are scans in the past; (b) A solution found including seven tracks of four cars and one bus represented by red boxes and one pedestrian represented by red dots which are imposed on the range data; (c) situation reference.

problem. In Section 4.3 we present our algorithm to find the optimal trajectories of moving objects using a spatio-temporal MCMC sampling method. We discuss experiments and provide some initial results on real-life traffic datasets in Section 4.4, followed by some summary remarks in Section 4.5.

4.2 DATMO Formulation

We consider detection and tracking in a sliding window of time which is comprised of $T \in N^+$ last frames. Let Z be the set of all data measurements within the time interval $[1, T]$ and $Z = \{z_1, ..., z_T\}$ where z_t denotes the laser scan measurement at time t. The current time corresponds to $t = T$. Assuming that within $[1, T]$ there are K unknown number of objects moving in the vicinity of the vehicle.

The moving object detection and tracking problem is formulated as maximizing a posterior probability (MAP) of an interpretation of tracks ω of moving objects, given a set of laser measurements Z over T frames:

$$\omega^* = \underset{\omega \in \Omega}{\operatorname{argmax}} P(\omega|Z) \tag{4.1}$$

According to the Bayes rule, the posterior probability is decomposed into a prior term and a likelihood term:

$$P(\omega|Z) \propto P(\omega)P(Z|\omega) \tag{4.2}$$

In the following, we describe object shape models, the prior model and the likelihood model .

4.2.1 Object models

In general, there are so many classes of objects in traffic scenes but the number classes of moving objects of interest is quite limited. Here we distinguish four classes of moving objects: bus, car, bike, pedestrian (motorcycles and bicycles belong to the bike class). We use a box model of fixed size to represent bus, car, bike and a point model to represent pedestrian. Our approach is different with the approach proposed by Petrovskaya [Petrovskaya & Thrun 2008] who used a flexible box model to represent cars and introduced a method to learn object sizes during tracking. A problem with this approach is that during most of the time objects being tracked, they are not totally visible to the laser sensor so that the adaptive sizes do not necessarily correspond to the actual size of the objects being tracked. In addition, her work only deals with detection and tracking of vehicles.

To define a typical size for each object class, a priori knowledge about typical road users is utilized. Figure 4.2 shows, for example, the distribution of the physical dimensions of 250 different passenger cars sold in Europe [Dietmayer *et al.* 2001]. Therefore a fixed rectangular model of 1.7m width by 4.5m length can reasonably represent the car class.

Similar data are provided for buses with typical width of 2.5m and length of 12.0m, trucks without trailer with typical length of 9.0m and width of 2.3m, typical motorcycles length 2.1m and width 0.5m and persons with a typical

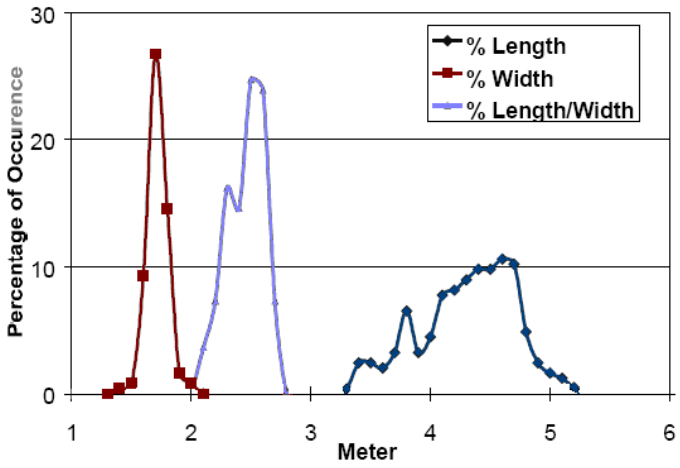


Figure 4.2: Distribution of length, width and the length/width ratio of passenger cars in Europe.

diameter: 0.5m.

To generalize, we represent models of objects as follows. For the box model, object is parameterized by $M = \{c, x, y, w_c, l_c, \theta\}$ which are object class, object center position, width, length and orientation respectively. Herein w_c and l_c are constants with respect to the width and length of each object class c. For the point model, object is parameterized by $M = \{c, x, y\}$ which are object class and object center (see Figure 4.3).

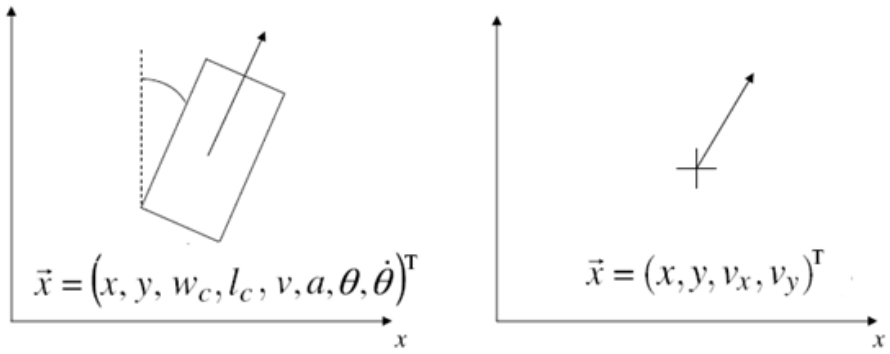


Figure 4.3: Box model and point model to represent moving objects.

4.2.2 Solution Space

A solution ω for the Equation (4.1) includes a set of K trajectories (tracks) of moving objects appeared during the tracking over the last T frames:

$$\omega = \{\tau_1, \tau_2, ..., \tau_K\} \tag{4.3}$$

Each track τ_k in ω is defined as a sequence of the same object appears in time:

$$\tau_k = \{\tau_k(t_1), ..., \tau_k(t_{|\tau_k|})\} \tag{4.4}$$

where $t_i \in [1, T]$, $|\tau_k|$ is the length of track, $\tau_k(t)$ represents moving object detected at time t with its associated properties which is either of form $\{c, x, y, w_c, l_c, \theta\}$ or $\{c, x, y\}$.

Figure 4.1 shows an example of one possible interpretation of moving objects from a sequence of four laser scans. The solution found is comprised of seven tracks of five cars, one bus and one pedestrian $\omega = \{\tau_1, ..., \tau_7\}$.

We introduce the notation $\omega_t = \bigcup_{k=1}^{K} \tau_k(t)$ representing the set of moving objects visible at time t (see Figure 4.4). Since object occlusion or missing object detection might happen, we set $1 \leq (t_{i+1} - t_i) \leq t_{max}$. Looking at this figure, vertically at each time slice ω_t can be seen as the result of the moving object detection at time t and horizontally ω can be seen as a data association process over object observation space given by the detector. In this meaning, searching the solution ω means that we simultaneously deal with both detection and tracking problems.

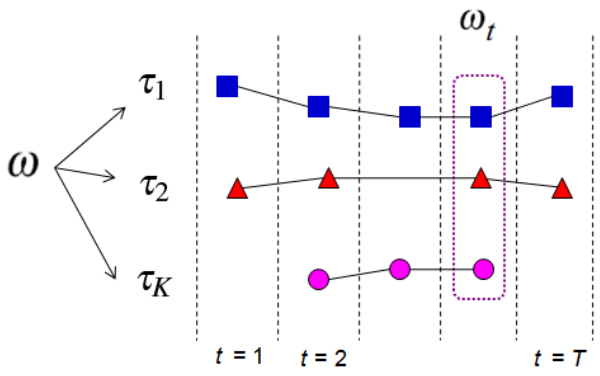


Figure 4.4: Illustration for the notation in use.

The complexity of the entire solution space Ω ($\omega \in \Omega$) can be roughly estimated as follows:

$$\begin{aligned}\Omega &= \cup_{k=0}^{\infty}\Omega_k \\ \Omega_k &= \{\cup_{l=1}^{T}\mathcal{T}_l\}^k, \\ \mathcal{T}_l &= (3 \times \mathcal{R}^3 + \mathcal{R}^2)^l\end{aligned}$$

where Ω_k is the subspace of solutions comprised of exactly k tracks, $\mathcal{T}_l$ is the space for tracks with length of l, $\mathcal{R}^3$ is the space for position and orientation parameters of non-people object classes (we have three classes) and $\mathcal{R}^2$ is the space for position parameters of the people class.

With this complexity of the solution space Ω, it is impossible to solve (4.1) in a conventional method. In addition, since we do not know in advance how many objects (number of tracks), how long they exist (track lengths) in the scene, the solution space contains subspaces of varying dimensions which creates difficulties in pursuing the solutions.

To tackle these difficulties, we take a sampling approach to explore in the solution space to search for the maximum a posterior (MAP) in (4.2). Before going into detail, we describe how we model the prior and the likelihood in order to compute the posterior of a solution ω given measurements Z.

4.2.3 Prior Probability

The prior $P(\omega)$ in the Equation (4.2) simply reflects the probability we have seen an arrangement of K object trajectories on road without taking into account the sensor observations. Assuming that the occurrence of an object is independent of the others, we define the prior of a solution ω is a product of probabilities of individual tracks:

$$P(\omega) = \prod_{k=1}^{K} P(\tau_k) \tag{4.5}$$

The probability $P(\tau_k)$ encodes probabilities of the appearance of objects independent over time which is denoted by $P_O(.)$ and the temporal consistency of object positions within the track which is denoted by $P_T(.)$. The probability $P_T(.)$ controls the inner-smoothness of each track independently (Figure 4.5). However,

without an *a priori* knowledge of the number of targets, the inner-smoothness constraint will favor shorter paths, and therefore will split a trajectory into a large number of sub-tracks. To overcome this overfitting problem, we add a prior term $P_L(.)$ which encodes the preference of longer track. Now we can write:

$$P(\tau_k) = P_L(\tau_k)P_O(\tau_k)P_T(\tau_k) \tag{4.6}$$

1. We adopt an exponential model over the length of each track:

$$P_L(\tau_k) \propto \exp(\lambda_L|\tau_k|) \tag{4.7}$$

   where λ_L is a preset constant to weight down short tracks and favor longer tracks.

2. The probability of appearance of objects in track independent of time $P_O(.)$ means how often we see an object of a given class at a specific position and direction:

$$P_O(\tau_k) = \prod_{i=1}^{|\tau_k|} P(\tau_k(t_i)) = P(c_k) \prod_{i=1}^{|\tau_k|} P(x_i, y_i)P(\theta_i) \tag{4.8}$$

   where $P(c_k)$, $P(x_i, y_i)$ and $P(\theta_i)$ are the prior probabilities over the object class, object position and object orientation respectively. In our current implementation, we set these prior probabilities as uniform distributions. However, knowledge of road type and road border, if available can be added for better performance. For example, vehicle movement direction is usually parallel with the road border. Or we unlikely see a pedestrian when moving on a highway.

3. There remains how to model the temporal consistency within each track $P_T(\tau_k)$. This term is measured by the smoothness of object motion according to its underlying dynamics model. Since in urban scenarios objects are high maneuvers, we opt for a multiple model approach to model object dynamics (subsection 2.3.2). For classes of bus, car, bike, four modes of dynamics are used: constant velocity, constant acceleration, turning and stationary mode. For pedestrians, we force the acceleration to zero and only one constant velocity model is used. Box-model dynamic states are

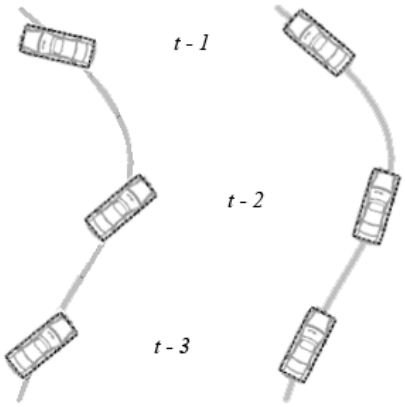


Figure 4.5: Temporal consistency of a track. Obviously the arrangement of cars on the right is more relevant to a correct motion than that on the left.

$(x, y, \theta, \dot{\theta}, v, a)$ with the velocity v and acceleration a are always in the direction θ of the longer edge l. Dynamic states for point-model objects are (x, y, v_x, v_y).

We apply an IMM filter similar to [Zhao & Thorpe 1998] to estimate states of objects in the track sequentially: $\hat{\tau_k}(t_1), ..., \hat{\tau_k}(t_{|\tau_k|})$ and their corresponding covariances: $cov(\hat{\tau_k}(t_1)), ..., cov(\hat{\tau_k}(t_{|\tau_k|}))$. Set $cov(\tau_k) = cov(\hat{\tau_k}(t_{|\tau_k|}))$. A smaller value of $cov(\tau_k)$ implies that the track is more consistent to the underlying dynamics model. The temporal consistency of a track is then calculated by:

$$P_T(\tau_k) \propto \exp(-\lambda_T cov(\tau_k)) \tag{4.9}$$

where λ_T is a preset constant to weigh the importance of this probability component.

4.2.4 Likelihood Probability

The likelihood $P(Z|\omega)$ (4.2) reflects the probability we observe the measurements Z given ω which contains the states of all moving objects appeared during the time interval $[1, T]$. Note that the measurements Z comes from both static and dynamic objects.

To model this likelihood, first we identify laser measurements which are

caused by moving objects in ω. Figure 4.6 shows a box-model object (solid rectangle) and its dilated bounding box (dotted rectangles) representing the uncertainty of measurements when laser hits the object. Taking a laser beam Az, it is considered as being caused by the object if it has an impact lying on the segment BD that corresponds to a measurement on visible sides. We notice that if the impact lies on the segment AB, the object is occluded. If the impact lies on the ray Dz and it is not a maximum range reading, the visibility constraint is violated (rays seeing through glasses or black surfaces are not the case). For the point-model object, it is dilated by a circle of a fixed diameter and all measurements fall inside the circle are considered as measurements from that object.

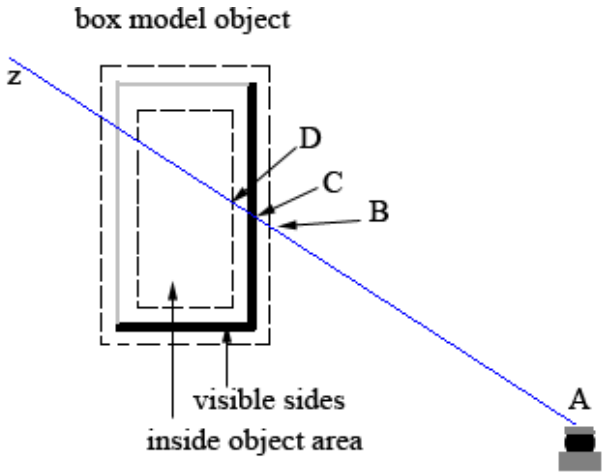


Figure 4.6: Object measurement likelihood computation.

