Contribution to Perception for Intelligent Vehicles

Olivier Aycard

UNIVERSITE DE GRENOBLE

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

HABILITATION À DIRIGER LES RECHERCHES

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

présentée et soutenue publiquement par

Olivier AYCARD

le 9 Décembre 2010

Titre :

Contribution to Perception for Intelligent Vehicles

Composition du jury :

M. Jim Crowley	Professor at Grenoble University (INPG)	President
M. Didier Aubert	Senior Researcher at LIVIC	Reviewer
M. Michel Devy	Senior Researcher at LAAS	Reviewer
M. Christoph Stiller	Professor at Karlsruhe University	Reviewer
M. Eric Gaussier	Professor at Grenoble University (UJF)	Examiner
M. Thierry Fraichard	Senior Researcher at INRIA	Examiner

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 focus on using laser scanner as a main perception sensor in context of dynamic outdoor environments. To solve this problem, we have to deal with 3 main tasks : (1) identify static part and dynamic entities moving in the environment, (2) use static part of the environment to build a map of the environment and localize the vehicle inside this map : this task is know as "Simultaneous Localization And Mapping" (SLAM) and finally (3) Detect And Track Moving Objects (DATMO).

Regarding SLAM, the first contribution of this research is made by a grid-based approach [1] to solve both problems of SLAM and 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. Our second contribution is an efficient, precise and multiscale representation of 2D/3D environment. This representation is actually an extension of occupancy grid where (1) only cells corresponding to occupied part of the environment are stored and updated (2) where cells are represented by a cluster of gaussian to have a fine representation of the environment and (3) where several occupancy grids are used to store and update a multiscale representation of the environment.

Regarding DATMO, we firstly 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 explore the hypothesis space and effectively find the most likely solution. An other important problem to solve regarding DATMO is the definition of an appropriate dynamic model. In practice,

1. An occupancy grid is a decomposition of the environment in rectangular cells where each cell contains the probability that it is occupied by an obstacle.

objects can change their dynamic behaviors over time (e.g. : stopped, moving, accelerating, etc...). To adapt to these changing behaviors, a multiple dynamic model is generally required. But, this set of dynamic models and interactions between these models are always given *a priori*. Our second contribution on DATMO is a method to guide in the choice of motion models and in the estimation of interactions between these motion models.

The last part of this thesis reports integration of these contributions on different experimental platforms in the framework of some national and european projects. Evaluations are presented which confirm the robustness and reliability of our contributions.

Table of Contents

1 Introduction **1**

1.1 Context . . . 1

1.2 Problematic . . . 4

1.2.1 Simultaneous Localization And Mapping (SLAM) . . . 6

1.2.2 Detection And Tracking of Moving Objects (DATMO) . . . 7

1.3 Contributions and Outline of the document . . . 8

2 Simultaneous Localization and Mapping **11**

2.1 Introduction . . . 11

2.1.1 Bayesian Filters . . . 12

2.1.2 Map Representation . . . 15

2.1.3 Simultaneous Localization And Mapping (SLAM) . . . 17

2.1.4 Synthesis and Contributions on SLAM . . . 19

2.2 Grid-based SLAM in dynamic environments . . . 21

2.2.1 Introduction . . . 21

2.2.2 Grid Mapping . . . 22

2.2.3 Dealing with dynamic environments . . . 28

2.3 Multiscale Gaussian Maps . . . 29

2.3.1 Introduction . . . 29

2.3.2 Map Representation . . . 32

2.3.3 Map Updating . . . 34

2.3.4 Map Refinement . . . 37

2.4 Conclusion . . . 40

3 Detection And Tracking of Moving Objects **41**

3.1 Introduction . . . 41

3.1.1 Moving Object Detection . . . 42

3.1.2 Tracking of Moving Objects . . . 43
3.1.3 Filtering - Multiple Dynamic Model . . . 43
3.1.4 Synthesis and Contributions on DATMO . . . 49
3.2 DATMO Formulation . . . 50
3.2.1 Introduction . . . 50
3.2.2 DATMO Formulation . . . 50
3.2.3 DATMO Decomposition . . . 54
3.2.4 Efficient DATMO Computation using MCMC . . . 57
3.3 Adaptative Interactive Multiple Models . . . 63
3.3.1 Introduction . . . 63
3.3.2 Principle . . . 64
3.3.3 Continuous and online adaptation of TPM . . . 66
3.3.4 Use of adaptation to define the set of motion models for IMM . . . 67
3.3.5 Interacting Multiple Models based Classification of Moving Objects . . . 69
3.4 Conclusion . . . 73

4 Applications **75**
4.1 Introduction . . . 75
4.2 National project PUVAME . . . 75
4.2.1 Project description . . . 75
4.2.2 Experimental Platform . . . 76
4.2.3 Experimental Results . . . 76
4.3 European IP PReVENT-ProFusion . . . 79
4.3.1 Project description . . . 79
4.3.2 Demonstrators . . . 79
4.3.3 Experimental Results . . . 80
4.4 NAVLAB . . . 82
4.4.1 Project description . . . 82
4.4.2 CMU Demonstrator . . . 82
4.4.3 Experimental Results . . . 83
4.5 3D Mapping . . . 84
4.5.1 Project description and Demonstrator . . . 84
4.5.2 Experimental Results . . . 84
4.6 European Project Intersafe2 . . . 86

4.6.1 Project description . . . 86
4.6.2 Demonstrator . . . 86
4.6.3 Experimental Results . . . 86
4.7 Conclusion . . . 87

5 Conclusion and Perspectives **89**
5.1 Conclusion . . . 89
5.2 Perspectives . . . 92
5.2.1 Environment Representation and Interpretation of Environment . . . 92
5.2.2 Frontal object perception . . . 93
5.2.3 Sensor Data Fusion . . . 94
5.2.4 Trajectory estimation and prediction of moving objects . . . 95
5.2.5 Ambiant Intelligence and Perception . . . 96

A Curiculum Vitae **99**
A.1 Adress . . . 99
A.2 Education . . . 99
A.3 Professional Experience . . . 100
A.4 Students supervision . . . 100
A.4.1 PhD students . . . 100
A.4.2 Post-Doctoral Students . . . 101
A.4.3 Master Students . . . 101
A.5 Scientific Activities . . . 102
A.5.1 Program Committee . . . 102
A.5.2 Thesis Committee . . . 102
A.5.3 Keynote Speech . . . 103
A.5.4 Reviewer of european projects . . . 103
A.6 Teaching Activities . . . 103
A.6.1 Lectures . . . 103
A.6.2 Tutorials . . . 104
A.6.3 Collective responsabilities . . . 104
A.7 Projects . . . 105
A.8 Miscellaneous . . . 105

B Publications **107**

C Bibliography 115

Chapitre 1

Introduction

1.1 Context

FIGURE 1.1 – Examples of intelligent vehicles : on the left, a cycab : experimental vehicle of INRIA and on the right, the Daimler demonstrator of european project PReVENT

We generally define an intelligent vehicle (FIGURE 1.1) as a vehicle designed to help driving automatically or to monitor a human driver and assist him in driving. 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. To solve these tasks, they are equipped with sensors to perceive their surrouding environment and with actuators to act in this environment.

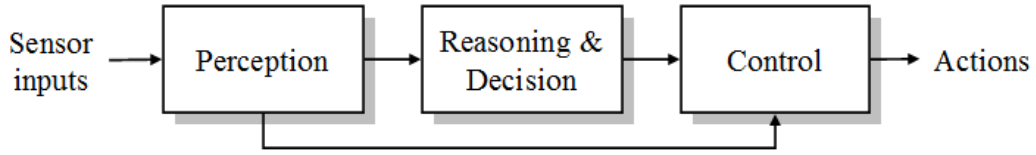


FIGURE 1.2 – Paradigm defining an intelligent vehicule

To work, an intelligent vehicule requires 3 fundamental components (FIGURE 1.2) : i) perceive and model the environnement where it is moving, ii) reason and decide about futur actions to execute iii) and finally perform these actions. Perception plays a fundamental role in construction of an intelligent vehicle as it constitutes its first component and provides informations to other components. Its objective is to interpret noisy and raw data of different sensors embedded on a vehicle to model environment and understand current situation in order to provide necessary informations to decide future actions to execute. The quality of perception processing has an impact on the quality of the whole process.

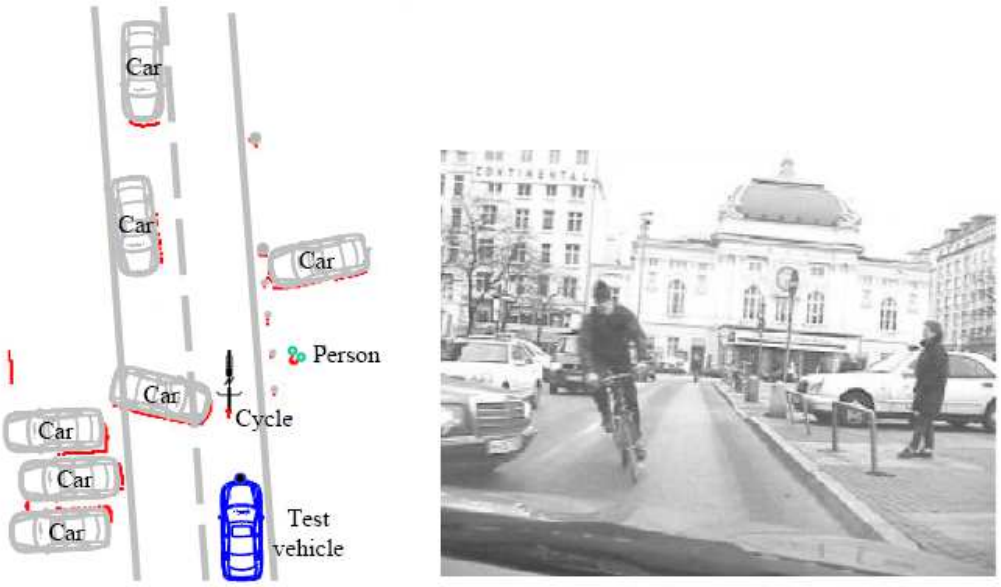


FIGURE 1.3 – Example of perception from a vehicule equipped with a laser scanner

FIGURE 1.3 illustrates problems we have to solve, to design a perception module in dynamic and outdoor environment. This figure shows data provided by a laser scanner embedded on a vehicule moving in an urban environment. On the right part, we see a picture to help us to understand the current situation. On the left, we have a top view of the scene. Data provided by laser scanner are represented by red dots. This data provides us with a 2D slice of the 3D environment. This slice is located at the height of the laser scanner. Each red dot corresponds to a hit of the laser. On the left in grey color, we also find the "real" scene with objects it contains. The first thing that we can note is the important difference between real scene and raw data provided by laser scanner. So, we have to interpret this raw data to understand the current situation. For instance, the environment is composed of static and dynamic objects. Usually, sensors do not provide this kind of information and this is while interpreting the sensor data that we are able to determine if an object is static or dynamic. Moreover, the process done with dynamic objects is different of the process done with static objects. With static objects, we build a map of the environment and localize the ego vehicle inside this map. This process is known as SLAM [1] and is described in section 1.2.1. Regarding dynamic entities, we have to identify them (we usually speak about detection) and track them (ie, sequentially estimate their position and some other quantities). This process is known as DATMO [2] and is described in section 1.2.2.