Let $Z^{(d)}$ denote all laser measurements which is identified from the step described above as being caused by moving objects from the solution ω. We have $Z^{(s)} = Z - Z^{(d)}$ the remained measurements which are supposed to be caused by static objects. Note that Z include T laser scans $Z = \{z_1, ..., z_T\}$. We denote $z_t^{(d)}$ and $z_t^{(s)}$ as measurements from dynamic and static objects at time t respectively.

By this definition, besides information about dynamic objects, the solution ω can be considered as a partition of Z into dynamic and static measurements. From frame to frame, these measurements should be consistent with each others.

We can therefore decompose the likelihood into a product of two terms:

$$p(Z|\omega) = \prod_{i,j=1}^{T} P(z_i|\omega_j) \prod_{i,j=1}^{T} P(z_i|z_j^{(s)}) \tag{4.10}$$

where the first term encodes the likelihood of a laser scan at a time given observations of moving objects and the second term encodes the consistency of the scan with static parts of the environment. (Remember that ω_i denotes list of moving objects at time i).

The first term is then further decomposed:

$$\begin{aligned}\prod_{i,j=1}^{T} P(z_i|\omega_j) &= \prod_{i} P(z_i|\omega_i) \prod_{i\neq j} P(z_i|\omega_j) \\ &= \prod_{i} P_{M_1}(z_i^{(d)}|\omega_i) P_{M_2}(z_i^{(s)}|\omega_i) \prod_{i\neq j} P_{M_3}(z_i^{(s)}|\omega_j)\end{aligned} \tag{4.11}$$

The second term in (4.10) is rewritten as:

$$\prod_{i,j=1}^{T} P(z_i|z_j^{(s)}) = \prod_{i\neq j} P_{M_4}(z_i|z_j^{(s)}) \tag{4.12}$$

The meaning of each probability component is as follows. P_{M_1} scores the fitness of dynamic measurements to the moving objects. P_{M_2}, P_{M_3} and P_{M_4} penalizes the violation of laser visibility constraint. In particularly, P_{M_2} penalizes situations that laser can see through dynamic objects. P_{M_3} penalizes the situations where moving objects are detected at position that has seen static objects. P_{M_4} penalizes the situations where laser can see through static objects. Figure 4.7, in order from left to right, illustrates the meanings of probabilities $P_{M_1}, P_{M_2}, P_{M_3}$ and P_{M_4} respectively. Note that here we do not consider maximum range laser readings so that rays caused by glasses or black surfaces will not be penalized.

In the following, we discuss in detail how we model these probabilities:

1. Assuming that $z_i^{(d)}$ is comprised of N dynamic measurements $z_i^{(d)} = \{z_i^1, ..., z_i^N\}$. We consider measurements obtained along each laser ray independent of each other. The measurement likelihood P_{M_1} factors as:

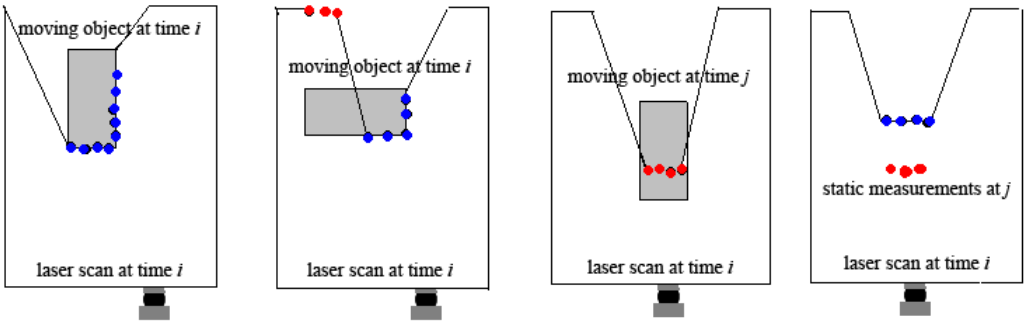


Figure 4.7: Four types of constraint used to compute the likelihood. The red dots are measurements that violate the laser visibility constraint.

$$P_{M_1}(z_i^{(d)}|\omega_i) = \prod_{n=1}^{N} P(z_i^n|\omega_i) \tag{4.13}$$

Each measurement z_i^n corresponds to a laser beam that hits an object in ω_i where the laser impact lies on the segment BD (Figure 4.6). We model each ray's likelihood as a zero-mean Gaussian with respect to the distance d_n from the impact to the ideal measurement point C:

$$P(z_i^n|\omega_i) \propto exp(-\lambda_1 d_n) \tag{4.14}$$

2. For each object in ω_i, we count the number of non-maximal measurements $z_i^{(s)}$ that fall behind the object model, call c_2 is the total number.

$$P_{M_2}(z_i^{(s)}|\omega_i) \propto \exp(-\lambda_2 c_2) \tag{4.15}$$

P_{M_2} supports the remark following: If there exists such a violation of visibility constraint, there is unlikely a moving object appeared at that position.

3. Similarly, we count c_3 the number of measurements $z_i^{(s)}$ that fall inside occupied areas of the objects in ω_j.

$$P_{M_3}(z_i^{(s)}|\omega_j) \propto \exp(-\lambda_3 c_3) \tag{4.16}$$

P_{M_3} is in favor of the fact that if there is a moving object appeared at one time step then all measurements backward or afterward falling inside the object area should be dynamic measurements.

4. In the same way, let c_4 be the total number of violated measurements:

$$P_{M_4}(z_i|z_j^{(s)}) \propto \exp(-\lambda_4 c_4) \tag{4.17}$$

P_{M_4} holds the observation that if there are measurements in one frame observed passing through other measurements in another frame then the later measurements are likely dynamic measurements.

4.2.5 Posterior Probability

By combining the above likelihood and the prior probability, we get the posterior probability function as:

$$P(\omega|Z) \propto \exp\{\ \lambda_L S_{len} - \lambda_T S_{mot} - \lambda_1 S_{ms1} - \lambda_2 S_{ms2} - \lambda_3 S_{ms3} - \lambda_4 S_{ms4}\} \tag{4.18}$$

where $\lambda_L, \lambda_T, \lambda_1, \lambda_2, \lambda_3, \lambda_4$ are positive real constants being chosen empirically beforehand and:

$$\begin{aligned} S_{len} &= \sum_{k=1}^{K} |\tau_k| & S_{mot} &= \sum_{k=1}^{K} cov(\tau_k) \\ S_{ms1} &= \sum_{i=1}^{T}\sum_{n=1}^{N} d_n^i & S_{ms2} &= \sum_{i=1}^{T} c_2^i \\ S_{ms3} &= \sum_{i\neq j} c_3^{ij} & S_{ms4} &= \sum_{i\neq j} c_4^{ij} \end{aligned} \tag{4.19}$$

4.3 Efficient MAP Computation using MCMC

We want to find the solution ω that maximizes the posterior probability defined in (4.18). However, as stated in Section 4.2.2, the solution space contains subspaces of varying dimensions (number of tracks). It also includes both discrete

variable (object-track associations) and continuous variables (object positions and directions). These makes the optimization challenging.

Dealing with this difficulty, we employ a Markov chain Monte Carlo (MCMC) method with jump/diffusion dynamics to sample the posterior probability in such a complex solution space to search for the maximum. Jumps cause the Markov chain to move between subspaces with different dimensions and traverse the discrete variables; diffusions make the Markov chain sample continuous variables. In the process of sampling, the best solution is recorded and the uncertainty associated with the solution is also obtained. We introduce the basic idea of MCMC in the following.

4.3.1 MCMC algorithm

The basic idea of MCMC is as follows. A Markov chain can be designed to sample a probability distribution $\pi(\omega)$ (in our case $\pi(\omega) = P(\omega|Z)$). At each iteration, we sample a new state ω' from the current state ω_n following *a proposal distribution* $q(\omega'|\omega_{n-1})$ (in simple words, what new state should the Markov chain go from the previous state). The new candidate state ω' is accepted with the following probability $A(\omega_{n-1}, \omega')$ where

$$A(\omega_{n-1}, \omega') = min(1, \frac{\pi(\omega')}{\pi(\omega_{n-1})} \frac{q(\omega_{n-1}|\omega')}{q(\omega'|\omega_{n-1})}) \tag{4.20}$$

otherwise the sampler stay at ω_{n-1}.

The term $\frac{\pi(\omega')}{\pi(\omega_{n-1})}$ is the relative probability of the states ω_{n-1} and ω'; the term $\frac{q(\omega_{n-1}|\omega')}{q(\omega'|\omega_{n-1})}$ expresses the relative difficulty of going from ω' to ω_{n-1} and from ω_{n-1} to ω'. If the probability of the proposed state, combined with the probability of going back to ω_{n-1}, is greater than the reverse, the move is accepted; otherwise, it is still accepted probabilistically, which is different from the gradient search. If the candidate state ω' is accepted, we set $\omega_n = \omega'$; otherwise, $\omega_n = \omega_{n-1}$

The overview of MCMC algorithm is shown in Algorithm 4.1. This is the well-known Metropolis-Hasting algorithm [Tierney 1996]. The proposal probability $q(.)$ is called the *dynamics* of the Markov chain. There are usually two kinds of dynamics: *jump* and *diffusion*. Jump refers to the motion of Markov chain between subspaces of different dimensions and diffusion refers to its motion within a subspace.

Algorithm 4.1 MCMC Sampler

Input: $Z, n_{mc}, \omega^* = \omega_0$ **Output**: ω^*
for $n = 1$ to n_{mc} **do**
  Propose ω' according to $q(\omega'|\omega_{n-1})$
  Sample U from $Uniform[0, 1]$
  if $U < A(\omega, \omega')$ **then**
    $\omega_n = \omega'$
    if $P(\omega_n|Z) > P(\omega^*|Z)$ **then** $\omega^* = \omega_n$
  else
    $\omega_n = \omega_{n-1}$
end for

It is proved that the Markov chain constructed this way has its stationary distribution equal to $\pi(\omega)$, independent of the choice of the proposal probability $q(.)$ and the initial state ω_0. However, the choice of the proposal probability $q(.)$ can affect the efficiency of the MCMC significantly. A random proposal probability will lead to a very slow convergence rate while a proposal probability designed with domain knowledge will make the Markov chain traverse the solution space more efficiently. If the proposal probability is informative enough so that each sample can be thought of as a *hypothesis*, then the MCMC approach can be though of as a stochastic version of the *hypothesize and test* approach [Zhu *et al.* 2000] that earns the approach its name *Data-Driven* MCMC method (DDMCMC).

To make the proposals more informative, we take advantage of the detection module in the previous chapter. Avoiding the confusion, we make it clear that the detector presented in previous chapter is only able to detect parts of moving objects not the actual objects (with models). We can consider this step as a *coarse detector* and can consider moving object parts detected as evidences about moving objects. The idea is that the coarse detector and these moving evidences can help us to focus on the search of solution for moving trajectories on regions where moving objects are potentially present to accelerate the convergence rate of the MCMC sampler.

We present a two-stage DATMO process: The first stage, moving object hypothesis generation, where coarse hypotheses for the presence of moving objects (with associated models) are generated from moving evidences (data-driven). This stage is designed to have high detection rates but also may have

Figure 4.8: A simple illustration of Metropolis algorithm to sample from a Gaussian distribution whose one standard-deviation contour is shown by the ellipse. Steps that are accepted are shown in green and rejected steps are shown in red.

many false alarms. These hypotheses then provide proposals for a second stage MCMC sampler, with jump/diffusion dymamics, that performs a finer search over the spatio-temporal space of potential moving objects to find the most likely trajectories of moving objects with a maximum of posterior probability.

We detail the two-stage process in the following.

4.3.2 Moving object hypothesis generation

We recall the process of detecting moving evidences described in Chapter 3 which will be used to help generating model-based moving object hypotheses. As presented previously, we incrementally constructs an online occupancy grid representing a local map of the vehicle environment based on good vehicle localization obtained by a fast scan matching technique. Moving parts of dynamic objects are then detected when objects enter object-free regions. Here the detection condition is loosened to include the situations when objects are observed in unexplored regions that could be static or new moving objects. Taking all these as moving evidences increases the detection rate and false alarms as well. Result of this step is a set of dynamic segments corresponding to potential moving objects. Note that objects can be seen as several parts so that several segments might be related to the same object.

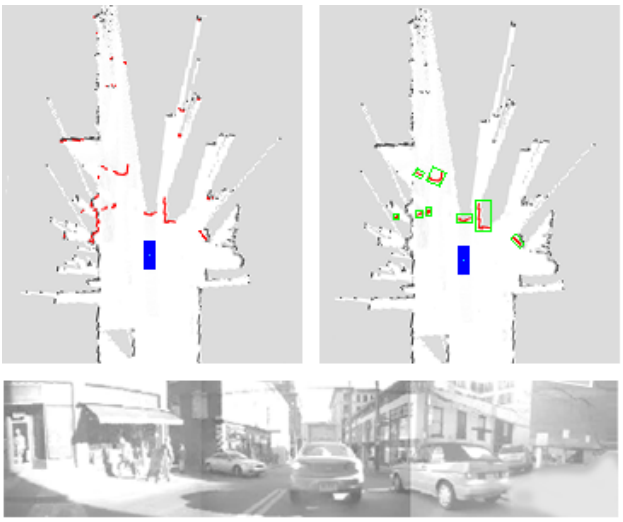


Figure 4.9: Object detection based on occupancy grid.

Figure 4.9 illustrates our detection process. In the figure, the bottom image describes a situation when the host vehicle moving along the street seeing two cars moving ahead, another car coming out of the left turn and two pedestrians walking on the left pavement. The image on top left shows the local grid map constructed around the host vehicle (blue box). In red color is the current laser scan. Laser impacts that fall into free or unexplored regions are detected as dynamic measurements and are displayed in the top right image. Dynamic measurements are then grouped into segments represented in green boxes corresponding to moving objects. Note that the car coming from the left turn is divided into two segments.Two false alarms are also displayed.

Starting from identified dynamic segments, we generate object hypotheses by fitting predefined object models to each segment. The objective is to generate all possible hypotheses corresponding to potential moving objects. The model fitting is carried out as follows. For each segment, a minimum bounding box is computed and corresponding sides of the segment are extracted. We remark that at one time instant, maximum two sides of a segment can be seen by the laser sensor. Providing that the size of a bounding box of a segment is larger than a threshold, the segment is classified as a L-shape if it has two visible sides, as an I-shape if only one side is visible. Otherwise it is classified as a ”mass point”-shape. Depending on the shape and size of segments, object hypotheses are generated

using suitable models. L-shape segments will generate bus, car hypotheses, I-shape segments create bus, car, bike hypotheses and "mass-point" segments will generate pedestrian hypotheses.

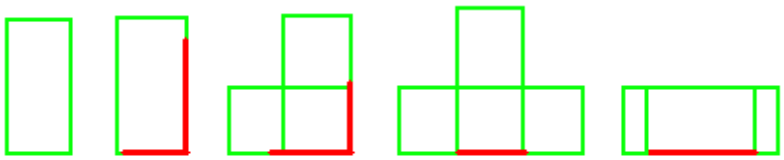


Figure 4.10: Illustration of fitting object box model (green) to L-shape and I-shape segments (red). The last two shapes show that using box model helps connecting discontinued segments.

Figure 4.10 shows possible hypotheses of an object as a box model given L-shape and I-shape segments of different sizes. Note that by fitting object models to segments in this way, models can cover segments nearby so that naturally overcome object splitting problem caused by laser measurement discontinuities (Figure 2.7).

To illustrate the object hypotheses space that is used to guide the proposals in the MCMC process for searching object trajectories, in Figure 4.11, an example image of all possible object hypotheses over $T = 10$ frames are displayed. For visibility purpose, only car model hypotheses are shown.

Now it remains how we use MCMC sampling to search for the solution of object trajectories from the space of moving object hypotheses constructed in the time span $[1, T]$ as described. We notice that in Figure 4.11 many false alarms are generated corresponding to noises or stationary cars on the right border. However, if taking the temporal information and vehicle motion models into account, they are not relevant and will be easily discarded. Four long shape sequences marked with numbers meet the motion consistency and correctly correspond to moving cars.

Basing on this remark, we introduce a neighborhood graph structure over the space of moving object hypotheses to make the sampling more efficient which is described next.

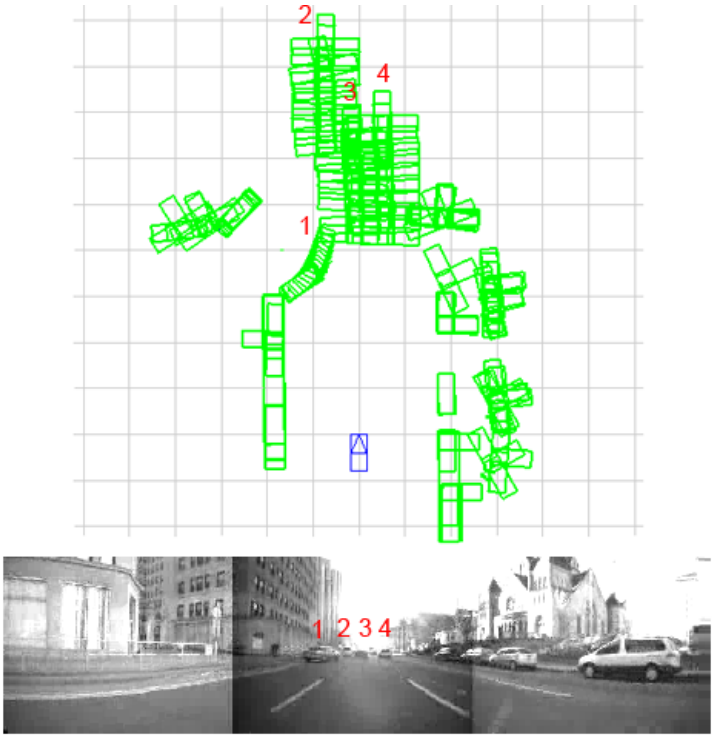


Figure 4.11: An example of moving object hypotheses generated over ten frames. For visibility purpose, only car model candidates are shown.

4.3.3 Neighborhood graph of hypotheses

We use a graph $\langle V, E \rangle$ to represent the relationship of all coarse moving object hypotheses generated as described above within the time interval $[1, T]$. Let h_t^i denote the i-th hypothesis generated at time t. Each hypothesis h_t^i is represented by a node in V. We define the neighborhood between two nodes in the graph by edges in E of two types: sibling edges and parent-child edges:

1. Sibling edges are defined by: $E_{sb} = \{(h_t^i, h_t^j)\}$ where h_t^i and h_t^j are object hypotheses generated at the same time step with the condition that h_t^i and h_t^j share spatial overlap.

2. Parent-child edges are defined by: $E_{pc} = \{(h_{t_1}^i, h_{t_2}^j)\}$ where $t_1 \neq t_2$, h_t^i and h_t^j are object hypotheses generated at different time step with the condition that $h_{t_1}^i$ and $h_{t_2}^j$ are of the same object class and $\|h_{t_1}^i(x, y) - h_{t_2}^j(x, y)\| < |t_1 - t_2| v_{max}$, where $\|.\|$ is the Euclidean distance, $1 \leq |t_1 - t_2| \leq t_{max}$ and

v_{max} is the maximum speed of the object class.

Sibling edges represent exclusion relationship between object hypotheses that are generated from the same moving evidence so that if one is selected to form a track then the other are excluded. Parent-child edges reflect possible temporal association between hypotheses (possible data association).

Figure 4.12 shows an example of neighborhood graph of moving object hypotheses generated over three frames. Object hypotheses at each frame are numbered and object classes are represented by nodes of different shapes. Sibling nodes are displayed inside gray circles. Parent-child nodes are connected by segments.

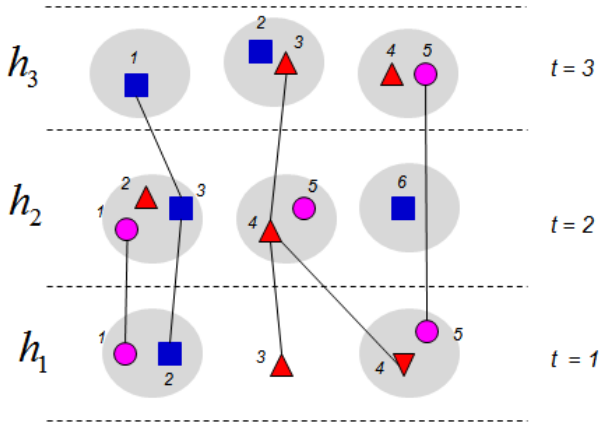


Figure 4.12: Example of a neighborhood graph of moving object hypotheses generated over three frames.

We use the notation $N(.)$ to denote the neighbor set of a hypothesis in the graph structure: $N(h^i_{t_1}) = \{h^j_{t_2} | (h^i_{t_1}, h^j_{t_2}) \in E\}$. Hypothesis $h^j_{t_2} \in N(h^i_{t_1})$ belongs to the parent set $N^{parent}(h^i_{t_1})$, child set $N^{child}(h^i_{t_1})$, sibling set $N^{sibling}(h^i_{t_1})$, when $t_2 < t_1$, $t_2 > t_1$ and $t_2 = t_1$ respectively.

Taking the node h^4_2 in Figure 4.12 for example, we have $N^{parent}(h^4_2) = \{h^3_1, h^4_1\}$, $N^{child}(h^4_2) = \{h^3_3\}$ and $N^{sibling}(h^4_2) = \{h^5_2\}$.

Now instead of search over the entire solution space for the maximum a posterior solution in the Equation 4.1, we only need to search for trajectories over the neighborhood graph of moving object hypotheses. In the following, we describe MCMC dynamics to traverse this reduced solution space using the graph notations mentioned above.

4.3.4 Markov chain dynamics

The Markov chain dynamics correspond to sampling the proposal distribution $q(\omega'|\omega_{n-1})$ described in Algorithm 4.1 (line 3). We assume that at the $(n-1)$-th iteration we have a sample $\omega_{n-1} = \{\tau_1, ..., \tau_K\}$ comprised of K tracks which is formed from nodes of moving objects in V and now propose a candidate ω' for the n-th iteration. Let V^* denote the set of all unselected nodes in V and do not share any sibling edge with nodes contained in ω_{n-1}.

In order for the Markov chain to traverse the solution space, we design the following reversible dynamics:

Track Extension/Reduction: The purpose of the extension/reduction move is to extend or shorten the estimated trajectories. For a forward extension, we select uniformly at random (*u.a.r*) a track $\tau_k \in \omega_{n-1}$. Let $\tau_k(end)$ denote the last node in the track τ_k. Then select *u.a.r* node $h \in \{N^{child}(\tau_k(end)) \cap V^*\}$ and append the new hypothesis h to τ_k. Similarly, for a backward extension, we take a node $h \in \{N^{parent}(\tau_k(start)) \cap V^*\}$. We keep on extending the track τ_k with a probability $\gamma \in [0, 1]$.

The reduction move consists of randomly shortening a track τ_k by selecting a cutting index r *u.a.r* from $\{2, ..., |\tau_k| - 1\}$. In the case of a forward reduction the track τ_k is shortened to $\{\tau_k(t_1), ..., \tau_k(t_r)\}$, while in a backward reduction we take the sub-track $\{\tau_k(t_r), ..., \tau_k(t_{|\tau_k|})\}$.

Track Birth/Death: This move controls the creation of new track or termination of an existing trajectory. In a birth move, we select *u.a.r* a node $h \in V^*$, associate it to a new track and increase the number of tracks $K' = K + 1$. The birth move is always followed by an extension move. From the node h we select the extension direction forward or backward *u.a.r* to extend the track $\tau_{K'}$. If $|\tau_{K'}| < 2$ the move is rejected.

For a death move, we simply choose *u.a.r* a track τ_k and delete it.

Track Split/Merge: For a split move, we *u.a.r* select a track τ_k with $|\tau_k| \geq 4$ and a split point $s \in \{2, ..., |\tau_k| - 2\}$. And we split τ_k into two new tracks $\tau_{s_1} = \{\tau_k(t_1), ..., \tau_k(t_s)\}$ and $\tau_{s_2} = \{\tau_k(t_{s+1}), ..., \tau_k(t_{|\tau_k|})\}$ and increase the number of tracks $K' = K + 1$.

Often, due to missing detection or erroneous detection, trajectories of objects are fragmented. The merge move provides the ability to link these fragmented sub-tracks. If a tracks's (τ_{k_1}) end node is in the parent set of another track's

(τ_{k_2}) start node, this pair of two tracks is candidate for a merge move. We select *u.a.r* a pair of tracks from candidates and merge the two tracks into a new track $\tau_k = \{\tau_{k_1}\} \cup \{\tau_{k_2}\}$ and reduce the number of tracks $K' = K - 1$.

Track Switch: If there exist two break points p, q in two tracks τ_{k_1}, τ_{k_2} such that $\tau_{k_1}(t_p) \in N^{parent}(\tau_{k_2}(t_{q+1}))$ and $\tau_{k_2}(t_q) \in N^{parent}(\tau_{k_1}(t_{p+1}))$ as well, this pair of nodes is one candidate for a switch move. We *u.a.r* select a candidate and define two new tracks as:

$$\tau'_{k_1} = \{\tau_{k_1}(t_1), ..., \tau_{k_1}(t_p), \tau_{k_2}(t_{q+1}), ..., \tau_{k_2}(t_{|t_{k_2}|})\} \text{ and}$$
$$\tau'_{k_2} = \{\tau_{k_2}(t_1), ..., \tau_{k_2}(t_q), \tau_{k_1}(t_{p+1}), ..., \tau_{k_1}(t_{|t_{k_1}|})\}$$

Track Diffusion: Randomly select a track τ_k and an index d from $\{1, ..., |\tau_k|\}$ and update the position and orientation of the object $\tau_k(t_d)$ under some random noise.

Figure 4.13 illustrates the described dynamics moves of the Markov chain which we use to traverse the solution space. Four first types of moves are temporal dynamics and the last one is a spatial dynamics. The temporal moves help to form tracks (data association of object hypotheses over T frames) and the spatial move helps to improve the detection results (to compensate errors from the model-fitting step described in the subsection 4.3.3). At each iteration, one of the above dynamics is chosen randomly. Since the dynamics moves are stochastic and reversible, it is guaranteed that the Markov chain designed this way is *ergodic* (*i.e.*, any state is reachable from any other state within finite number of iterations) and *aperiodic* (*i.e.*, the Markov chain does not repeat in a fixed pattern). This ensures that the entire solution space can be explored if the MCMC sampler constructed this way is run with unlimited resources.

4.3.5 Incremental computation

For each MCMC move, we need to compute the ratio $\frac{\pi(\omega')}{\pi(\omega)} = \frac{P(\omega'|Z)}{P(\omega|Z)}$ in (4.20). In one iteration, our algorithm only changes maximum two tracks. Thus the new posterior probability can be computed more efficiently by incrementally computing it only within the related terms in (4.19). This is in contrast to the particle filters where the evaluation of each particle (joint state) needs the computation of the full joint likelihood. One more interesting property of the

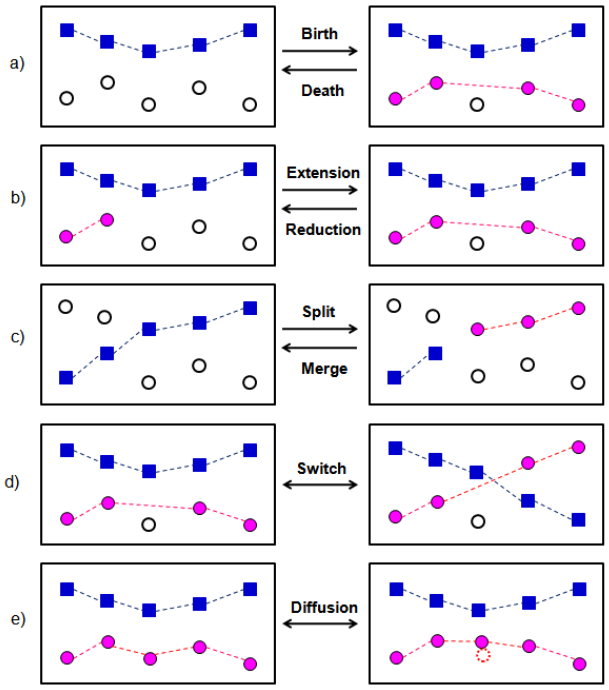


Figure 4.13: Dynamics moves of the Markov chain.