An other important problem is due to the fact that we are only able to see a part of objects present in the environment. For instance, this is the case for the car on the left foreground that is parking ; we only see its left part and a part of its bumper. Finally, perception that we have of objects (especially moving objects) will change during time as it is moving.

As illustrated on the previous figure, design and development of perception modules in outdoor and dynamic environment is still a scientific challenge. In our research activities, we are interested in perception in outdoor and dynamic environment. This document summarizes our contributions in this field : on one hand on the static part of the environnement and on the other hand on the dynamic part of the environment. In next section, we give an overview of the problematic of

1. SLAM stands for Simultaneous Localization And Mapping.
2. DATMO stands for Detection And Tracking of Moving Objects.

perception in dynamic and outdoor environments. Afterwards, we introduce our contributions and give the outline of the document.

1.2 Problematic

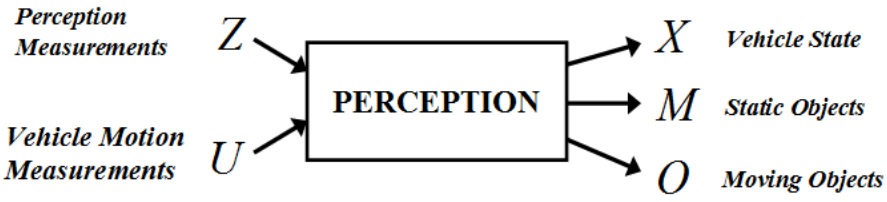


FIGURE 1.4 – The general perception process. Z denotes the perception measurements, U denotes the motion measurements, X is the vehicle state, M is the map of stationary objects and O denotes the states of moving objects.

The perception problem, as shown in FIGURE 1.4, 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 O in the vicinity of the vehicle. 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. The moving objects list O contains information about dynamic objects, their locations and dynamic states such as velocity and moving direction.

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 :

$$X = x_{0:t} = \{x_0, x_1, ..., x_t\} \tag{1.1}$$

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 :

$$Z = z_{0:t} = \{z_0, z_1, ..., z_t\} \tag{1.2}$$

$$U = u_{1:t} = \{u_1, u_2, ..., u_t\} \tag{1.3}$$

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

$$M = \{m^1, m^2, ..., m^K\} \tag{1.4}$$

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.

O denotes the moving objects lists up to time t :

$$O = O_{1:t} \tag{1.5}$$

The moving objects 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 :

$$O_t = \{o_t^1, o_t^2, ..., o_t^N\} \tag{1.6}$$

Our objective here is to estimate (1) the state of the vehicle X, the static map M and the state of moving objects O given sensor measurements Z and U over time. This problem is known as the SLAM with DATMO.

[WTT03] introduced a mathematical framework to solve SLAM with DATMO in dynamic environments in which SLAM is integrated with tracking generalized objects (both static and dynamic). The SLAM with DATMO problem can be represented by a joint posterior over states of all objects need to estimate (ego-vehicle pose, stationary objects, dynamic objects) given all sensor measurements :

$$P(X, M, O|Z, U) \tag{1.7}$$

He showed that estimating (1.7) 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 SLAM with DATMO estimation problem into two separate estimators :

$$\underbrace{P(X, M, O|Z, U)}_{\text{SLAMMOT}} = \underbrace{P(X, M|Z^{(s)}, U)}_{\text{SLAM}} \quad \underbrace{P(O|Z^{(d)})}_{\text{Moving objects Tracking}} \tag{1.8}$$

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

$$\underbrace{Z = Z^{(s)} + Z^{(d)}}_{\text{Moving objects Detection}} \tag{1.9}$$

where $Z^{(s)}$ and $Z^{(d)}$ denote measurements corresponding to static objects and dynamic objects respectively. The decomposition in (1.9) corresponds to the moving objects detection step in DATMO.

By such maintenance of separate posteriors for stationary objects and moving objects, the resulting estimation problem of SLAM with DATMO (1.8) are much lower dimensional than problem of direct estimating SLAM with DATMO in (1.7). In the two next sub-sections, we briefly introduce SLAM problem and DATMO problem.

1.2.1 Simultaneous Localization And Mapping (SLAM)

When a vehicule is moving, we need to know its relative position to its environment. 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 objects map in advance, the map-based localization techniques such as those proposed by [Ols00, FBDT99] and [DBFT99] can be used to increase the accuracy of the pose estimate. Unfortunately, it is difficult to build an usable stationary objects map a priori because of temporary stationary objects such as parked cars. Stationary objects 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 objects map. This problem is known as Simultaneous Localization And Mapping (SLAM). It allows intelligent vehicles to operate in an unknown environment and then incrementally build a map of this environment and concurrently use this map to localize themselves. To solve this problem, we

have to find a representation to map the environment and to design techniques to localize the vehicle inside the environment.

Over the last decade, the SLAM problem has attracted immense attention in the mobile robotics literature [Chr02], and SLAM techniques are at the core of many successful systems [Thr02]. However, [WT02] 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.

1.2.2 Detection And Tracking of Moving Objects (DATMO)

The moving objects tracking problem has been originated from radar tracking systems and extensively studied for several decades [BSF88, BP99]. Many tracking studies suppose that the measurements correspond uniquely to moving objects and focus on the multi-objects tracking 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 objects tracking system.

Moving objects detection in crowded urban environments is not easy because of a wide variety of targets. With cameras, detection of moving objects is usually based on a definition of some predefined classes of objects as a set of features [3] (for instance, pedestrians or vehicles) or based on a statistical model of their appearance in images [4][VJ01]. When laser scanner are used, motion-based approaches are usually the preferred solutions since both appearance-based and feature-based methods rely on prior knowledge of the objects. 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-objects tracking problem arises in order to estimate dynamic states of each object. In general, the problem of tracking multiple objects consists of two parts : *Filtering* and *Data Association* [BSF88]. 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

3. We speak about feature-based detection.
4. We speak about appearance-based detection.

observation corresponds to which object being tracked, then filtering techniques are applied to estimate object states with known observations. 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 difficult.

1.3 Contributions and Outline of the document

This document is composed of 5 chapters :

– Regarding the SLAM problem, we present our two contributions in chapter 2.

  Our first contribution [29, 5], in the framework of the PhD Thesis of Trung-Dung Vu, is made by a complete solution to solve both problems of SLAM and detection of moving objects which is based on occupancy grid[Elf89a] to represent the vehicle map. An occupancy grid is a discretization of the environment in rectangular cells (see section 2.1.2 for more details). In addition, a probabilistic measure of occupancy is estimated for each cell of the grid which indicates that cell is occupied by an object or not. 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. To correct vehicle location from odometry, we introduce a new and fast grid-based scan matching method which works 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. One important advantage of this approach is the fact that no model assumption is required to separate moving and stationary objects.

  For large 2D or 3D environments, it is trivial to say that a vehicle evolves in an environment where it has enough space to move, therefore most of this space is empty with respect to the vehicle volume. It implies that a lot of the time the algorithm used to update the grid with new observations is dedicated to update empty cells. Thus, it is not appropriate to use classical

occupancy grids for large maps and neither in 3D where most of the space is empty. An other problem, when using occupancy grids, is that a cell is considered as a full block, so all the information concerning the shape of the cell contents is lost. A way to alleviate this problem is to attach some sort of statistical shape description to every occupied cell. Finally, grids are neither unable to provide a multi-scale representation of the environment. Nevertheless, this kind of representation is usefull for some tasks in mobile robotics : for instance, coarse maps are used in path planning [YSSB98] or localization [RB05] algorithms in order to obtain a rough trajectory or position estimate at a low computational cost. Then, at a second stage, this estimate is used to initialize the fine scale search algorithms, thus accelerating convergence. Our second contribution [21, 4], in the framework of the PhD Thesis of Manuel Yguel, is a new map representation to overcome these limitations of classical occupancy grids : multi-scale gaussian maps. We firstly detail this representation composed of 3 scale map where each scale is a sparse grid and each cell is represented by one or more gaussian. We also define a method to update a multi-scale gaussian map with observations. Finally, we detail an error-driven refinement algorithm for coarser scales to improve the quality of the representation.

– Regarding the DATMO problem, our two contributions are detailled in chapter 3.

  In our first contribution [23, 5], in the framework of the PhD Thesis of Trung-Dung Vu, 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. 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 [ZZT00].

Regarding filtering, the definition of an appropriate dynamic model is one of the most important problem to solve. In practice, objects can change their dynamic behaviors over time (e.g. : stopped, moving, accelerating, etc...). To adapt to these changing behaviors, a multiple dynamic model is generally required. The definition of this set of dynamic models is crucial for tracking of moving objects. In our second contribution [33, 6], in the framework of the PhD Thesis of Julien Burlet, we present a method to help us design this set of motion models and also to automatically model interactions between the different motion models instead of defining them *a priori*. This method used a set of trajectories of past moving objects present in the environment to learn the interactions between motion models and secondly analyzing this learned set of motion models help us to design the final set of motion models. The main advantage of this method is to design multiple motions models that are very close to the real motions of tracked objects. Finally, we detail how we improve our adaptive method to track objects by adding a classification module [17]. For this, we consider the set of motion models as a set of behaviors and use classification techniques to identify typical behaviors.

– Our last contribution is an integration and validation of this work on different experimental platforms in the framework of some national and european projects. This contribution is reported in Chapter 4. For each project, we describe experimental platforms that have been used for integration and experimentation. Afterwards, we detail for each experimental platform the used contributions and how they have been integrated. Finally, evaluations are presented which confirm the robustness and reliability of our contributions.
– We conclude and give some perspectives to our researches in chapter 5.

Chapitre 2

Simultaneous Localization and Mapping

2.1 Introduction

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 :

$$P(x_t, M|z_{0:t}, u_{1:t}) \text{ [1]} \tag{2.1}$$

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

1. Instead of determining the whole trajectory of the ego vehicle X as in equation 1.8, in practise we are usually only interested in determining the current state of the ego vehicle x_t.

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 :

$$\underbrace{P(x_t, M|z_{0:t}, u_{1:t})}_{\text{posterior at } t} \propto \underbrace{P(z_t|x_t, M)}_{\text{update}} \underbrace{\int_{x_{t-1}} P(x_t|x_{t-1}, u_t) \underbrace{P(x_{t-1}, M|z_{0:t-1}, u_{1:t-1})}_{\text{posterior at } t-1}}_{\text{prediction}} \tag{2.2}$$

For this computation, we need to specify two probabilities : the sensor measurement model $P(z_t \,|\, x_t, M)$ and the vehicle motion model $P(x_t \,|\, x_{t-1}, u_t)$. 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.

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 (2.2) which involves finding an appropriate representation for the measurement model and motion model. Before describing state-of-the-art SLAM algorithms, 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.

2.1.1 Bayesian Filters

The task of estimating system states from sensor data is at the core of any intelligent vehicle. State estimation addresses the problem of estimating quantities from sensor data that are not directly observable, but that can be inferred. In most intelligent vehicles applications, determining what to do is relatively easy if one only knew certain quantities. For example, moving a vehicle is relatively easy if the exact location of the vehicle and the position of all nearby obstacles

are known. Unfortunately, these variables are not directly measurable. Instead, a vehicle 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) [AM79] 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.