MCMC approach is that, we only needs to keep one hypothesis of object trajectory solution in the memory at one time instant compared with all solution hypotheses have to be maintained in case of tracking with MHT. Moreover, the execution time can be controlled by the number of sampling iterations n_{mc}.

4.4 Experimental Results

We test our approach to DATMO described above using the same Navlab dataset [Wang *et al.* 2004] as used in Chapter 3. Actually, our tracking process is performed for every four "original" scans which was collected at 37.5Hz so that the data cycle time from frame to frame in our program is about 100ms.

We have implemented the described algorithm as an online process within a sliding window which contains $T = 10$ frames and only moving object hypotheses generated within this sliding window are stored in the graph structure which is implemented as a rounded-list. For each time step when a new scan arrives, the trajectory solution obtained in the previous step is used to initialize a new MCMC

search (ω_0). Tracks are confirmed if their lengths are greater than or equal three (objects must be observed at least three times to confirm), otherwise they are considered false alarms. The MCMC sampler is run for a total of 300 iterations. The average computational time for a total detection and tracking process is about $60 - 80$ms on Pentium IV 3.0GHz PC with unoptimized codes so that it can fulfill the real time requirement.

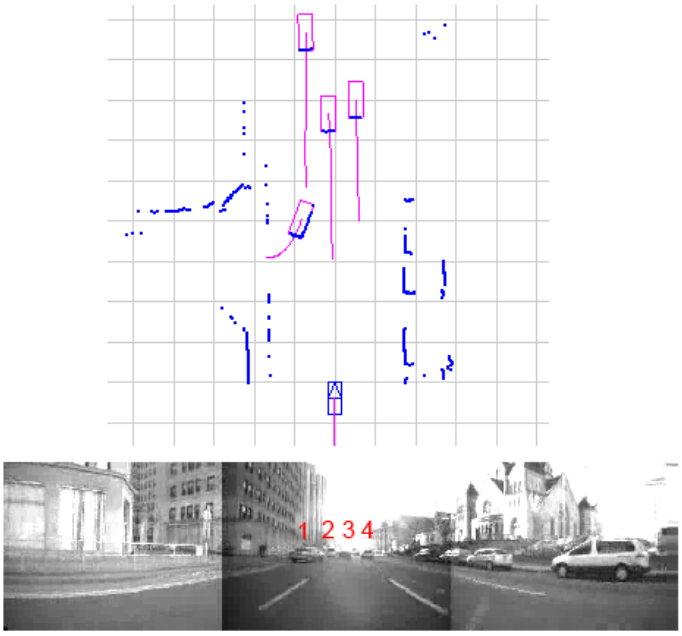
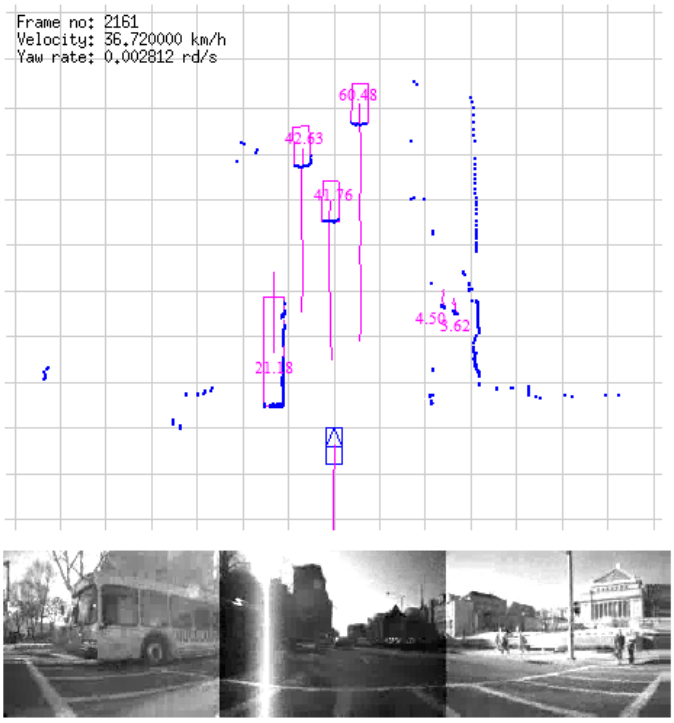


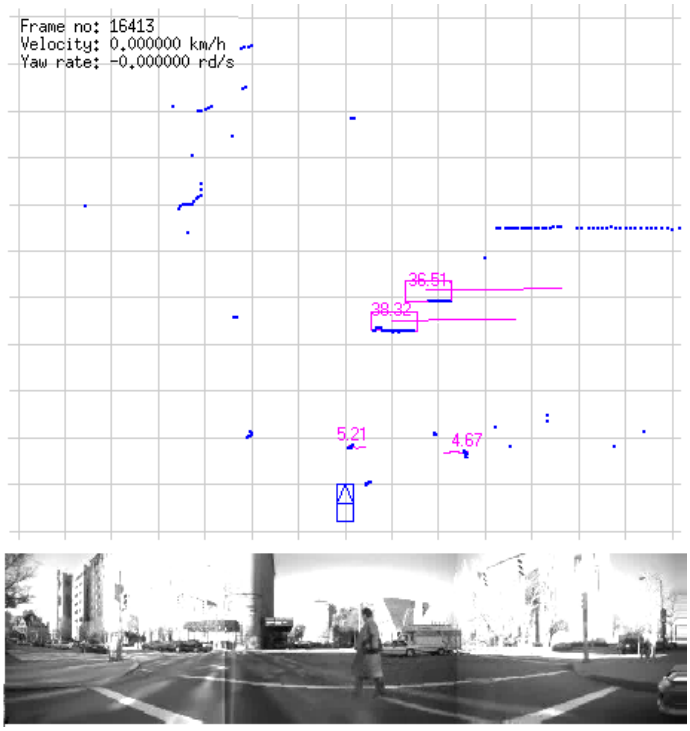
Figure 4.14: Result of searching the hypotheses space in Figure 4.11 for the solution.

Figure 4.14 shows the result after searching the moving object trajectories from the space of hypotheses in Figure 4.11. Four consistent tracks are found correctly corresponding to the four moving cars. False-alarm generated hypotheses are quickly discarded when not finding temporal neighbors satisfying the motion consistency condition.

Figure 4.15(a) shows an example of our detection and tracking algorithm in action. In the ego-vehicle's view, only current laser scan is displayed in blue color, moving objects detected and their trajectories are shown in pink color. Moving objects in the situation include a bus moving in the opposite direction on the left, three cars moving ahead, two pedestrians walking on the left pavement and the



(a) Different object classes are successfully detected and tracked.



(b) Example of tracking with occlusion.

Figure 4.15: Moving object detection and tracking in action.

other two pedestrians waiting at the intersection. The bus is divided into several segments and pedestrians are easy to be confused with noises and other small objects that make the detection and tracking challenging. However, thanks to our approach, the bus model helps connecting the discontinued segments and temporal information helps to distinguish pedestrians with noises and reduce ambiguities. All moving objects are identified and tracked successfully.

Figure 4.15(b) is another example of tracking with partial occlusions. The figure shows our method is able to detect two cars passing in front of the ego-vehicle where one is partially occluded by another from only one visible side of these two cars. Other two pedestrians are also detected and tracked.

Note that in our model-based approach moving objects are naturally classified. The readers can refer to: http://emotion.inrialpes.fr/~tdvu/videos/ for resulting videos.

4.4.1 Performance Evaluation

Since there are no ground-truth data for the testing dataset, we have tried to evaluate the DATMO performance manually. We choose some typical subsequences from the dataset and perform a frame-based evaluation. In each frame of data we labeled all moving objects identifiable by human eyes in the video sequence. For each moving object, we count how many frames it is detected and wrong detections (false positives) are also counted. The results of the evaluation are summarized in Table 4.1. Note that the maximum theoretically possible true positive rate is lower than 100% because at least three frames are required to detect a new moving object.

Table 4.1: Quantitative results

Sequence	Total Frames	Total Objects	Correct Detected	False Alarms	MP %	TP (%)	FP (%)
1	850	3850	3726	120	98%	96.7%	0.4%
2	405	1625	1580	67	98.2%	97.2%	0.4%
3	520	2510	2465	80	99.0%	98.2%	0.3%

We can see that in all sequences, high detection rates are achieved with

relatively low false positives (less than 0.4%). The result obtained thanks to the temporal information which helps to reduce the ambiguities and make the detection more robust.

4.5 Summary

In summary, we have presented a method for simultaneous detection and tracking moving objects (DATMO) in real time with a laser scanner using a Bayesian-MCMC approach. The success of the method described in this chapter lies in a general framework which integrates both top-down and bottom-up processing.

First, a top-down strategy is introduced to treat DATMO problem as interpreting moving object trajectories from a sequence of laser measurements. This allows explicitly incorporating various aspects including prior information, object model, object motion model and measurement model into a theoretically sound formulation. In theory, the optimum solution can be computed from the posterior probability. In practice, the computation is infeasible due to the large and complex solution space.

To make the search more efficiently, then we consider the detection result in Chapter 3 as a *coarse moving object detection*. This is used as bottom-up evidences to generate hypotheses about potential moving objects. And using results from bottom-up processing to guide proposal probabilities for the Markov chain in an MCMC computational engine both takes advantages of the computational efficiency of the bottom-up process and retains the optimality and robustness of the Bayesian formation from the global view (top-down).

Our approach in this chapter emphasizes on the use of object models to overcome existing problems of tracking using laser sensors. With the use of object models, segmented objects caused by laser discontinuities are no longer a problem and tracking results are more accurate. In our model-based approach moving objects are naturally classified.

The proposed algorithm is tested on a real life urban traffic dataset. With initial evaluations, the MCMC detection and tracking performs well in terms of high detection rates and low false alarms. In addition, different classes of moving objects are recognized successfully.

Chapter 5

Appplication: PreCrash

5.1 Introduction

In this chapter, we will present an integration of our perception module described in Chapter 3 on a real vehicle for a particular safety application, namely PreCrash. This work is conducted in collaboration with Daimler research department within the framework of the PReVENT-ProFusion project[1]. This project was aimed at promoting research activities in Europe to contribute to road safety by developing preventive safety applications and technologies.

The goal of the PreCrash system is to detect unavoidable collisions before they take place in order to trigger restraint systems to protect car passengers from accidents or mitigate the consequences. In this project, we focus on detecting frontal collisions with obstacles from a vehicle which is equipped a laser scanner and two short range radars. Since at the time carrying out the project, the DATMO module presented in Chapter 4 was not available, here we report results obtained using the algorithm presented in Chapter 3 to detect moving objects from laser data.

In PreCrash applications, we have to take stationary objects into account so that the algorithm is modified to detect both dynamic and static objects. For a more robust performance, we performed data fusion of detection results obtained by laser with data provided by radar. The perception output is a list of objects with their locations and estimated dynamic states. These information is then sent

[1] http://www.prevent-ip.org/profusion

to a risk assessment module developed by Daimler to compute the probability of a collision of the vehicle with the observed obstacles.

In the case a crash is predicted to happen, depending on the time to collision (TTC), the system will respond to the corresponding situations by activating appropriate actuators (see Figure 5.1). For instance, triggering of autonomous braking several hundred milliseconds before the collision would help to diminish its effects. Within the next $200-300$ milliseconds, reversible belt pretensioners are triggered to remove the belt slack in advance and bring the vehicle passengers into an upright position which keep them safer during the crash. In the last moment, irreversible preconditioned airbags are activated to maximize the capacity of protecting the driver and passengers from severe accidents.

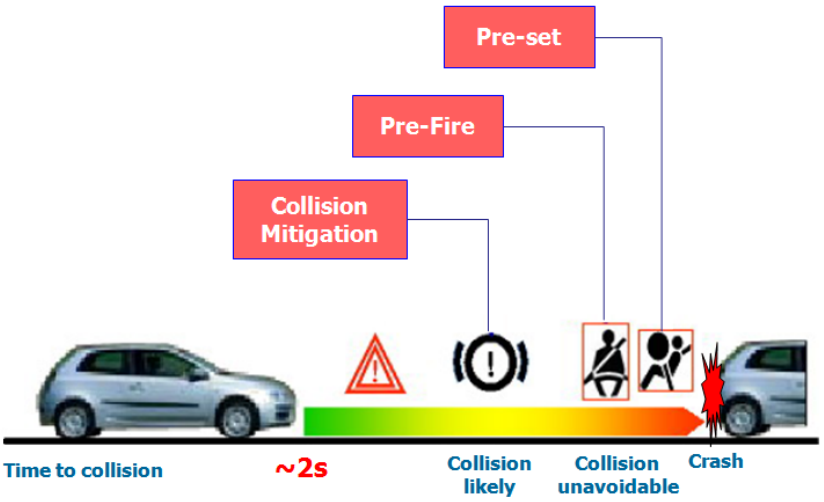


Figure 5.1: Pre-crash phases and functionalities of the PreCrash application.

In order to deploy appropriate actuators at the right time, a quick responding time requirement for the perception and risk assessment modules is very important. Surrounding objects and imminent collisions must be detected as fast as possible. The earlier a potential collision is detected, the more possibilities are available to protect vehicle passengers and other road users. The second requirement is the accuracy and reliability of the system which is critical especially for safety applications like the PreCrash system since incorrect functionalities are not acceptable for users in practice. Incorrect functionalities due to false alarms must be avoided since otherwise potentially dangerous

situations can be provoked, not to mention the loss of confidence and acceptance of such systems. On the other hand, missing collision detections make the system become useless. The final objective will surely be to reach a trade-off between false alarms and missed relevant dangerous situations.

In the following sections we will describe the design of PreCrash system on the real vehicle including both hardware and software architecture. Then we present in detail how the perception and collision detection modules actually work. We test the system extensively through various scenarios and demonstrate our system can fulfill the aforementioned requirements. The performance is compared to the results obtained with the perception module developed at Daimler using the same risk assessment module.

5.2 The Demonstrator Vehicle

The experimental vehicle, a Mercedes-Benz E-Class, is equipped with an Ibeo's ALASCA laser scanner mounted below the number plate and two M/A-COM SRS100 24 GHz short range radar prototypes mounted in the front bumper besides the number plate (Figure 5.2). The laser scanner is hermetically covered by a box having a black plastic faceplate which is transparent for the emission wavelength while the radars are mounted behind the serial plastic bumper. The technical specifications of the sensors are listed in Table 5.1.

Table 5.1: Technical Data of the Sensors

Property	Laser scanner	Short range radar
Angle	160°	80°
Angle accuracy	+/-0.5°	+/-5..10°
Range	0.3-80 m	0.2-30 m
Range accuracy	+/-5 cm	+/-7.5 cm
Scan frequency	25 Hz	25 Hz

The radar sensors and the laser scanner controller are connected to a controller unit in the trunk by private CAN and Ethernet, respectively. This real time unit hosts a 366 MHz Motorola Power-PC-processor which runs the software for sensor data processing and activation decision. In case of unavoidable collisions

the reversible seatbelt pretensioners of the front seats are deployed via a private CAN. An additional PC in the trunk acts as a display server connected to a monitor in front of the co-drivers seat to visualize the environment perception and the activation decision. The hardware architecture in the vehicle is shown in Figure 5.3.

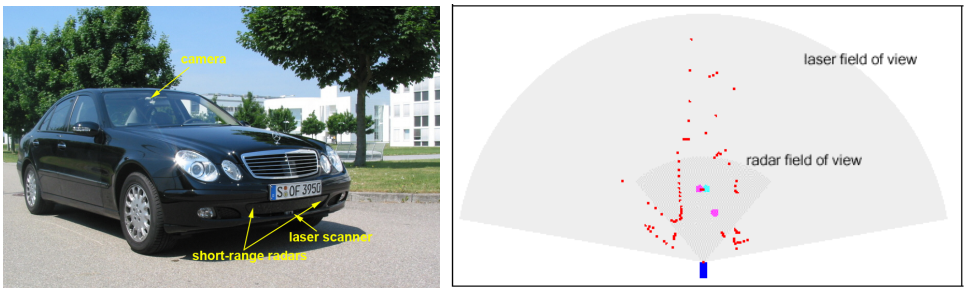


Figure 5.2: The Daimler demonstrator car and an example of sensor data.

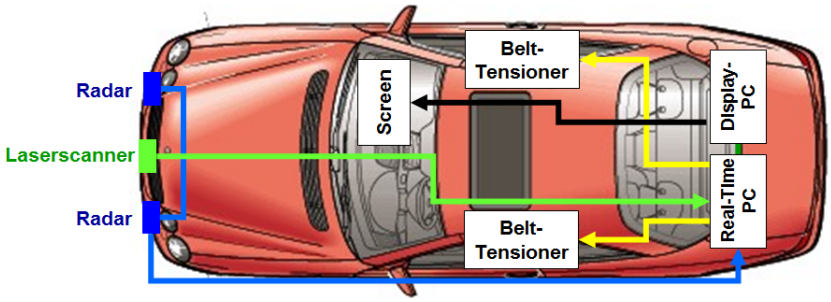


Figure 5.3: Hardware architecture of the experimental vehicle showing sensors, actuators, computers and interconnections.

Figure 5.4 shows a cutout of the screen exactly at the moment of deployment when the car approaches a foam cube with a constant speed of 50km/h. On the screen, the targets seen by the laser scanner and the radars are shown as small dots and circles, respectively. The colors symbolize the mounting side of the radars and accordingly the four vertical beam layers of the laser scanner. Object segments, generated from the scanner targets, are depicted as rectangles. The actual time to collision (TTC) of 174ms corresponds to a distance of 2.4m. The inset of the figure shows the picture of corresponding situation captured by the in-vehicle camera behind the windshield.

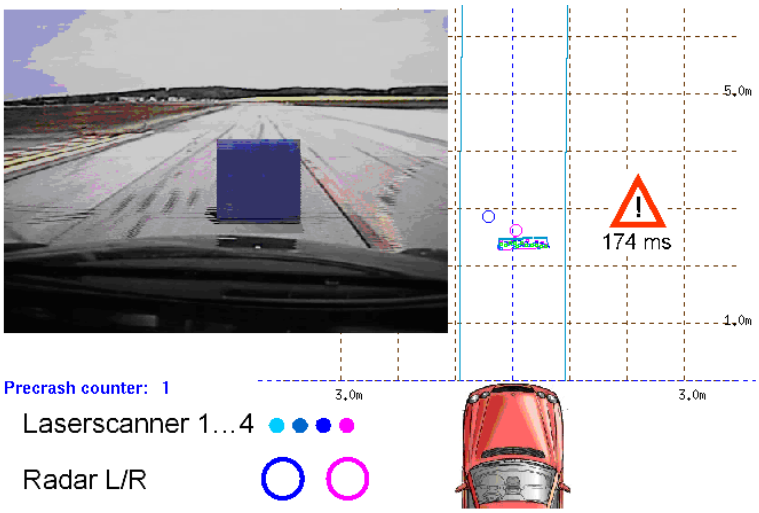


Figure 5.4: Visualization of the environment perception on the in-vehicle screen. The inset shows the scene recorded by the camera behind the windshield.

5.3 PreCrash System

As shown in Figure 5.5, the PreCrash system is comprised of three interconnected modules: perception, decision and action. In the first step, the perception module performs data processing from separate sensor sources and their results are fused to give a model of the vehicle environment. Output of this part is an estimated state of the host vehicle as well as a description of the surrounding environment comprised of static and moving objects and their estimated corresponding dynamic states of each object (e.g: position, velocity, moving direction). Based on these information, in the second step, the decision module will estimate whether a collision is likely going to take place within the next several hundred milliseconds. In the case there is an inevitable collision, the decision module drives the action module that physically activates appropriate actuators like: brakes, belt pretensioners, airbags... in order to reduce the injures of the crash.

We can see in the architecture, the collision detection module plays a very important role deciding the performance of the PreCrash system. And without a reliable perception module, a good performance of collision detection cannot be achieved. In the following we will describe in detail these two modules.

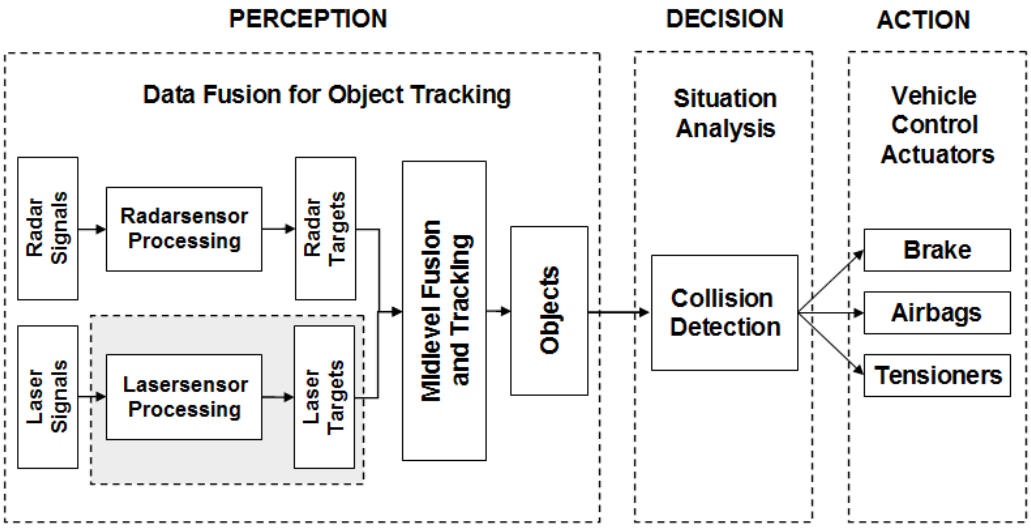


Figure 5.5: The PreCrash system architecture.

5.3.1 Data Fusion for Object Tracking

The perception sensors include one laser scanner and two short range radars. While the scanner provides raw data as a list of laser impacts, the radars provide high-level data as a pre-filtered list of moving objects. Here we adopt the perception module presented in Chapter 3 to extract objects from laser data. Results obtained are fused with radar data at object-level in order to provide a more reliable performance.

Object Extraction from Laser Data

Using the algorithm described in Chapter 3, moving objects are detected whenever a new scan measurement is received based on the local grid map constructed from previous measurements. We implement a simple tracking using GNN and Kalman filter to track detected objects.

Concerning static objects, we can take the grid map as a snapshot of stationary obstacles and then can use any image segmentation algorithm to extract these objects. However, image segmentation is not a trivial problem and might be costly in terms of computation time.

Here we opt for an alternative method which extracts static objects only

from the latest scan. After detecting moving objects, we filter out dynamic measurements and perform a segmentation over the remaining measurements. The returned results is a list of segments and we consider each segment as a static object. At an instant time, these object-segments do not necessarily reflect actual forms of the real objects. However, for our PreCrash application, we only care about frontal collisions and normally the sensor range is wide enough to cover collidable parts of obstacles which is reflected in the current scan.

Fusion with Radar

After objects are extracted from laser data, we confirm the moving object detection results by fusing with radar data. Since data returned from radar sensors are pre-filtered as lists of potential targets, each target in the lists is provided with information about the location and the estimated Doppler velocity, the data fusion is performed at object-level.

For each moving object detected by laser as described previously, a rectangular bounding box is calculated and the radar measurements which lie within the box region are then assigned to corresponding object. The velocity of the detected moving object is estimated as the average of these corresponding radar measurements.

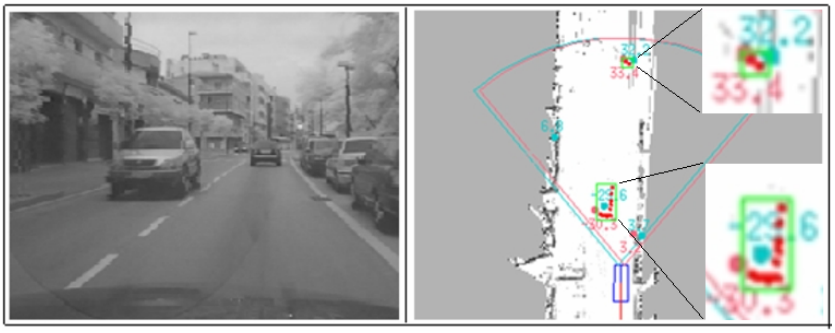


Figure 5.6: Moving object detected from laser data are confirmed by radar data.

Figure 5.6 shows an example of how the fusion process takes place. Moving objects detected by laser data are displayed as dots within bounding boxes. The targets detected by two radar sensors are represented as circles along with corresponding velocities. We can see in the radar field of view, two objects detected by laser data are also seen by two radars so that they are confirmed.

Radar measurements that do not correspond to any dynamic object or fall into other region of the grid are not considered.

An example of fusion results obtained with the perception module is shown in the Figure 5.7. All objects are detected and related information are displayed.

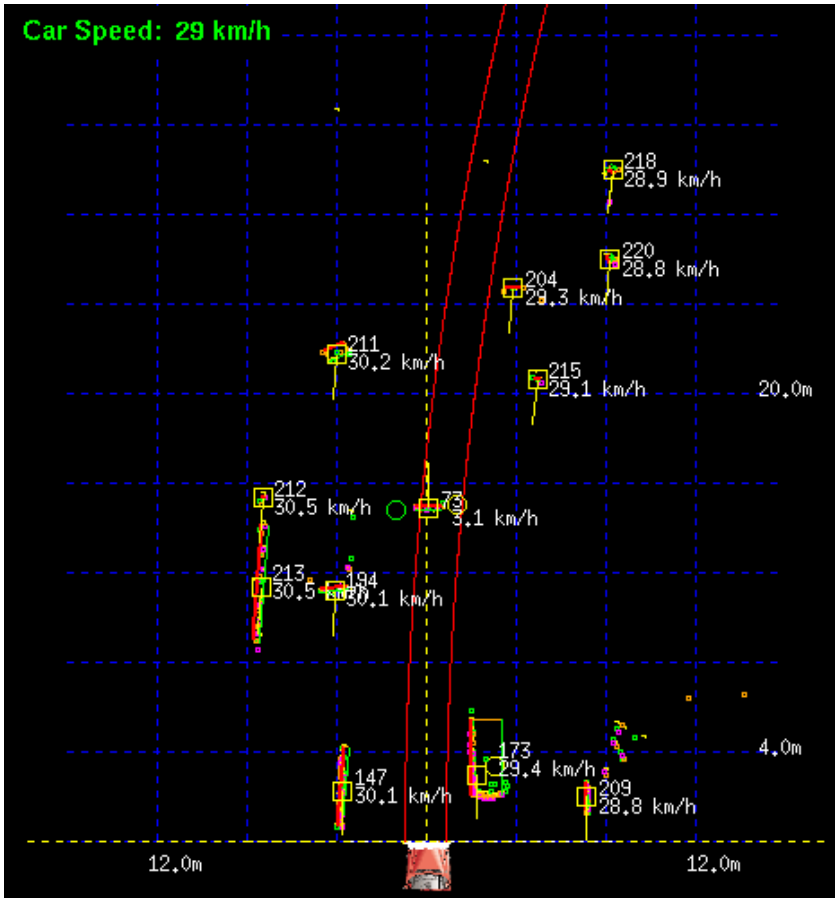


Figure 5.7: Interface of the perception module in action. The laser data is displayed as small colored points. Radar data is represented by small circles: left in green and right in yellow. Obstacles surrounding of the ego-vehicle are detected and represented by bounding-boxes of segments whose centers of gravity are highlighted by yellow squares. Note that all static and dynamic objects are all detected and their velocities plus movement directions *relative* to the car are computed and displayed next to each object along with objects ID.