The underlying task of the filter is to estimate the posterior probability $P(s_t|z_{0:t}, u_{1:t})$ where s_t represents the state of the system at time t, $z_{0:t}$ represents observations on the state of the system from time 0 to time t and $u_{1:t}$ represents actions of the system from time 1 to time t. An important property of the Bayesian filter is that this probability can be solved recursively using the Bayesian theorem :

$$\begin{aligned} \underbrace{P(s_t|z_{0:t}, u_{1:t})}_{\text{posterior at } t} &= \eta\ P(z_t|s_t)P(s_t|z_{0:t-1}, u_{1:t}) \\ &= \eta\ \underbrace{P(z_t|s_t)}_{\text{update}} \underbrace{\int_{s_{t-1}} P(s_t|s_{t-1}, u_t)\ \underbrace{P(s_{t-1}|z_{0:t-1}, u_{1:t-1})}_{\text{posterior at } t-1}}_{\text{prediction}} \end{aligned} \tag{2.3}$$

where η is the normalization constant. The recursive computation is initialized by the prior distribution $p(s_0|z_0) = p(s_0)$.

This algorithm can be interpreted as transformations over distributions of probability. Using the state transition function $P(s_t|s_{t-1}, u_t)$ and the previously estimated probability $P(s_{t-1}|z_{0:t-1}, u_{1:t-1})$, we obtain a distribution $P(s_t|z_{0:t-1}, u_{1:t})$ which is commonly called the prediction step. Then introducing the new measure we update the distribution with the likelihood $P(z_t|s_t)$ to obtain the desired result $P(s_t|z_{0:t}, u_{1:t})$. The process is illustrated in FIGURE 2.1. We note that the system used for the Bayesian filtering example presented 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.

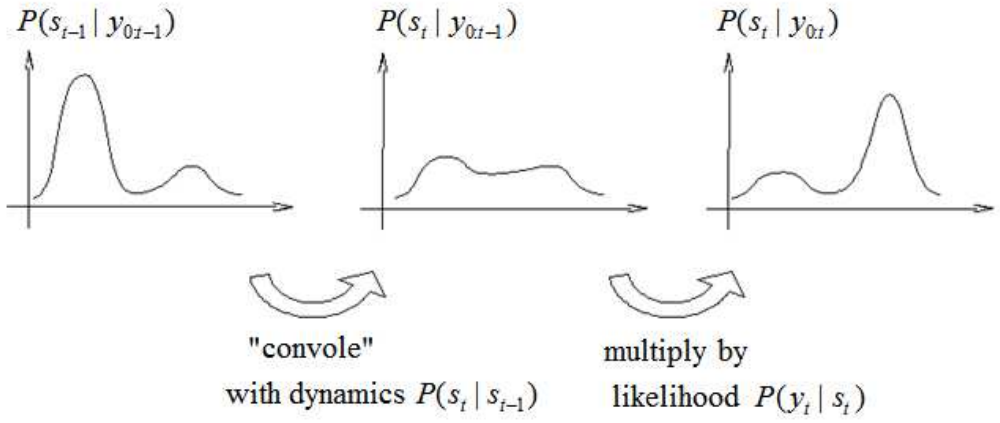


FIGURE 2.1 – Sequential Bayesian Filtering.

It is clear that in order to compute the sequential Bayesian filter, definitions of the likelihood function (or measurement model) $P(z_t|s_t)$ and the state transition function (dynamics model) $P(s_t|s_{t-1}, u_t)$ are required. And for any implementation also the description of the probability density functions have to be 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 [Kal60]. 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) [AM79] 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.

To handle highly nonlinear and non-Gaussian models in Bayesian filtering, Particle filters [AMGC02] 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.

2.1.2 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 [LDW91], grid-based approach [Elf89a], direct approach [LM97a].

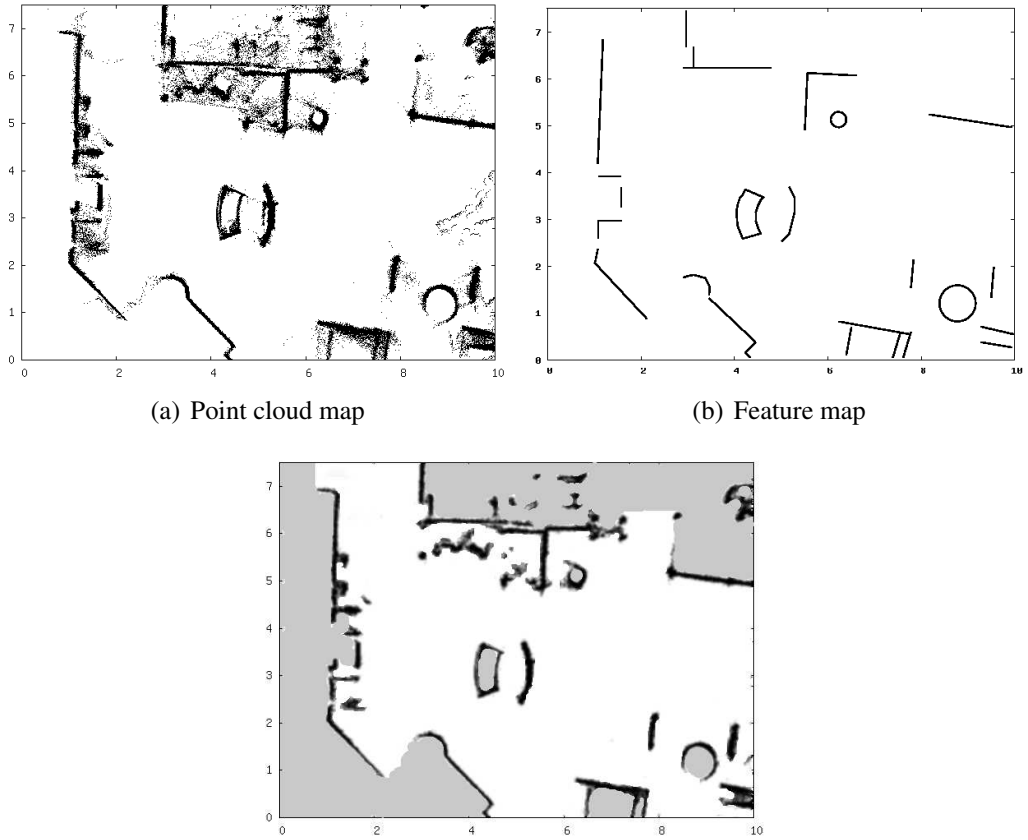
Feature-based Representation

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 [EF97] for detecting line segments, the Hough-transform [PRB03] or RANSAC [FP02] methods for detecting lines or circles. An example of a two-dimensional geometric feature map is presented in the image FIGURE 2.2(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 [Elf89a]. In this representation, the environment is subdivided into a regular array or a grid of rectangular cells. The resolution of the environment representation directly



(a) Point cloud map

(b) Feature map

(c) Grid map

FIGURE 2.2 – Example of different representation methods of the map built from the same set of laser data measurements.

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.

An example of an occupancy grid map representation is shown in FIGURE 2.2(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.

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 2.2(a)).

2.1.3 Simultaneous Localization And Mapping (SLAM)

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 (2.2). 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 [Tho].

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. 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.

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.

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 :

$$\hat{x}_t = \underset{x_t}{\operatorname{argmax}}\, p(x_t|z_t, u_t, \hat{x}_{t-1}, \hat{M}_{t-1}) \quad = \underset{x_t}{\operatorname{argmax}}\{P(z_t|x_t, \hat{M}_{t-1})\, P(x_t|u_t, \hat{x}_{t-1})\} \tag{2.4}$$

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 :

$$\hat{M}_t = \hat{M}_{t-1} \cup \{\langle \hat{x}_t, z_t \rangle\} \tag{2.5}$$

Maximizing (2.4) 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 2.1.2), we have corresponding scan matching methods as seen in the literature : direct [LM97b], feature-based [RFDW07] or grid-based [TBF00].

FastSLAM

An alternative SLAM approach is FastSLAM [MTKW02] 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 (2.1), 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 [Mur99], 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 :

$$P(x_{0:t}, M \mid z_{0:t}, u_{1:t}) = \underbrace{P(x_{0:t} \mid z_{0:t}, u_{1:t})}_{\text{estimate of trajectory}} \underbrace{P(M \mid x_{0:t}, z_{0:t})}_{\text{map with known trajectory}} \tag{2.6}$$

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 (2.6) efficiently.

2.1.4 Synthesis and Contributions on SLAM

Synthesis of SLAM techniques

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. Because of advantages over others, nowadays, occupancy grid have become the most common choice among

map representation methods, particular for applications in outdoor environments [WTT03, 29, MBB+08].

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, KF 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.

Contributions on SLAM

Our first contribution [29, 5], in the framework of the PhD Thesis of Trung-Dung Vu, is an algorithm to solve SLAM with moving object detection in outdoor and dynamic environment 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 [Wan04]. It is based on Occupancy Grid to represent the environment and ML SLAM. This contribution is detailled in the next section.

Our second contribution [21, 4], in the framework of the PhD Thesis of Manuel Yguel, is an efficient, precise and multiscale representation. This representation is actually an extension of occupancy grid where (1) only cells corresponding to occupied part of the environment are stored and updated (2) where cells are represented by a cluster of gaussian to have a fine representation of the

environment and (3) where several occupancy grids are used to store and update a multiscale representation of the environment. This contribution is detailled in section 2.3.

2.2 Grid-based SLAM in dynamic environments

2.2.1 Introduction

In this section, 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 [Wan04].

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 2.1.3) 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 [Wan04] 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.

In the next section, we describe the different components of the grid-based mapping process : incremental grid mapping with known trajectories and Grid-based Scan Matching. Algorithm for detecting moving objects is presented in section 2.2.3.

2.2.2 Grid Mapping

Incremental Grid Mapping with Known Trajectories

In the occupancy grid representation, the 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 grid indicates the cell is occupied and a low value means the cell is free. Suppose that occupancy states of individual grid cells are independent, the objective of a mapping algorithm is to estimate the posterior probability of occupancy $P(M^i \,|\, x_{1:t}, z_{1:t})$ for each cell of grid M^i, given observations $z_{1:t} = \{z_1, ..., z_t\}$ at corresponding known poses $x_{1:t} = \{x_1, ..., x_t\}$.

In the literature, many methods are used for occupancy grid mapping, such as Bayesian [Elf89a], Dempster-Shafer [PNDW98] and Fuzzy Logic [OUV97]. Here we apply Bayesian Update scheme [TBF05] that provides an elegant recursive formula to update the posterior under log-odds form :

$$\begin{aligned} log\, O(M^i \,|\, x_{1:t}, z_{1:t}) &= log\, O(M^i \,|\, x_{1:t-1}, z_{1:t-1}) + \\ &\quad + log\, O(M^i \,|\, z_t, x_t) \\ \text{where } O(a \,|\, b) &= odds(a \,|\, b) = P(a \,|\, b) \,/\, (1 - P(a \,|\, b)) \end{aligned} \tag{2.7}$$

It is easy to see that the desired probability of occupancy, $P(M^i \,|\, x_{1:t}, z_{1:t})$, can be recovered from the log-odds representation. Moreover, since the updating algorithm is recursive, it allows for incremental map updating when new sensor data arrives.