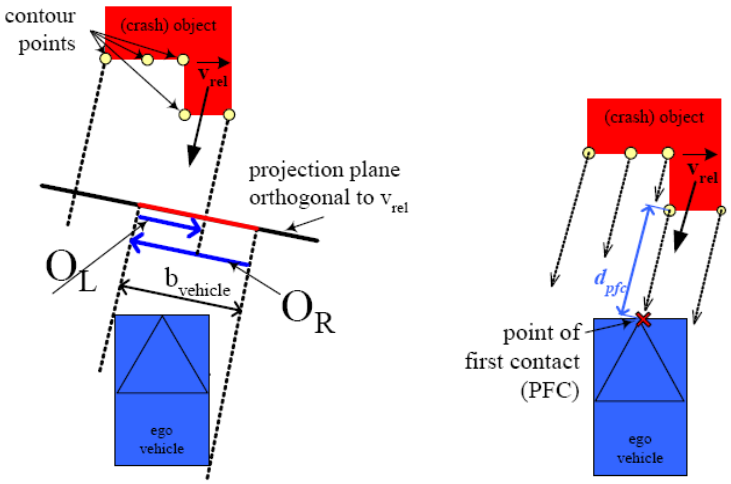
5.3.2 Situation Analysis and Decision

The algorithm for situation analysis and decision is developed at Daimler and is independent from the perception module. Its objective is to detect unavoidable crash situations given a description of the vehicle environment provided by the perception module. The perception module delivers a description of the environment by means of a list of objects with estimated information about their position, speed, movement direction and from which sensor(s) they are observed. Subsequent steps calculate for all these objects whether they potentially hit the vehicle according to the estimation of their movement and the time to collision (TTC), if applicable.

For each object to be analyzed, we calculate the overlap O of a potential crash with regard to the ego vehicle by:

$$O = \frac{min(O_L, O_R) + \frac{\Delta b}{2}}{b_{vehicle} + \Delta b} \tag{5.1}$$

The measured object width as well as the width of the ego vehicle is enlarged by a safety factor Δb respectively $\frac{\Delta b}{2}$ (Figure 5.8).



(a) Overlap calculation (b) TTC calculation

Figure 5.8: Collision Detection

Since all objects are described by contour points. The point of first contact (PFC) is the contour point of the object with the shortest distance to the outline of

the ego vehicle. From the PFC calculation, the hit point distance from the object to our vehicle is known. The time to collision (TTC) is now simply given by:

$$TTC = \frac{d_{PFC}}{v_{rel}} \tag{5.2}$$

Then the crash probability is calculated from the TTC together with the overlap by:

$$CP = \frac{O}{TTC - \delta t} \tag{5.3}$$

δt is a user defined parameter for a delay time, which can be adjusted by the OEM to meet the requirement for a specific safety system (e.g. safety belt pretensioner).

Besides the collision detection at an instant time, the situation analysis stage is also based on a data history collected for each object during its life time. In this step, a pre-selection is made between objects, that will potentially hit the ego vehicle and those that are most likely not hazardous or exceedingly unconfident. Only potentially dangerous objects are considered in the decision step. The most important criterion is the time to collision. Objects that reach the decision step have a calculated time to collision within the time frame of 200ms, that is relevant for the application. For a robust system behavior, further attributes of an object are inspected to ensure their reliability. Objects with following attributes are rejected:

- calculated point of impact is located outside the forefront of the car
- object's state is not "confirmed"
- velocity (relative to sensor vehicle) too small
- object is near the border of the field of view
- too high variation of velocity and/or acceleration over time

Nevertheless, uncertainties remain due to noise in measuring and preprocessing and simplifying model assumptions. Another aspect is that any kind of sensor may deliver so-called ghost targets that do not correspond to any real-existing object. Therefore, in the decision step we have to deal with two questions:

- Will we really collide with the object?
- Does the object really exist?

For answering the first question, a Bayesian classifier is applied. Let K be the event "object collides" with the probabilities $P(K) + P(\neg K) = 1$. Then, the probability of a collision given a certain measurement z is:

$$P(K|z) = \frac{P(K,z)}{P(z)} \tag{5.4}$$

Applying Bayes rule, this is the same as:

$$\begin{aligned} P(K|z) &= \frac{P(z|K)P(K)}{P(z)} \\ &= \frac{P(z|K)P(K)}{P(z|K)P(K) + P(z|\neg K)P(\neg K)} \end{aligned} \tag{5.5}$$

where z is composed of different attributes z_i : the variance of the x-component of the velocity, the lifetime of the object and the number of cycles the object was categorized as critical. To judge the criticality of an object, it is inspected within a determined time period that is longer than the time to collision. Assuming independent attributes z_i and, furthermore, K and $\neg K$ equiprobable, the probability of a collision can be calculated using 5.6.

$$P(K|z) = \frac{\prod P(z_j|K)}{\prod P(z_j|K) + \prod P(z_j|\neg K)} \tag{5.6}$$

The conditional probabilities $P(z_j|K)$ and $P(z_j|\neg K)$ are determined beforehand in an off-line procedure by inspecting numerous examples with different situations. Finally, an object is considered as crashing object if $P(K|z)$ exceeds a predefined threshold.

Beside the probability of a collision, the probability of existence has to be considered in order to prevent false alerts that may arise from ghost targets delivered by the sensors or failures in associating measurements to objects. We use a method based on evidence theory introduced by Dempster and Shafer [Dempster 1968, Shafer 1976].

For a classification of existing and non-existing objects we define the hypothesis space as $\Theta = \{E, \neg E\}$ where E stands for "object exists" and $\neg E$ stands for "object does not exist". In evidence theory the power set $2^\Theta = \{\emptyset, E, \neg E, E \cup \neg E\}$ is considered. Sensor-specific mass functions assign probability masses

to the elements in the power set. For the laser scanner, the mass functions are implemented as:

$$\begin{aligned} m_l(E \cup \neg E) &= c_l \\ m_l(E) &= \frac{\sum_{i=0}^{N} 2^{N-i} \cdot h_l[i]}{\sum_{i=0}^{N} 2^{N-i}} \cdot (1 - m_l(E \cup \neg E)) \\ m_l(\neg E) &= 1 - (m_l(E) + m_l(E \cup \neg E)) \end{aligned} \tag{5.7}$$

By definition, $m_l(\emptyset) = 0$. The constant term c_l denotes a mass probability for uncertainty. The mass function for the hypothesis "object exists" considers the weighted ratio of the number of detections to the lifetime of an object within a given time frame N. In this connection, $h_l[i]$ contains the information about the object being detected by the laser scanner at time i. Younger data is exponential higher weighted than older data. The remaining mass for the hypothesis "object does not exist" is derived from the condition

$$\sum\nolimits_{X \subseteq \Theta} m(X) = 1 \tag{5.8}$$

Mass functions for radar sensors are implemented in an analogous way. The fusion of masses from the different sensors is performed in two steps. First, the masses from the two radar sensors are combined. Second, the resulting radar masses are combined with the masses calculated for the laser scanner. For fusion, Dempster's combination rule is used [Dempster 1968].

The final step of the decision module combines the probability of collision that is provided by the Bayes classifier with the probability of existence. For the Bayes classifier we define the hypothesis space $\Theta = \{C, \neg C\}$ with the hypothesis C for a colliding object and $\neg C$ for a non-colliding object. The probability masses $m_b(C)$ and $m_b(\neg C)$ are directly taken from the conditional probabilities for K (see 5.6).

$$\begin{aligned} m_b(C) &= P(K|z) \\ m_b(\neg C) &= P(\neg K|z) = 1 - m_b(C) \\ m_b(C \cup \neg C) &= 0 \end{aligned} \tag{5.9}$$

In this case, the uncertainty $C \cup \neg C$ is equal to zero, because Bayesian

probabilities do not provide a measure for uncertainty. For the interesting case "object exists and collides" the combined probability mass results in

$$m_f(E \cap C) = m_b(C) \cdot (m_c(E) + m_c(E \cup \neg E)) \tag{5.10}$$

If m_f exceeds a predefined threshold, actuators are triggered.

In general, laser measurements are able to describe the position and shape of real existing objects very accurately. Radar sensors help to suppress ghost targets or targets based on objects that are irrelevant for PreCrash applications like plants or steam coming out of street drains. All in all, the presented PreCrash system based on a laser scanner fused with short range radars reliably detects different kinds of collisions with obstacles in front of the car, as our evaluation in the next section shows.

5.4 Experimental Results

The application has been validated in complex crash and non-crash scenarios. To conduct the experiments, we built up a comprehensive database that consists of short sequences of measurements recorded during predefined driving maneuvers. These are comprised of actual and nearly-missed collisions with objects at different velocities, in curves, sudden lane changes and lane changes of a leading target vehicle blocking the sight to the obstacle. For the maneuvers, foam cubes and cylinders served as crash objects.

Figure 5.9 shows some scenarios in the test track. In Figure 5.9(a) is the situation the car is passing an artificial gate without collision. Figure 5.9(b) sees a wheel rolling out into the test track when the car is moving. In Figure 5.9(c), the test vehicle follows another car preventing it from seeing another obstacle until the leading car suddenly changes the lane. In all situations, the obstacles are detected and tracked successfully and we display their velocities plus moving directions estimated *relatively* to the ego-vehicle (yellow lines).

For the performance evaluation, we count the false alarms (false negative) that occurred in non-crash scenarios and the missing detections (false positive) in case a collision was not detected by the system. We compare the results obtained using our perception module with results obtained using the perception module

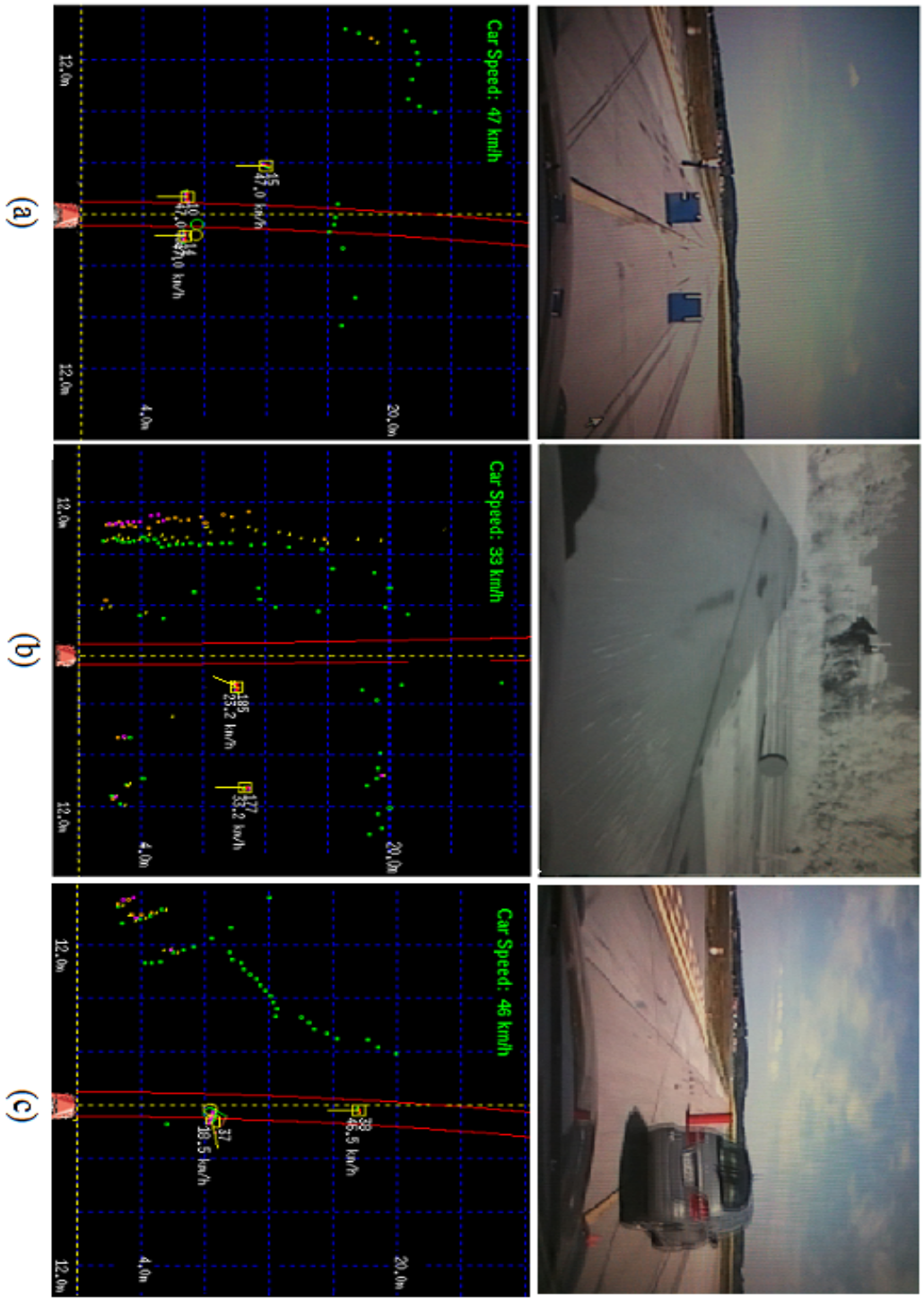


Figure 5.9: Some test scenarios. (a) the car passing an artificial gate without collision; (b) a wheel is rolling out in front of the car; (c) the leading car changes its moving direction to show an obstacle appeared suddenly. Note that the velocities of obstacles is computed relatively to the ego-vehicle.

Table 5.2: Results for Complex Non-Crash Scenarios

Scenario	Ego velocity [km/h]	No. of tests	False alarms/False alarm rate	
			DC module	Our module
Near-missed passing of cylinder	40, 60	9	0 / 0%	0 / 0%
Near-missed passing of cube	40, 60	6	0 / 0%	0 / 0%
Near-missed passing of cylinder after curve (45°)	40, 60	29	0 / 0%	3 / 10.3%
Emergency brake, distance to cylinder after brake not greater than 1.5 m	40, 60 (at start)	19	1 / 5.3%	1 / 5.3 %
Lane change maneuver to avoid a collision with a cube	30, 40, 50, 60, 70	22	0 / 0%	0 / 0%
Gate passing	30, 50	6	0 / 0%	0 / 0%
Gate passing after curve (45°)	30, 50	4	0 / 0%	0 / 0%
Total		**95**	**1 / 1.1%**	**4 / 4.2%**

Table 5.3: Results for Complex Crash Scenarios

Scenario	Ego velocity [km/h]	No. tests	Missings/Missing rate	
			DC module	Our module
Collision with cylinder, varying points of impact	20, 40	24	0 / 0%	0 / 0%
Collision with (paper) cylinder at high speed, varying points of impact	60, 120	8	2 / 25.0%	0 / 0%
Collision with cube, point of impact has high offset	40	7	0 / 0%	1 / 14.3%
Collision with cylinder after curve (30°, 45°)	30, 40, 60	20	2 / 10.0%	0 / 0%
Collision with cylinder or cube after emergency brake	20, 40 (at crash time)	7	0 / 0%	0 / 0%
Collision with (paper) cylinder after emergency brake at high speed	60, 80 (at crash time)	9	2 / 22.2%	0 / 0%
Collision with cylinder after lane change maneuver	40, 50	23	1 / 4.3%	1 / 4.3%
Collision with cylinder after leading car lane change	40, 50	4	0 / 0%	0 / 0%
Total		**102**	**7 / 6.9%**	**2 / 1.9%**

developed at Daimler (called DC module). Table 5.2 shows the results for the non-crash scenarios for the two different modules and Table 5.3 lists the results for the crash scenarios.