In (2.7), the probability $P(M^i \,|\, x_t, z_t)$, is called the *inverse sensor model*. It specifies the probability that a grid cell M^i is occupied based on a single sensor measurement z_t at location x_t. 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.

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.

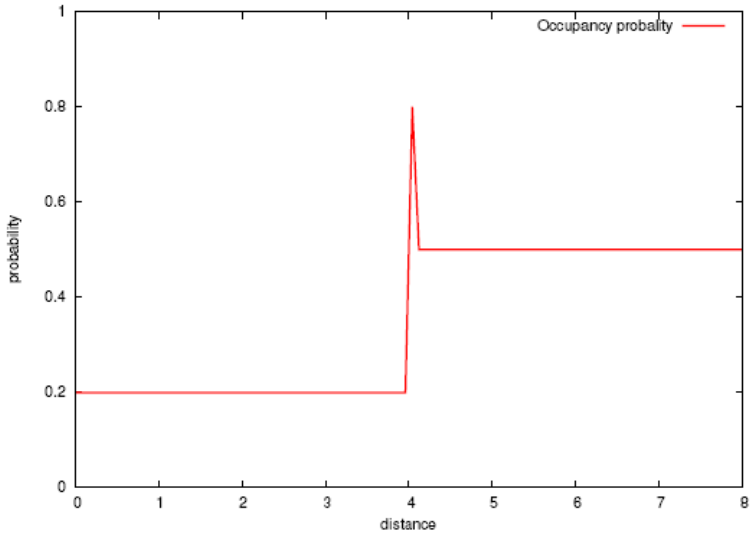


FIGURE 2.3 – This figure shows the occupancy probability of cells along a beam measuring a distance of 4m.

Figure 2.2.2 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). 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. Note that because of the small error of the sensor this function has only a narrow peak. So, 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, should be decreased.

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 2.4 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 2.1. Here we opt for the incremental maximum likelihood SLAM method due to the advantage of its computational and memory complexity over other methods (subsection 2.1.3).

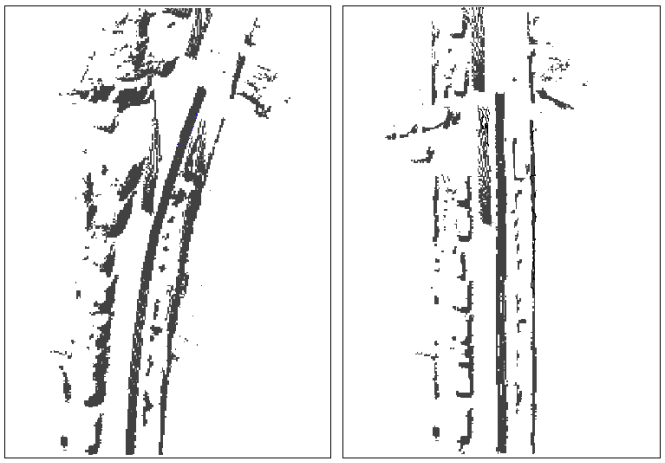


FIGURE 2.4 – Maps built directly from raw laser data collected from a vehicle moving along a straight street : with vehicle localization using odometry (left) ; and using results of scan matching (right). Note that the scan matching results are not affected by moving objects in the street. See FIGURE 2.10 for the resulting occupancy grid map.

Here we introduce an alternative grid-based scan matching method to solve (2.4). 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 (2.4).

For the motion model, we adopt a probabilistic velocity motion model similar to that described in [TBF05]. The vehicle motion u_t is comprised of two components, the translational velocity v_t and the yaw rate ω_t. FIGURE 2.5 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.

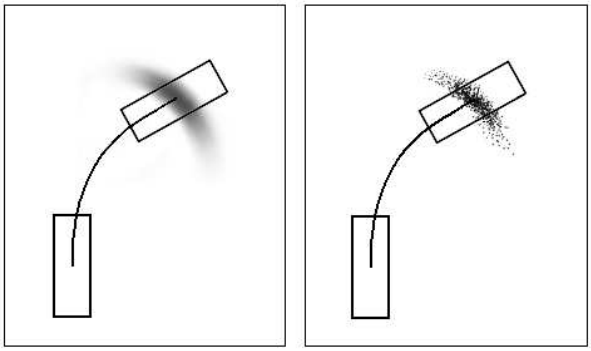


FIGURE 2.5 – The probabilistic velocity motion model $P(x_t \mid x_{t-1}, u_t)$ of the vehicle (left) and its sampling version (right).

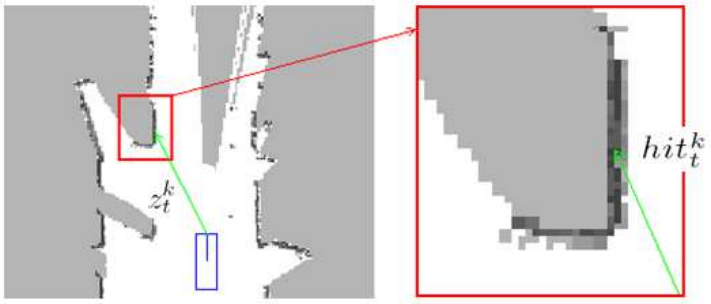


FIGURE 2.6 – The measurement model $P(z_t \mid x_t, M_{t-1})$.

For the measurement model $P(z_t \mid x_t, M_{t-1})$, mixture beam-based model is widely used in the literature [FBT99, HSB03]. 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 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, we can write the measurement model under the sum following :

$$P(z_t \mid x_t, M_{t-1}) \propto \sum_{n=1}^{N} \{\, P(M_{t-1}^{hit_t^n}) \text{ such that } M_{t-1}^{hit_t^n} \text{ is occupied} \,\} \tag{2.8}$$

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 (2.4) to find the correct pose $\hat{x}_t$. Hill climbing strategy in [TBF00, HSB03] 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 2.5 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 2.7. 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 2.7 – 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.

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). 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

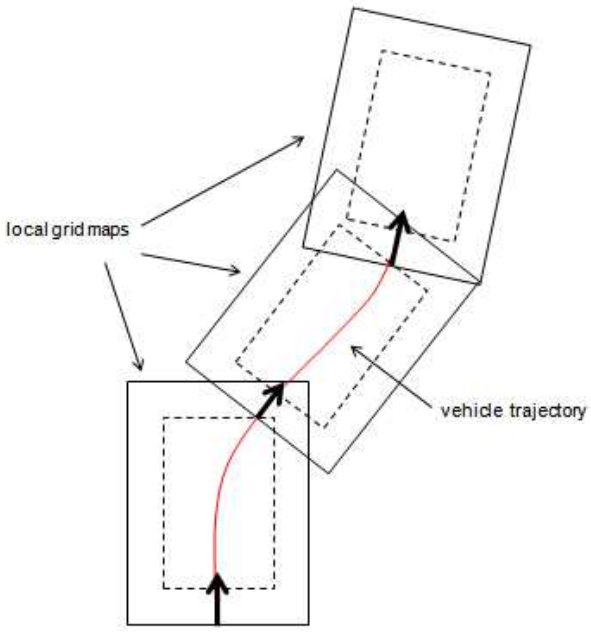


FIGURE 2.8 – A local map is maintained at a time. When the vehicle arrives at the boundary of the grid map, a new grid is created.

intersection area are copied from the old map (see FIGURE 2.8). A global map is constructed by concatenating local maps sequentially.

2.2.3 Dealing with dynamic environments

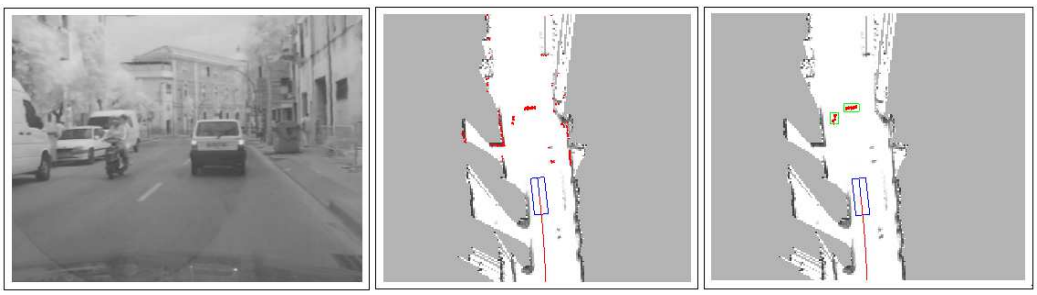


FIGURE 2.9 – Moving object detection example. See text for more details.

Besides the computational effectiveness, one attraction of our algorithm is that it is not affected by dynamic entities in the environment (see FIGURE 2.4 right).

Since we only consider occupied cells, spurious regions in the occupancy grid map that might belong to dynamic objects do not contribute to the sum (2.8). 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.

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 2.9 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 2.10 shows two occupancy grid maps constructed from the same laser data in FIGURE 2.4 with and without filtering out dynamic measurements. We can see that the left one built without the filtering step results in many fuzzy regions.

2.3 Multiscale Gaussian Maps

2.3.1 Introduction

To update an occupation grid with sensor data, one should find all the cells that need to be updated. In practice, this is the task that is the most computationally expensive. The parts of the grid that must be updated are the empty part and the

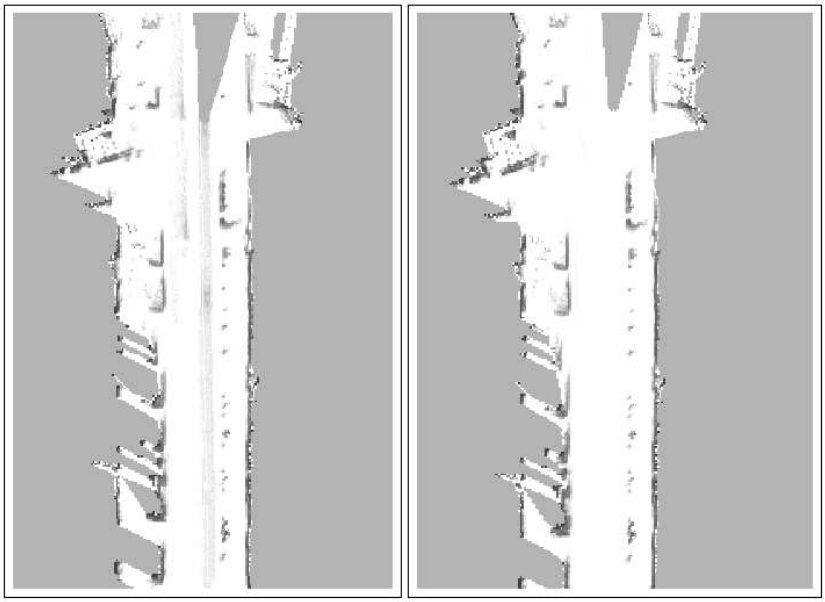


FIGURE 2.10 – Occupancy grid maps built with and without filtering out detected moving objects.