As a general result it can be stated that a reliable collision detection is achieved with both perception modules. Whereas DC module enables a lower false alarm rate, the crash detection rate of our module is very high (98.1%). The three false alarms in the scenario where we pass the cylinder in a curve occurred in cases of getting extremely close to the obstacle. In contrast, no false alarms occurred at all when the subject vehicle suddenly changes the lane to avoid a collision with an obstacle standing on the road. Emergency brake maneuvers challenge the tracking system because of the divergent motion scheme. In our evaluation, only 1 out of 19 test drives resulted in a false alarm for each module.

In motion estimation, there is always a trade-off between stabilization of the current state and the adaptation to dynamic situations. It becomes apparent when looking at the scenarios where the system fails. In case of cornering, for example, the direction of the obstacle's relative movement continuously changes. From the results in curve scenarios, it can be seen, that the two modules handle such situation in a different way. Our module produces more false alarms, whereas DC module risks more missing detections. Looking at Table 5.3, missing detections provoked by DC module are overrepresented in high speed scenarios. An object with a high relative velocity is registered infrequently during the time period available for creating a data history for this object as described in subsection 5.3.2. In this case, the decision is supported by less data. Depending on the influence of new measurements on the current state, the grid map approach may be advantageous over a single frame processing realized in DC module, in this special case. In general, it should be highlighted that a lot of the test maneuvers have been performed at the vehicle dynamics limit.

In addition, we tested the application in normal traffic on highways, rural roads and in urban areas. To achieve representative results we performed the test drives during day time to cover different traffic situations like rush hour, traffic jam and stop-and-go. Furthermore, the test drives were partly conducted under adverse weather conditions like rain, fog, wet roads and traffic spray. All in all, we covered a distance of 1600km, running the application in real time. This test was performed where there were no wrongly detected collisions in any of these

environments.

To demonstrate that our perception algorithm can fulfill the real-time requirement, we evaluate the computational cost for each test scenario and show the accumulated results in Figure 5.10. We can see that the performance of our module is comparable with the module developed by Daimler. In every cases, the computational time required is less than 20ms which is totally fit in the 40ms data time cycle of the whole system.

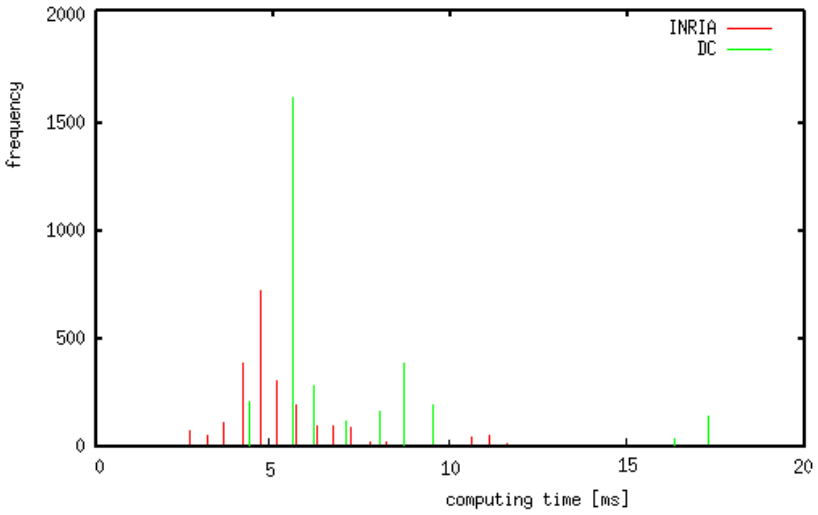


Figure 5.10: Comparison of computational time required for both perception modules.

5.5 Summary

In this chapter, we present an integration of our perception module with a specific PreCrash application on a real vehicle which is equipped with a laser scanner and two short range radars. The perception module performs data processing and object extraction using data fusion from laser and radars. A model of the environment obtained in terms of static and moving objects serves as a basis for the safety system that trigger restraint systems in case an unavoidable collision takes place.

We compared the performance using our perception module with results obtained with the perception module developed at Daimler. Comprehensive tests

show, that a good detection performance for frontal collisions is achieved for both modules. Comparing the results of both approaches, the sums of false and missed alarms balance each other. The application was running stable in a hard real-time environment and has been extensively tested in real traffic scenarios and with artificial crash and near crash maneuvers carried out on test tracks. The function has been successfully demonstrated in a public event during the PReVENT IP Exhibition in Versailles 2007 and at the IEEE Intelligent Vehicles Symposium, in Eindhoven 2008.

The PreCrash system described in this chapter is developed within context of the PReVENT-ProFusion project, which was set up as a research project, are not expected to be available on the market on the short run. However, the gained experience and knowledge will be further exploited by the involved partners in different domains: our perception results can be used as a general base for numerous driver assistant or driver information systems where the perception of the vehicle environment by means of sensor systems plays a crucial role. On the other hand, the prototype system developed in ProFusion are still some time away from market introduction. Hence, an important future task of OEMs and suppliers will be to reduce costs and to increase performance and robustness of such systems in order to commercialize them and bring their benefit in terms of increased safety directly to the customers and road users.

Chapter 6

Conclusions and Perspectives

6.1 Conclusions

In this thesis, we have studied core tasks of the vehicle perception problem including localization, mapping (SLAM) with detection and tracking of moving objects (DATMO) in context of dynamic environments. Particularly, we have focused on using a laser scanner as the main perception sensor. Being prerequisites for driving assistant systems and autonomous navigation systems, the vehicle perception plays a very important role since any wrong information perceived from the environments will affect the performance of the whole system. Keeping this criterion in mind with an objective to obtain a fast, robust and reliable solution to these essential perception tasks, our approaches have been presented, tested though various off-line datasets and demonstrated on real platforms to show that the requirements for real applications can be achievable. We summarize contributions of the thesis in the following.

In Chapter 3 we described a grid-based algorithm to solve SLAM with detection of moving objects. In the literature, SLAM algorithms have got to a mature state but traditional approaches to SLAM usually assume that the environments are static. To deal with dynamic environments, we propose to solve both SLAM and detection of moving objects simultaneously and show that results from the moving object detection step help to filter out spurious objects resulting in a better map. The key point of our approach lies in a new grid-based scan matching technique that works fast and quite robust in the presence of dynamic entities. This allowed obtaining a precise vehicle localization in order to build a

consistent map of the environment. After a consistent map is constructed, moving objects can be detected reliably without a prior knowledge of detected objects. Experiments on real-life traffic data have shown that our proposed algorithms can successfully perform a real-time mapping with moving object detection from a vehicle moving at high speeds in different environments.

Chapter 4 follows the results in Chapter 3 where we focused on problems of detection and tracking moving objects with the assumption that a good vehicle localization is obtained. With the detection algorithm presented in Chapter 3, moving objects are identified irrespective of its type and objects are represented as free-form by a cluster of points. We showed that tracking objects using free-forms leads to a degraded result and proposed to use model-based approach to represent objects. Fortunately, number of moving object classes appearing on roads are quite limited and we classified them into several categories such as: buses, cars, bikes and pedestrians.

Different from most previous works, we tackle the detection and tracking as a whole process. We take a top-down approach and try to interpret the laser measurement sequence with object models (shape) and their trajectories (motion). The approach follows a Bayesian formulation and solution is sought by computing the maximum of the posterior probability in a joint multiple object-trajectory space. The computation is generally intractable in such a complex solution space and we employ a Markov chain Monte Carlo (MCMC)-based method. We design a reversible Markov chain to explore the solution space in which detection results from Chapter 3 provide evidences (so-called data-driven or bottom-up techniques) to make the top-down search more efficient than traditional MCMC. This Bayesian-DDMCMC approach is more general and can successfully handle the ambiguities in the presence of persistent occlusions. The proposed algorithm is tested on challenging urban traffic datasets and initial evaluations showed promising results.

In Chapter 5, we showed how our perception module actually works on a real platform. We present the integration of our module for a specific PreCrash application on the real car. This is carried out in collaboration with Daimler within the context of European project PReVENT-ProFusion. Extensive tests on variety of scenarios demonstrated that a good detection performance for frontal collisions is achieved thanks to a reliable perception module. Although the scenarios for

testing were still simple but it is shown that our perception module can fulfill the real-time requirements.

To summary, the thesis is aimed at developing an efficient perception system for vehicles and we have demonstrated that we are able to perform simultaneous localization, mapping with detection, classification and tracking of moving objects in real-time from a ground vehicle moving at high speeds in urban environments. We hope that the obtained results will open a wide range of potential safety and convenient automobile applications as well as serve as basis for pursuing the dream of building autonomous vehicles.

6.2 Perspectives

The thesis raises several interesting topics and opens a number of possible extensions for the future works in order to improve the performance of the algorithms to the vehicle perception problem.

Firstly, a possible extension is related to the DATMO algorithm presented in Chapter 4. Although showing significant speedup compared to uninformed MCMC, the current computation is still heavy for complex scenes. An optimization of code is ongoing and our objective is to reduce the required computational time to below 40ms. Additional optimization can be made by integrating a road border detection from laser data and use the obtained result as *a prior* knowledge about the road regions. The idea is that this information gives more constraints about the appearances and trajectories of moving objects in the scene, for example: moving objects should be appeared in the road regions and vehicles should be moving along the road direction. These constraints helps to reduce significantly the solution space and thus accelerate the search. Moreover, a fusion of mapping, road detection, object tracking and object classification certainly will enable a better understanding of driving situations. (Figure 6.1).

Secondly, we would like to extend our perception module currently implemented in the PreCrash application with the DATMO algorithm in Chapter 4 in the hope that the collaboration with Daimler will be continued. We believe that the more meaningful representation of detected moving objects with specific shapes and the better estimation of their behaviors certainly improve the detection of collisions with moving objects. In addition, the better estimated dynamics

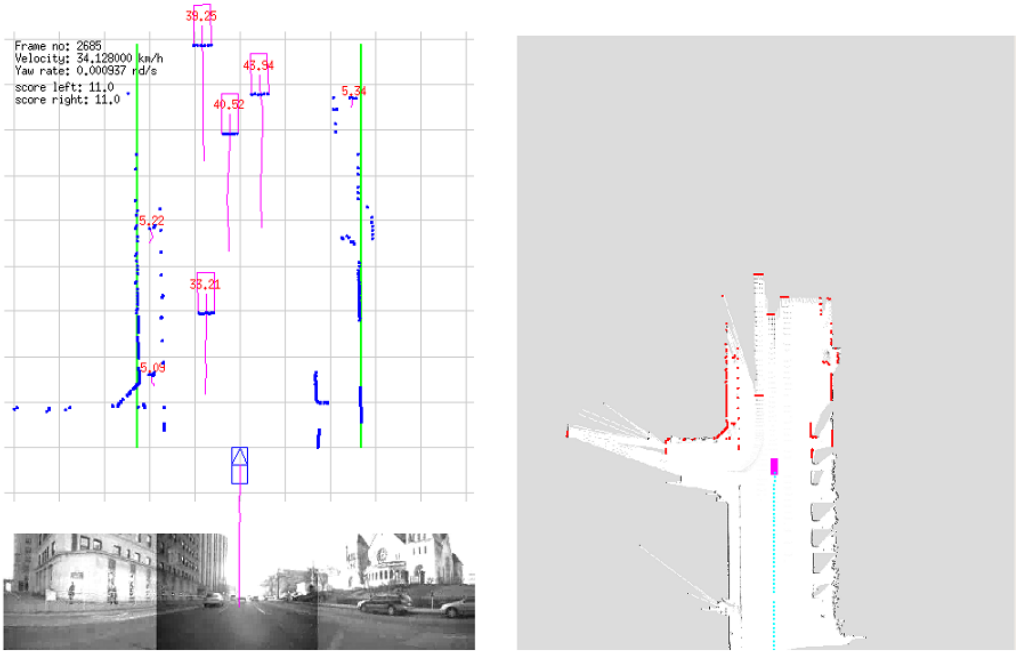


Figure 6.1: In all, a combination of vehicle map, road detection, moving object tracking and classification certainly will enable a better understanding of driving situations.

of moving objects allows more accurate predictions of their future behaviors enabling a wider range of activation decision (e.g. 200ms to several seconds).

Finally, the laser-based DATMO algorithm presented in this thesis can be combined with vision sensors to improve the performance. Since images contain rich information and compensate for some of the disadvantages of laser sensors. There are a number of ways to improve DATMO performance using state-of-the-art algorithms from the computer vision literature. For example, a reliable pedestrian detection using laser scanners is difficult to be obtained because the number of measurement points associated with a pedestrian is often small. Vision-based recognition algorithms can be used to confirm the results of laser-based detection. Because only portions of the image with high likelihood have to be processed and range measurements from laser scanners can be used to solve the scale issue, the recognition process can be speeded up and run in real-time.

Bibliography

[Anderson & Moore 1979] B.D. Anderson and J.B. Moore. Optimal filtering. Prentice-Hall, Englewood Cliffs, New Jersey, 1979.

[Arulampalam *et al.* 2002] S. Arulampalam, S Maskell, N Gordon and T Clapp. *A Tutorial on Particle Filter for Online Nonlinear/Non-Gaussian Bayesian Tracking*. IEEE Transactions on Signal Processing, vol. 50, no. 2, 2002.

[Bar-Shalom & Fortman 1988] Y. Bar-Shalom and T.E. Fortman. Tracking and data association. Academic Press, 1988.

[Blackman & Popoli 1999] Samuel Blackman and Robert Popoli. Design and analysis of modern tracking systems. Artech House, Norwood, MA, 1999.

[Blackman 2004] S. S. Blackman. *Multiple hypothesis tracking for multiple target tracking*. IEEE Aerospace and Electronic Systems Magazine, vol. 19, no. 1, pages 5–18, Jan 2004.

[Blom & Bar-Shalom 1988] H. A. P. Blom and Y. Bar-Shalom. *The interacting multiple model algorithm for systems*. IEEE Trans. On Automatic Control, vol. 33(8), 1988.

[Burlet 2007] Julien Burlet. *Suivi Multi-Objets Adaptif: Application a la Classification de Comportements de Mobiles*. Phd thesis, Institut National Polytechnique de Grenoble, Grenoble, France, December 2007.

[Choset & Nagatani 2001] H. Choset and K. Nagatani. *Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization*. Robotics and Automation, IEEE Transactions on, vol. 17, no. 2, pages 125–137, 2001.

[Christensen 2002] Henrik I. Christensen. Lecture notes: Slam summer school. 2002.

[DARPA 2007] DARPA. *Urban Challenge*, 2007. http://www.darpa.mil/grandchallenge/index.asp.

[Dellaert *et al.* 1999] Frank Dellaert, Wolfram Burgard, Dieter Fox and Sebastian Thrun. *Using the Condensation Algorithm for Robust, Vision-based Mobile Robot Localization*. CVPR, June 1999.

[Dempster 1968] A. Dempster. *A Generalization of Bayesian Inference*. Journal of the Royal Statistical Society, vol. 30, pages 205–247, 1968.

[Dietmayer *et al.* 2001] K. C. J. Dietmayer, J. Sparbert and D. Streller. *Model-Based Object Classification and Object Tracking in Traffic Scenes from Range-Images*. In Proceedings of the IEEE Intelligent Vehicle Symposium, Tokyo, 2001.

[Einsele & Farber 1997] Tobias Einsele and Georg Farber. *Real-Time Self-Localization in Unknown Indoor Environments using a Panorama Laser Range Finder*. In In IEEE/RSJ International Workshop on Robots ans Systems, IROS 97, pages 697–703. IEEE Press, 1997.

[Elfes 1989] A. Elfes. *Occupancy grids: a probabilistic framework for robot percpetion and navigation*. PhD thesis, Carnegie Mellon University, 1989.

[Elfes 1992] A. Elfes. *Multi-source spatial data fusion using bayesian reasoning*. Data Fusion in Robotics and Machine Intelligence, page 137163, 1992.

[Forsyth & Ponce 2002] David A. Forsyth and Jean Ponce. Computer vision: A modern approach. Prentice Hall, August 2002.

[Fox *et al.* 1999a] D. Fox, W. Burgard and S. Thrun. *Markov Localization for Mobile Robots in Dynamic Environments*. Journal of Artificial Intelligence Research, vol. 11, 1999.

[Fox *et al.* 1999b] Dieter Fox, Wolfram Burgard, Frank Dellaert and Sebastian Thrun. *Monte Carlo Localization: Efficient Position Estimation for Mobile Robots*. In Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99)., July 1999.

[Gambino *et al.* 1997] F. Gambino, F. Ulivi and M. Vendittelli. *The transferable belief model in ultrasonic map building*. In Sixth IEEE International Conference on Fuzzy Systems, page 601608, 1997.

[Hähnel *et al.* 2003a] D. Hähnel, D. Schulz and W. Burgard. *Mobile Robot Mapping in Populated Environments*. Advanced Robotics, vol. 17, no. 7, pages 579–598, 2003.

[Hähnel *et al.* 2003b] D. Hähnel, S. Thrun, B. Wegbreit and W. Burgard. *Towards Lazy Data Association in SLAM*. In Proc. of the International Symposium on Robotics Research (ISRR), 2003.

[Hähnel *et al.* 2003c] D. Hähnel, R. Triebel, W. Burgard and S. Thrun. *Map Building with Mobile Robots in Dynamic Environments*. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2003.

[Hähnel *et al.* 2003d] Dirk Hähnel, Wolfram Burgard, Dieter Fox and Sebastian Thrun. *An Efficient FastSLAM Algorithm for Generating Maps of Large-scale cyclic environments from raw laser range measurements*. In In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS, pages 206–211, 2003.

[Julier & Uhlmann 1997] S. Julier and J. Uhlmann. *A new extension of the Kalman filter to nonlinear systems*. In Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL, 1997.

[Kalman 1960] R.E. Kalman. *A New Approach to Linear Filtering and Prediction Problems*. Journal of basic Engineering, vol. 35, Mars 1960.

[Leonard & Durrant-Whyte 1991] John Leonard and Hugh Durrant-Whyte. *Simultaneous map building and localization for an autonomous mobile robot*. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1991.

[Lu & Milios 1997a] F. Lu and E. Milios. *Globally Consistent Range Scan Alignment for Environment Mapping*. Autonomous Robots, 1997.

[Lu & Milios 1997b] Feng Lu and Evangelos Milios. *Robot pose estimation in unknown environments by matching 2d range scans*. Journal of Intelligent and Robotic Systems, vol. 18, pages 249–275, 1997.

[Montemerlo *et al.* 2002] Michael Montemerlo, Sebastian Thrun, Daphne Koller and Ben Wegbreit. *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem*. In In Proceedings of the AAAI National Conference on Artificial Intelligence, pages 593–598. AAAI, 2002.

[Montemerlo *et al.* 2008] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger,

G. Stanek, D. Stavens, A. Vogt and S. Thrun. *Junior: The Stanford Entry in the Urban Challenge*. Journal of Field Robotics, 2008.

[Montesano *et al.* 2005] L. Montesano, J. Minguez and L. Montano. *Modeling the Static and the Dynamic Parts of the Environment to Improve Sensor-based Navigation*. In Proc. IEEE International Conference on Robotics and Automation (ICRA), pages 4556–4562, 18–22 April 2005.

[Murphy 1999] Kevin P. Murphy. *Bayesian Map Learning in Dynamic Environments*. Neural Information Processing Systems (NIPS), pages 1015–1021, 1999.

[Oh *et al.* 2004] Songhwai Oh, Stuart Russell and Shankar Sastry. *Markov Chain Monte Carlo Data Association for General Multiple-Target Tracking Problems*. In IEEE Conference on Decision and Control, 2004.

[Olson 2000] Clark F. Olson. *Probabilistic self-localization for mobile robots*. IEEE Transactions on Robotics and Automation, vol. 16, pages 55–66, 2000.

[Oriolo *et al.* 1997] G. Oriolo, G. Ulivi and M. Vendittelli. *Fuzzy maps: a new tool for mobile robot perception and planning*. J. of Robotic Systems, vol. 14, no. 3, pages 179–197, 1997.

[Pagac *et al.* 1998] D. Pagac, E.M. Nebot and H. Durrant-Whyte. *An evidential approach to map-building for autonomous vehicles*. IEEE Transactions on Robotics and Automation, vol. 14, 1998.

[Petrovskaya & Thrun 2008] Anya Petrovskaya and Sebastian Thrun. *Model Based Vehicle Tracking for Autonomous Driving in Urban Environments*. In Proceedings of Robotics: Science and Systems IV (RSS), Zurich, Switzerland, June 2008.

[Pfister *et al.* 2003] Samuel T. Pfister, Stergios I. Roumeliotis and Joel W. Burdick. *Weighted Line Fitting Algorithms for Mobile Robot Map Building and Efficient Data Representation*. In In ICRA, pages 14–19, 2003.

[Prassler *et al.* 1999] E. Prassler, J. Scholz and P. Fiorini. *Navigating a Robotic Wheelchair in a Railway Station during Rush Hour*. International Journal on Robotics Research, vol. 18, no. 7, pages 760–772, 1999.

[Ramos *et al.* 2007] F. Ramos, D. Fox and H. Durrant-Whyte. *CRF-Matching: Conditional Random Fields for Feature-Based Scan Matching*. In Proc. of Robotics: Science & Systems (RSS), 2007.

[Reid 1979] Reid. *An Algorithm for Tracking Multiple Targets*. IEEE Transactions on Automatic Control, vol. 24, 1979.

[Schulz *et al.* 2001] D. Schulz, W. Burgard, D. Fox and A.B. Cremers. *Tracking Multiple Moving Targets with a Mobile Robot using Particle Filters and Statistical Data Association*. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2001.

[Shafer 1976] G. Shafer. *A meathematical theory of evidence*. Princeton University Press, 1976.

[Song & Nevatia 2005] X. F. Song and R. Nevatia. *A Model-Based Vehicle Segmentation Method for Tracking*. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2005.

[Thrun *et al.* 2000] 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), 2000.

[Thrun *et al.* 2005] S. Thrun, W. Burgard and D. Fox. Probabilistic robotics (intelligent robotics and autonomous agents). The MIT Press, September 2005.

[Thrun 2002] Sebastian Thrun. *Robotic Mapping: A Survey*. Exploring Artificial Intelligence in the New Millenium, 2002.

[Tierney 1996] Luke Tierney. *Markov chain concepts related to sampling algorithms*. Markov Chain Monte Carlo in Practice, pages 59–74, 1996.

[Tugnait 1982] J. K. Tugnait. *Detection and estimation for abruptly changing systems*. Automatica, vol. 18, pages 607–615, 1982.

[Urmson *et al.* 2008] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.W Seo, S. Singh, J. Snider, A. Stentz, W. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J.V. Nickolaou Sadekar, w. Zhang, J. Struble, M. Taylor, M. Darms and D. Ferguson. *Autonomous driving in urban environments: Boss and the Urban Challenge*. Journal of Field Robotics, vol. 25(8), pages 425–466, 2008.

[Viola & Jones 2001] Paul Viola and Michael Jones. *Robust Real-time Object Detection*. International Journal of Computer Vision, 2001.

[Vu & Aycard 2009] Trung-Dung Vu and Olivier Aycard. *Lased-based Detection and Tracking Moving Object using Data-Driven Markov Chain Monte Carlo*. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan, May 2009.

[Vu *et al.* 2007] Trung-Dung Vu, Olivier Aycard and Nils Appenrodt. *Online Localization and Mapping with Moving Object Tracking in Dynamic Outdoor Environment*. In Proceedings of the IEEE Intelligent Vehicle Symposium, Istanbul, Turkey, June 2007.

[Wang & Thorpe 2002] 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.

[Wang *et al.* 2003] C.-C. Wang, C. Thorpe and S. Thrun. *Online Simultaneous Localization and Mapping with Detection and Tracking of Moving Objects: Theory and Results from a Ground Vehicle in Crowded Urban Areas*. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), volume 1, 2003.

[Wang *et al.* 2004] C.-C. Wang, D. Duggins, J. Gowdy, J. Kozar, R. MacLachlan, C. Mertz, A.Suppe and C. Thorpe. *Navlab SLAMMOT Datasets*. http://www.cs.cmu.edu/~bobwang/datasets.html, May 2004. CMU.

[Wang 2004] Chieh-Chih Wang. *Simultaneous Localization, Mapping and Moving Object Tracking*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 2004.

[Wolf & Sukhatme 2005] Denis F. Wolf and Gaurav S. Sukhatme. *Mobile Robot Simultaneous Localization and Mapping in Dynamic Environments*. Autonomous Robots, vol. 19, 2005.

[Yu *et al.* 2006] Q. Yu, I. Cohen, G. Medioni and B. Wu. *Boosted Markov Chain Monte Carlo Data Association for Multiple Target Detection and Tracking*. In ICPR, pages II: 675–678, 2006.

[Zhao & Thorpe 1998] Liang Zhao and Chuck Thorpe. *Qualitative and Quantitative Car Tracking from a Range Image Sequence*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 496–501. IEEE, June 1998.

[Zhao *et al.* 2008] Tao Zhao, Ramakant Nevatia and Bo Wu. *Segmentation and Tracking of Multiple Humans in Crowded Environments*. PAMI, vol. 30, no. 7, pages 1198–1211, 2008.

[Zhu *et al.* 2000] S.C. Zhu, R. Zhang and Z. Tu. *Integrating Top-down/Bottom-up for Object Recognition by Data Driven Markov Chain Monte Carlo*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2000.

List of Publications

Journal Articles

1. Sylvia Pietzsch, Trung-Dung Vu, Julien Burlet, Olivier Aycard, Thomas Hackbarth, Nils Appenrodt, Jurgen Dickmann, and Bernd Radig. *Results of a PreCrash Application based on Laserscanner and Short Range Radars*. Accepted for publication in the IEEE Transaction on Intelligent Transport System.

2. Trung-Dung Vu, Julien Burlet and Olivier Aycard. *Grid-based Localization and Local Mapping with Moving Object Detection and Tracking*. Accepted for publication in the special issue on Intelligent Transportation Systems of the Elsevier Journal Information Fusion.

Conference Articles

1. Trung-Dung Vu and Olivier Aycard. *Laser-based Detection and Tracking Moving Objects using Data-Driven Markov Chain Monte Carlo*. In Proceedings of the IEEE International Conference on Robotics and Automation, 2009.

2. Trung-Dung Vu, Julien Burlet and Olivier Aycard. *Grid-based Localization and Online Mapping with Moving Object Detection and Tracking: New Results*. In Proceedings of the IEEE Intelligent Vehicle Symposium, 2008.

3. Sylvia Pietzsch, Olivier Aycard, Julien Burlet, Trung-Dung Vu, Thomas Hackbarth, Nils Appenrodt, J. Dickmann, and B. Radig. *Results of a PreCrash Application based on Laserscanner and Short Range Radars*. In Proceedings of the IEEE Intelligent Vehicle Symposium, 2008.

4. Trung-Dung Vu, Olivier Aycard and Nils Appenrodt. *Localization and Mapping with Moving Object Tracking in Dynamic Outdoor Environments*. In Proceedings of the IEEE Intelligent Vehicle Symposium, 2007.

5. Olivier Aycard, Anne Spalanzani, Julien Burlet, Chiara Fulgenzi, Trung-Dung Vu, David Raulo and Manuel Yguel. *Pedestrian Tracking using off-board Camera*. In Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems, 2006.

6. Olivier Aycard, Anne Spalanzani, Julien Burlet, Chiara Fulgenzi, Trung-Dung Vu, David Raulo and Manuel Yguel. *Grid-based Fusion and Tracking*. In Proceedings of the IEEE Int. Conf. on Intelligent Transportation Systems, 2006.