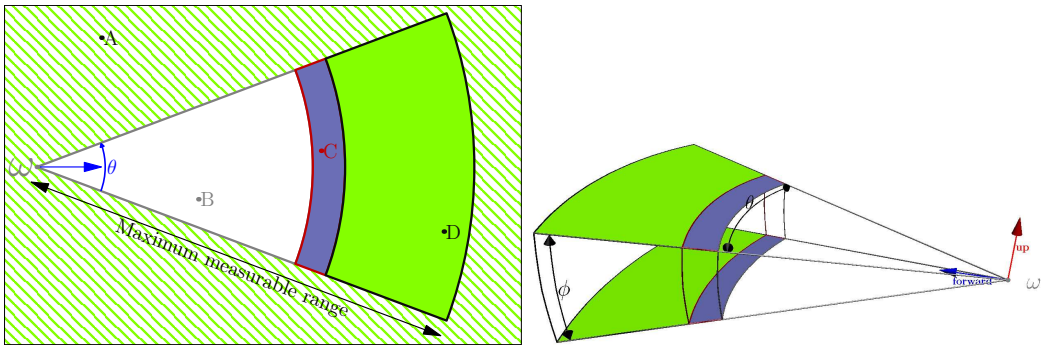


FIGURE 2.11 – 2D and 3D definitions of the different space regions according to their visibility. Left : for a 2D cone, the point $\boldsymbol{\omega}$ is the sensor position, it is empty otherwise the sensor could not have been put there. **A** is outside the sensor field of view. This place is hidden and treated as hidden in an occupancy update algorithm. **B** is in the empty area. **C** is in the occupied area. **D** is in the hidden area inside the sensor field of view. The blue arrow is the sensor direction (or forward vector) and with the point ω it defines the sensor position. The angular field of view, θ, and the maximum measurable range define the 2D cone geometry. Right : for a 3D cone. The red arrow is the up vector, the blue one is the forward vector and ω defines the sensor position. The angles θ and ϕ (horizontal and vertical, respectively) define the angular field of view ; with the maximum measurable range, they define the 3D cone geometry.

occupied part. In particular, as a real sensor has a limited field of view, it is not necessary to consider most of the grid. In 2D, the field of view is usually defined by a single angle, the angular field of view (θ in fig. 2.11(a)) and the maximum distance that the range-finder can detect : *the maximum measurable range*. In 3D it is defined by two angles : the angular horizontal field of view and the angular vertical field of view and the maximum measurable range (see fig. 2.11(b)). The shape of the field of view is called a cone measurement, see fig. 2.11.

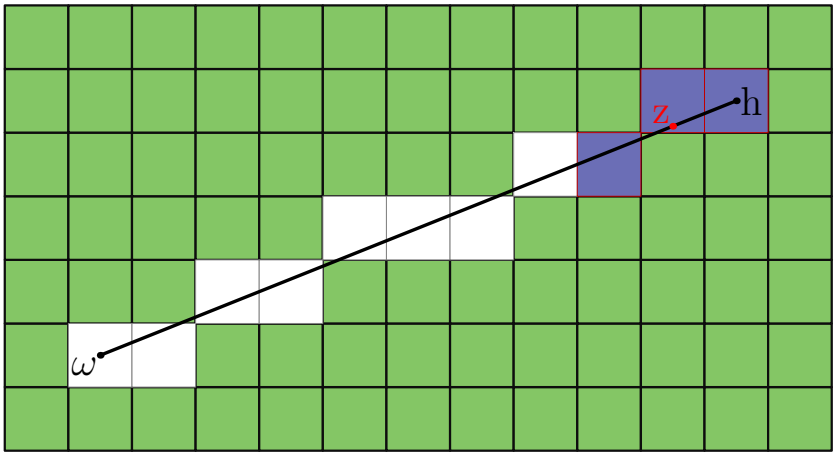


FIGURE 2.12 – Ray tracing techniques to update an occupancy grid.

With a laser range-finder the cone measurement is very narrow. Therefore, it is acceptable to make the approximation that it is a segment. This segment links the sensor origin $\boldsymbol{\omega}$ and the point on the line of sight at the beginning of the hidden area $\mathbf{h}$. Based on that assumption, the common algorithm in 2D traverses the segment $[\boldsymbol{\omega}, \mathbf{h}]$ using the well known, Bresenham algorithm ([FVFH90], see FIGURE 2.12). This algorithm is well suited for small, dense occupancy grids : for instance, it is very well suited for updating the local occupancy grid built in section 2.2. However, for a wide map, it is trivial to say that a vehicle evolves in an environment where it has enough space to move, therefore most of this space is empty with respect to the vehicle volume. It implies that a lot of the time the Bresenham algorithm is dedicated to update empty cells. Thus, it is not appropriate to use classical occupancy grids for large maps and neither in 3D where most of the space is empty.

An other problem, when using occupancy grids, is that a cell is considered

as a full block, so all the information concerning the shape of the cell contents is lost. A way to alleviate this problem is to attach some sort of statistical shape description to every occupied cell. Two seminal works in this direction are gaussian maps [MLT00] and the Normal Distribution Approach (NDT) [BS03], these approaches significantly improve accuracy by approximating the shape of the cell contents with gaussian clusters. The accuracy of these approaches, together with their relative simplicity has contributed to making them very popular in the map building community [GSB05, RB05]. That being said, a single gaussian is still a poor representation when there are several objects with different orientations in the cell.

Finally, grids are neither unable to provide a multi-scale representation of the environment. Nevertheless, this kind of representation is usefull for some tasks in mobile robotics : for instance, coarse maps are used in path planning [YSSB98, PR98] or localization [RB05] algorithms in order to obtain a rough trajectory or position estimate at a low computational cost. Then, at a second stage, this estimate is used to initialize the fine scale search algorithms, thus accelerating convergence.

In this section, we present a new map representation to overcome these limitations of classical occupancy grids : multi-scale gaussian maps. In the next subsection, we present this representation. In section 2.3.3, we explain how to update a multi-scale gaussian map. This procedure adapts existing clusters in order to minimize the representation error. It is similar to the standard update of conventional Gaussian maps, except that it takes into account the fact that a cell may contain several Gaussians. Section 2.3.4 presents our error-driven refinement algorithm for coarse scales to improve the quality of the representation at coarse level. This step is our main contribution, at every refinement step new Gaussian clusters are added to the cells where the representation error is maximum.

2.3.2 Map Representation

We suppose that a localization is provided and we construct a 3 scale map processing every scale independently :

- At the fine scale, cells have a side of 0.2m ;
- At the intermediate level, cells have a side of 1m ;
- At the coarse level, cells have a side of 5m.

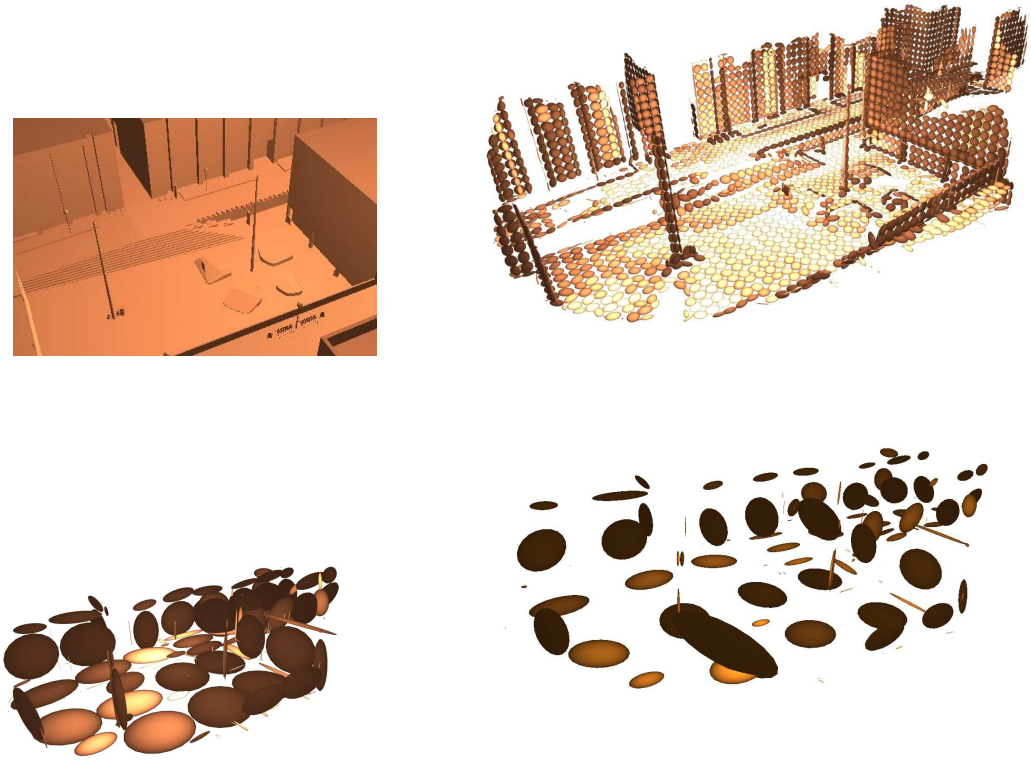


FIGURE 2.13 – Top left : 3D simulated scene. Top right : fine scale of the refined map. Bottom left : intermediate scale of the refined map. Bottom right : coarse scale of the refined map

Each scale is a sparse grid where only occupied cells are stored. A sparse grid is encoded using a special hash table : each occupied cell is indexed by a key built upon its integer coordinates inside the grid. Basically, the grid is seen as a 3-D array, and like for a matrix, an unique integer can be associated to a cell (i, j, k) :

$$\text{key} = k + w * j + h * w * i$$

where w and h are the number of cells in the width and the height of the grid. This supposes that the grid is contained inside a bounding box with dimensions (hs, ws, ds) where d is the number of cells in the depth of the grid and s is the cell size in meters. Thus for each real point, one compute the 3-D coordinates of its cell, then the key of the cell in the hash table. A cell is represented by one or

more gaussian : in the fine scale there is only one gaussian per cell and in coarser scales, a cell can contain several gaussians. FIGURE 2.13 gives an illustration of our map representation for a 3D simulated scene.

2.3.3 Map Updating

This section presents the procedure to update an existing map from sensor data. At this point, we assume that the number k of Gaussian clusters per cell is known. The actual estimation of k is handled by the refinement algorithm that we will discuss in section 2.3.4.

Our goal here is to update the Gaussians' mean value μ and covariance Σ in order to minimize the representation error with respect to the input data. Every observation is used to incrementally update the different scales independently. The basic idea is to find the cell where the input point falls and then updating the cluster in that cell that is "closest" to the input point.

As in most incremental approaches, an important question is how much to adapt the clusters – *i.e.* finding the 'right' learning rate. In the following subsection, we describe the use of the cluster's occupancy to control the adaptation. It can be intuitively explained as follows : the more a cluster has been observed, the more is known about it and the less reasonable it is to modify it. So, this adaptation depends on the occupancy of a cluster. In next subsection, we describe how this occupancy is computed for a point and a cluster.

Computing Cluster Occupancy

Basically, we project 2D (resp. 3D) grid modelling the environment on 1D (resp. 2D) grid. The idea is to have for each cell, the range of the closest point. Moreover, performing this "back projection" we avoid the computational burden associated to the Bresenham algorithm. The Bresenham algorithm need to access sequentially all the cell of the ray. However this access is very costly, since it is a search in the hash map even if the cell does not exist since it is empty. Furthermore the Bresenham algorithm accesses several times the cells that are close to the sensor since several rays traverse them. On the contrary, the back projection algorithm tests, in parallel, each occupied cell at most one time, and

only the occupied cells.

Afterwards, we use this "back projection" to compute occupancy. The occupancy of a cell, C, in the map is given by comparing the range measured in the pixel of its projection, I, in the range image with its actual range, δ. For a Gaussian, the occupancy is obtained by averaging the occupancy of n points sampled from the Gaussian distribution (rejecting those that fall outside the cell).

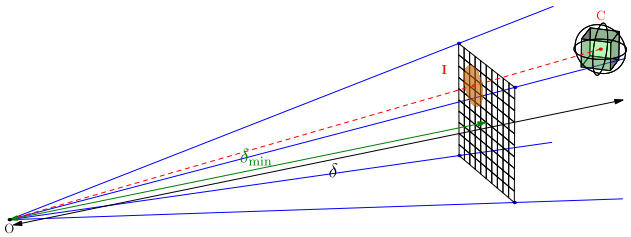


FIGURE 2.14 – Computation of occupancy in the range image of the bounding sphere of a cell.

We only need to guarantee that there are enough samples to provide a good estimate. To define n, we compute an upper bound of the number of points in the range image that can be contained in the cell. This is done using the projected bounding sphere of the cell. Let $\delta_{\min}$ and δ be the distance to the image plane in the camera coordinate system and the distance to the center of c (fig. 2.14), respectively. Then the projection of the bounding sphere of c occupies an area of $\frac{3\pi}{4}\left(\frac{\delta_{\min}}{\delta}a\right)^2$ (orange disc in Fig. 2.14), where a is the length of the side of c. Knowing the area of one pixel of the range image p, an upper bound for the number of pixels that may be projected back into the original cluster is :

$$B \triangleq \lceil \frac{3\pi}{4p}\left(\frac{\delta_{\min}}{\delta}a\right)^2 \rceil \tag{2.9}$$

which is the number of samples we are looking for. So, making $n = B$ gives us a good chance to cover every range image cell that effectively contains an observation from the cluster.

Updating Gaussian Clusters from Data

For every observation, a single cluster per scale will be updated. The cluster is selected by finding the cell that contains the observation and then finding the cluster having the minimum distance to that point.

Once the cluster has been selected, its parameters are updated by means of a stochastic gradient descent algorithm. The principle is to update the cluster by a fraction of the negative gradient with respect to each observation. As more and more samples are processed, the magnitude of the adaptation should decrease to ensure convergence. A good example is the on-line computation of the sample mean :

$$\boldsymbol{\mu}^n = \boldsymbol{\mu}^{n-1} + \tfrac{1}{n}(\mathbf{z}^n - \boldsymbol{\mu}^{n-1})$$

where n represents the number of samples processed so far, and $\mathbf{z}^n - \boldsymbol{\mu}^{n-1}$ can be understood as the negative gradient, and $\frac{1}{n}$ the fraction of the gradient to be taken into account. This decreasing weight is called the learning rate and is noted ϵ. In our approach, the value of ϵ depends on the occupancy, as described in section 2.3.3.

In the case of points, a distance metric between a point and a Gaussian should be used. We have chosen to use the probability measure given by (2.10) :

$$d(\mathbf{p}, \mathbf{w}) \triangleq \frac{1}{2}\left[(\mathbf{p} - \boldsymbol{\mu}_\mathbf{W})^T {\boldsymbol{\Sigma}_\mathbf{W}}^{-1}(\mathbf{p} - \boldsymbol{\mu}_\mathbf{W}) + \log\left(\det(\boldsymbol{\Sigma}_\mathbf{W})\right)\right] , \qquad (2.10)$$

This distance is the addition of the Mahalanobis distance and a volume term. Compared to the pure Mahalanobis distance, the volume term aims at compensating the fact that the Mahalanobis distance of a big cluster tends to make every point very close.

Learning Rate

Our idea is to define the learning rate from the occupancy : the higher the occupancy of a cluster, the better the accuracy of its position and shape is supposed to be ; thus, a small value of ϵ should be used. If, on the other hand, the occupancy is low, the current estimated state of the reference vector can be assumed to be

Algorithm 2.1 Map update with points : pointUpdate

$\{\mathbf{w}_1, \ldots, \mathbf{w}_K\} \leftarrow$ the k Gaussian reference vectors of the cell
$\{\epsilon_j | i = 1, \ldots, k\} \leftarrow$ the associated learning rates
$\mathbf{z} = \mathbf{p} \leftarrow$ the observed point

$n \leftarrow \arg\min_{i=1,\ldots,k} d(\mathbf{z}, \mathbf{w}_i)$
$\boldsymbol{\mu}_{\mathbf{w}_n} \leftarrow \boldsymbol{\mu}_{\mathbf{w}_n} + \epsilon_n(\mathbf{p} - \boldsymbol{\mu}_{\mathbf{w}_n})$
$\boldsymbol{\Sigma}_{\mathbf{w}_n} \leftarrow \boldsymbol{\Sigma}_{\mathbf{w}_n} + \epsilon_n \left[(\mathbf{p} - \boldsymbol{\mu}_{\mathbf{w}_n})^T(\mathbf{p} - \boldsymbol{\mu}_{\mathbf{w}_n}) - \boldsymbol{\Sigma}_{\mathbf{w}_n}\right]$

based on insufficient statistics and the learning rate should be high to permit the reference vector to adapt itself.

In log-ratio the occupancy typically is bounded in $[-o_{\max}, o_{\max}]$ and the learning rate varies within $[\epsilon_{\min}, \epsilon_{\max}]$. For our approach we have chosen a linear mapping between both values :

$$\epsilon(o) = \frac{\epsilon_{\min} - \epsilon_{\max}}{2o_{\max}}(o + o_{\max}) + \epsilon_{\max} \ . \tag{2.11}$$

2.3.4 Map Refinement

The idea is to start with a single gaussian per cell, and then to *refine* it by inserting additional gaussian clusters in order to achieve a balance between representational complexity and accuracy. This refinement is only performed for intermediate and coarse scales. We aim at adding clusters only in those regions where the Gaussian shapes have already converged to their final shapes, which can be deduced from its occupancy. Accordingly, we choose to refine a cell c only if its occupancy probability is above 0.5. In particular, from now on, we will assume that there is a given budget of Gaussians per scale that needs to be allocated in an optimal way through a refinement process. The refinement process is driven by a measure of the representation error. The map is periodically refined by inserting a new cluster in the cell that has the maximum error. After every insertion, the shape of the other clusters in the same cell should be modified accordingly ; this is done by running a clustering algorithm using the cells of the finer scale as input.

The following subsections provide the details of the refinement algorithm : the error metric used to find where to add a new gaussian cluster is introduced in next subsection and secondly we present the clustering algorithm.

Error Computation

To find the cell to refine, we compute an error value per cell for the intermediate and coarse scale. For a given cell, this value is basically the sum of errors between each coarse or intermediate gaussian and the corresponding fine gaussians. So for a given cell, the value is the sum of errors of each coarse or intermediate gaussian. More formally, this value is the sum of the Mahalanobis distance between the center of each gaussian cluster and the Gaussian cluster of the finer scale.

For the cells c^s of the coarse scale, s, having reference vectors (*i.e.* mean values) $\{\mathbf{w}_1, \ldots, \mathbf{w}_K\}$ and finer data at $s-1$: $\{\mathbf{z}_1, \ldots, \mathbf{z}_N\} \in G(\phi(c^s))$, we compute the average distance of each datum to its closest reference vector :

$$\mathcal{E}(c^s) = \frac{1}{N} \sum_{i=1}^{k} \sum_{j=1}^{N} (1 - \epsilon_{\mathbf{Z}_j}) \delta(\mathbf{w}_i, \mathbf{z}_j) \ (\boldsymbol{\mu}_{\mathbf{w}_i} - \boldsymbol{\mu}_{\mathbf{z}_j})^T \boldsymbol{\Sigma}_{\mathbf{Z}_j}^{-1} (\boldsymbol{\mu}_{\mathbf{w}_i} - \boldsymbol{\mu}_{\mathbf{z}_j}) \ , \quad (2.12)$$

where $\delta(\mathbf{w}_i, \mathbf{z}_j)$ is one if $\mathbf{w}_i$ is the closest reference vector to $\mathbf{z}_j$ using the Mahalanobis distance defined by $\mathbf{z}_j$ and zero otherwise. The occupancy is used through the learning rate to assign higher error weights to occupied clusters, disregarding those whose occupancy is low and, in consequence, whose accuracy may still improve without the need of adding extra clusters.

Figure 2.15 illustrates the error computed for each cell of the coarse scale at the beginning of the refinement process where at the coarse level, we only have one gaussian per cell. The top of the figure represents the contribution of each cell of the fine scale to the error of the corresponding coarse cell. Most of cells at fine scale have a small contribution to the error : because at this level, the gaussian is usually a good approximation of corresponding data. The most important error (shown by yellow and red colors) is due to the stairs which are not very well modelized by gaussians. At the coarse level, we see that using only one gaussian per cell gives very poor approximation. For instance, the gaussian in red is a very poor approximation of the real data(see top left figure 2.13) : a part of stairs and a part of one pole and one background wall. In this case, the first refinement process will add one gaussian in this cell.

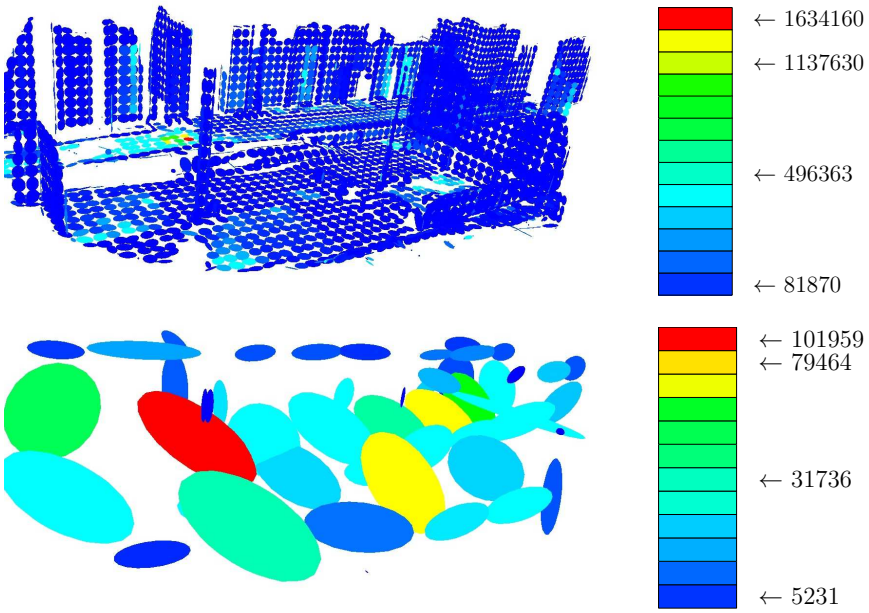


FIGURE 2.15 – Up : fine scale is colored with the magnitude of the contribution to the error at the coarser scale. Down : coarse scale, mean error. Error palettes are on the right.

Clustering for Map Refinement

In this section, we describe our clustering approach for map refinement. This method solves a hard clustering problem : we are looking for a partition $C^* = \{C_1^*, \ldots, C_k^*\}$ of the $G(\phi(c^s))$ into k classes represented by k reference vectors that minimizes the clustering distortion :

$$\mathcal{E}_{(C^*,\{\mathbf{w}_1^*,\ldots,\mathbf{w}_k^*\})} = \underset{\{C_1,\ldots,C_k\},\{\mathbf{w}_1,\ldots,\mathbf{w}_k\}}{\arg\min} \sum_{i=1}^{k} \mathcal{E}_{C_i}(\mathbf{w}_i) \tag{2.13}$$

This is done by using the well known k-means clustering algorithm [Llo82].The optimal clusters are computed iteratively from the set of reference vectors : each datum is associated to its closest reference vector ; then, the minimizer of each cluster energy is computed.

An important drawback of k-means is that is highly dependent on initialization. Moreover, even if the algorithm is guaranteed to converge, it often gets stuck in local minima of $\mathcal{E}_{W^*}$. To get out of the local minima a so called "swap" procedure is used. One cluster is chosen, either randomly or because of its short

distance to a different cluster with more elements. Then, a simulation is done by reallocating that reference vector to the region of space with maximum error. If the resulting partition has a lower clustering distortion, the result is accepted and a new simulation is done. Otherwise, the result is rejected and the procedure stops.

2.4 Conclusion

In our first contribution [29, 5], we have presented a method to perform SLAM with detection of moving objects. This method is 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 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 (see sections 4.3, 4.4 and 4.6).

In our second contribution [21, 4], we have proposed a comprehensive framework to build two and three-dimensional maps from range data. The proposed representation enhances the accuracy of previous approaches by enabling the presence of several Gaussians per cell. These Gaussians are added by means of a refinement algorithm which inserts them where the representation error is maximum. This contribution has been validated by modelling a 3D environment observed by a laser scanner (see section 4.5).

Chapitre 3

Detection And Tracking of Moving Objects

ïż£

3.1 Introduction

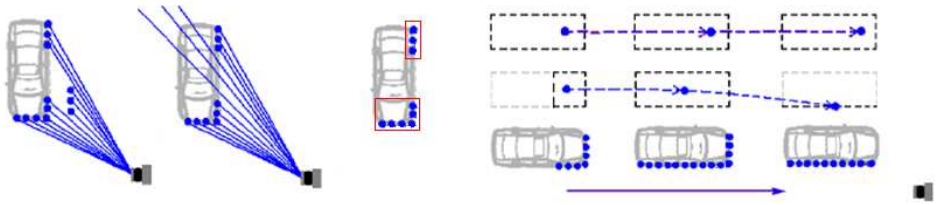
While SLAM as described previously are responsible for modeling static part of the environment, DATMO deals with dynamic part of the environment. Its objective is to detect and track moving objects which enables the prediction of their future behaviors. Many tracking works suppose that the measurements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

3.1.1 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.

The idea of feature-based method is to define a set of simple features and to use these features to detect specific moving entities. For instance, [SBFC01] presented a method using simple features to detect people in office environments. 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.

In section 2.2, we presented a motion-based method for moving object detection from a moving ground vehicle using occupancy grid. 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. This method is simple, clear and can be implemented in real-time.



(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 3.1 – Vehicle model can help better interpreting laser data.

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 [PT08], these mentioned methods pose several problems in particular 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 3.1(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 3.1(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. [PT08] proposed to use flexible models to detect moving vehicles. [PT08] introduced a method constructing a virtual grid in 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.

3.1.2 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* [BSF88]. Filtering methods deal with the problem of tracking one specific object which consists in estimating its state from given observations over time using a bayesian filter. 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.

3.1.3 Filtering - Multiple Dynamic Model

When the dynamics of a mobile object can be represented by a single model, we can apply directly Bayesian filtering methods such as the Kalman filter,

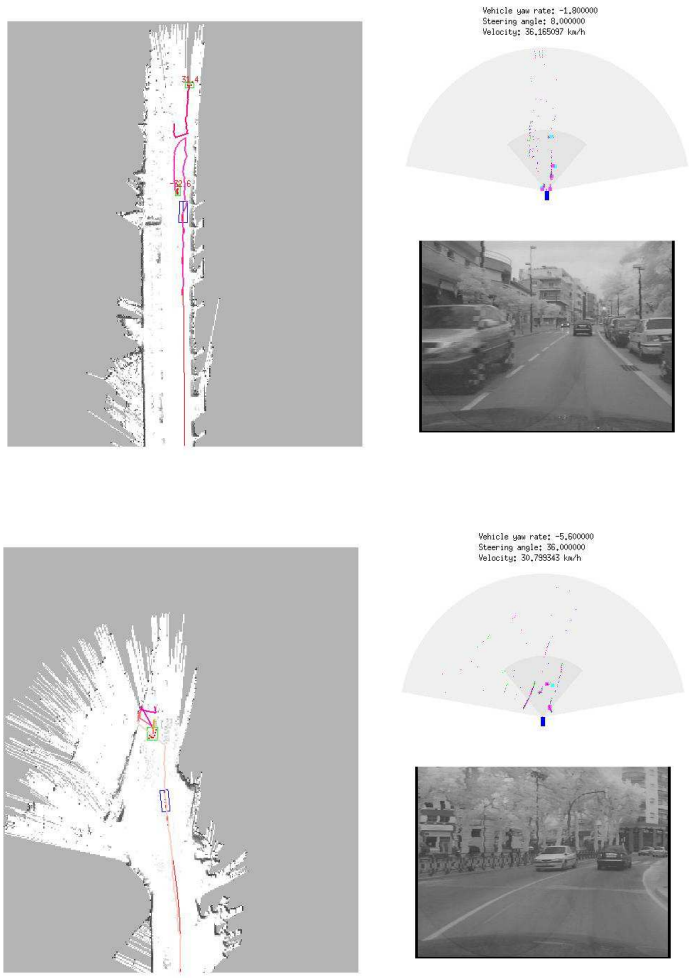


FIGURE 3.2 – two examples of wrong associations due to a wrong choice of set of motion models

Extended KF or Particle filter to estimate object dynamic states (see section 2.1.1). 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 generally required. The definition of this set of dynamic models is crucial for tracking of moving objects especially for the prediction phase. For instance, if motion models are defined with too low speed, during tracking we obtain a frequent lost of tracked objects, because detections are frequently outside of gating window. On the contrary, if motion models are

defined with too high speed, we obtain wrong associations because predictions are far from detected objects. In this case, association is sometimes done with an other object as illustrated in figure 3.2(a). For the same reasons, a tracked vehicle is sometimes associated with false alarms on road border as illustrated in figure 3.2(b).

To deal with these motion uncertainties, Interacting Multiple Models (IMM) [MABSD98, RLP03] have been successfully applied in several applications [Bla86, BB95, BP00a]. The IMM approach overcomes the difficulty due to motion uncertainty by using more than one motion model. The principle is to assume a set of models as possible candidates of the true displacement model of the object at one time. To do so, a bank of elemental filters is ran at each time, each corresponding to a specific motion model, and the final state estimation is obtained by merging the results of all elemental filters according to the distribution probabilty over the set of motion models.

Also, the probability the object changes of motion mode is encoded in a Transition Probability Matrix (TPM) which gives the distribution $P(\mu_t | \mu_{t-1})$, *i.e.* the transition between motion models which is assumed Markovian. Nevertheless, to apply IMM on a real application a number of critical parameters have to be defined for instance the set of motion models and the transition probability matrix(TPM). In practice, the TPM is often assumed known and is chosen *a priori*. Even if the design of TPM for different applications have been studied [BW92][MP98], its definition and construction do not rely on the real on-line data and so such TPMs can not be adapted to the real application. therefore it is an important issue to automatically adapt the TPM to fit the application of the IMM algorithms.

Few publications address this specific problem and in most of them, simplest problems are considered (binary system case) [SKF72] or the TPM is assumed to belong to a set of finite candidates TPMs [Tug82b]. Also, in [GH00] modes transition chain is formalized as a bayesian network and a maximum likelihood estimator of the TPM is proposed. However, [JLA03][JL04] give algorithms to adapt on-line the TPM under the assumption that the unknown TPM is random but time-invariant. In all these works, assumption on estimated TPM is relatively strong or algorithm complexity is too hight to address real time applications.

An iteration of an IMM with M filters is composed of four phases.

1. Filtering for each filter (for $i = 1, 2, \ldots, M$) :
   - Prediction phase using the displacement model of the filter ;
   - New estimation of i^{th} filter $P(x_t^i)$ with observation z_t.
2. Update of probability distribution over the set of models (for $i = 1, 2, \ldots, M$) :
   - Compute likelihood L_t^i that the new observation z_t corresponds to model i : $L_t^i = P(z_t|\mu_t^i)$;
   - Compute probability $P(\tilde{\mu}_t^i)$ to be at time t in the model i whatever the model j at previous time was :
     $P(\tilde{\mu}_t^i) = \sum_j P([\mu_t = i] \mid [\mu_{t-1} = j])P(\mu_{t-1}^j)$;
   - Compute probability to be in model i at time t :
     $P(\mu_t^i) = L_t^i P(\tilde{\mu}_t^i)$.
3. Fusion of estimations :
   - $P(X_t) = \sum_i P(\mu_t^i)P(x_t^i)$.
4. Reinitialisation of each filter (for $i = 1, 2, \ldots, M$) :
   - Compute probability $P(\tilde{\mu}_t^i)$ to be in model i at time t :
     $P(\tilde{\mu}_t^i) = \sum_j P([\mu_t = i]|\ [\mu_{t-1} = j]) \times P(\mu_{t-1}^j)$;
   - Compute probability to go from a specific model j at time $t-1$ knowing that we are in the model i at time t : $P(\mu_{t-1}^j|\mu_t^i) = P([\mu_t = i]|\ [\mu_{t-1} = j])/P(\tilde{\mu}_t^i)$;
   - Initialization at time t, using estimations at time $t-1$: $P(x_t^i) = \sum_j P(x_{t-1}^j)P(\mu_{t-1}^j|\mu_t^i)$;

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 3.3 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.

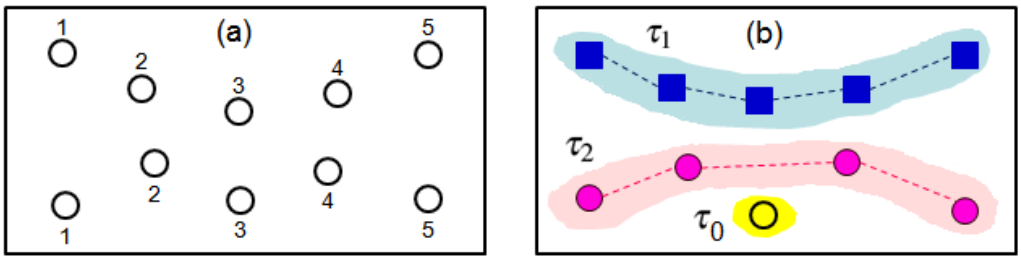


FIGURE 3.3 – Example of data association. a) A set of observations Y (each circle represents an object observation together with numbers representing time steps. b) A solution of data association which is comprised of two tracks and a false alarm.

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 generate 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) [BP99]. 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 [Rei79, CH96a]. 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 [Bla04] 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 [BSF88] 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 [SBFC01]. 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 [BSF88] 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 [SN05, YCMW06, ZNW08]. 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. [ORS04] 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.

3.1.4 Synthesis and Contributions on DATMO

To solve the SLAM and DATMO problem, the first step is to detect moving entities. In section 2.2, we proposed a motion-based method to solve this problem. As described previously, model-free approaches pose several problems in particular use with laser sensors. To overcome this difficulty, in section 3.2, we introduce a model-based approach [23, 5] to improve the detection process and also the tracking process. 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.

Regarding the multi objects tracking problem, it consists in general of two parts : *Filtering* and *Data Association* [BSF88]. Regarding filtering, the definition of an appropriate dynamic model is one of the most important problem to solve. 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 generally required. The definition of this set of dynamic models is crucial for tracking of moving objects especially for the prediction phase. In section 3.3, we define a method to guide us in the choice of motion models and to estimate the interactions between this set of motion models. Regarding Data Association problem, Markov Chain Monte Carlo (MCMC) have achieved notable success in vision tracking. This method is able to provide a true approximation scheme for the optimal Bayesian filter and is able to perform data association using multi-scans and some constraints on dynamic models to improve the robustness of the data association process. In section 3.2, we describe how we

used MCMC to solve the data association problem.

3.2 DATMO Formulation

3.2.1 Introduction

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 3.4a)). 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 [ZZT00].

The remaining of the section is organized as follows. In the following section, we introduce a general formulation of the moving object detection and tracking problem and detail this solution in section 3.2.3. In Section 3.2.4, we present the algorithm to find the optimal trajectories of moving objects using a spatio-temporal MCMC sampling method.

3.2.2 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 :

$$\omega^* = \arg\max_{\omega \in \Omega} P(\omega|Z) \tag{3.1}$$

In the following, we describe solution space Ω and object shape models.

Solution Space

A solution ω for the Equation (3.1) includes a set of K trajectories (tracks) of moving objects appeared during the tracking over the last T frames [1] :

$$\omega = \{\tau_1, \tau_2, ..., \tau_K\} \tag{3.2}$$

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

$$\tau_k = \{\tau_k(t_1), ..., \tau_k(t_{|\tau_k|})\} \tag{3.3}$$

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 will be described in the next subsection related to objects model.

FIGURE 3.4 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 3.5). 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

1. Here, we are interested in finding the trajectory of all moving objects present in the environment knowing the observations : $P(\omega|Z)$. This is slightly different from the classical problem of DATMO where we are interested in finding the list of all moving objects present in the environment knowing the observations : $P(O|Z)$.

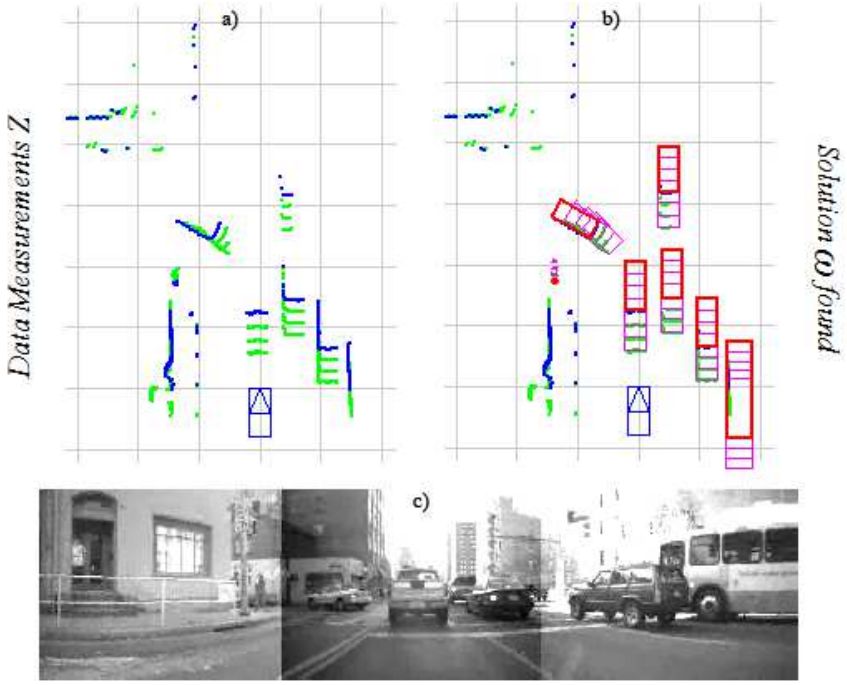


FIGURE 3.4 – Example of an interpretation of moving object trajectories from a laser data sequence. (a) Data comprised of four scans consecutive : in blue is current scan and in green are scans in the past ; (b) A solution found including seven tracks of four cars and one bus represented by red boxes and one pedestrian represented by red dots which are imposed on the range data ; (c) situation reference.

problems.

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.

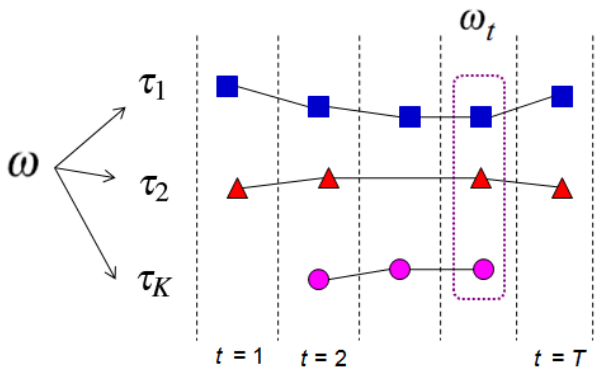


FIGURE 3.5 – Illustration for the notation in use.

Object models

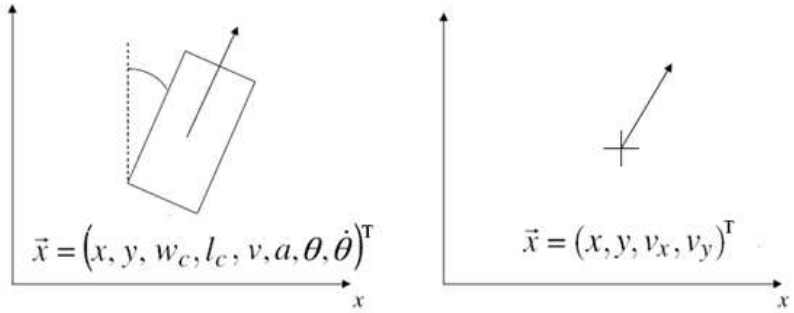


FIGURE 3.6 – Box model and point model to represent moving objects.

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 [PT08] 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 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 3.6).

3.2.3 DATMO Decomposition

According to the Bayes rule, the posterior probability $P(\omega|Z)$ is decomposed into :

$$P(\omega|Z) \propto P(\omega)P(Z|\omega) \tag{3.4}$$

where $P(\omega)$ and $P(Z|\omega)$ are the prior and likelihood probabilities respectively. In the following, we describe the prior model and the likelihood model.

Prior Probability

The prior $P(\omega)$ in the Equation (3.4) 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 :

$$P(\omega) = \prod_{k=1}^{K} P(\tau_k) \tag{3.5}$$

The probability $P(\tau_k)$ encodes 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 3.7). 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 :

$$P(\tau_k) = P_L(\tau_k)P_T(\tau_k) \tag{3.6}$$

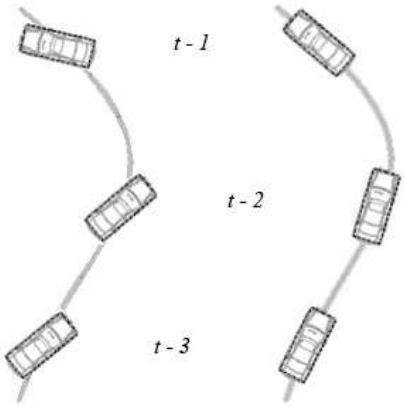


FIGURE 3.7 – Temporal consistency of a track. Obviously the arrangement of cars on the right is more relevant to a correct motion than that on the left.

Likelihood Probability

The likelihood $P(Z|\omega)$ (3.4) 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.

Let $Z^{(d)}$ denote all laser measurements which is identified from the step described in section 2.2 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 :

$$p(Z|\omega) = \prod_{i,j=1}^{T} P(z_i|\omega_j) \prod_{i,j=1}^{T} P(z_i|z_j^{(s)}) \tag{3.7}$$

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).

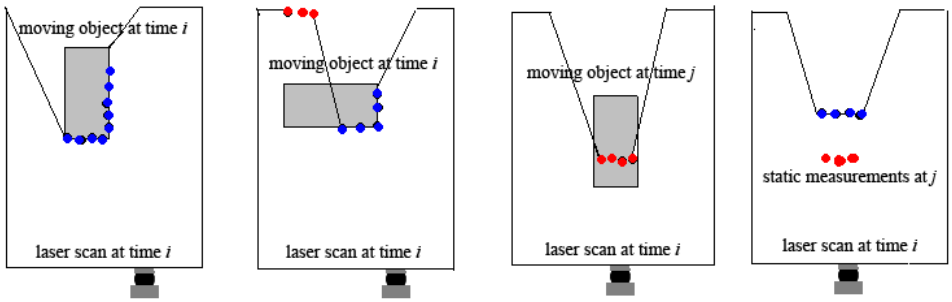


FIGURE 3.8 – Four types of constraint used to compute the likelihood. The red dots are measurements that violate the laser visibility constraint.

The first term is then further decomposed :

$$\begin{aligned}\prod_{i,j=1}^{T} P(z_i|\omega_j) &= \prod_{i} P(z_i|\omega_i) \prod_{i\neq j} P(z_i|\omega_j) \\ &= \prod_{i} P_{M_1}(z_i^{(d)}|\omega_i) P_{M_2}(z_i^{(s)}|\omega_i) \prod_{i\neq j} P_{M_3}(z_i^{(s)}|\omega_j)\end{aligned} \tag{3.8}$$

The second term in (3.7) is rewritten as :

$$\prod_{i,j=1}^{T} P(z_i|z_j^{(s)}) = \prod_{i\neq j} P_{M_4}(z_i|z_j^{(s)}) \tag{3.9}$$

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 3.8, 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.

3.2.4 Efficient DATMO Computation using MCMC

We want to find the solution ω that maximizes the posterior probability defined in (3.4). However, as stated in Section 3.2.2, 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. Although the problem is restricted within a sliding window, searching in the solution space for Equation (3.4) is still challenging.

Dealing with this difficulty, we employ a Markov chain Monte Carlo (MCMC) method. We introduce the basic idea of MCMC in the following. In next subsection, we detail how we use the detection results from the previous chapter as a coarse detector to generate potential moving objects hypotheses with predefined models that helps to drive the search more efficiently. Finally, we explain how we represent the hypothesis space and use it to generate hypothesis.

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

$$A(\omega_{n-1}, \omega') = min(1, \frac{\pi(\omega')}{\pi(\omega_{n-1})} \frac{q(\omega_{n-1}|\omega')}{q(\omega'|\omega_{n-1})}) \tag{3.10}$$

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

The overview of MCMC algorithm is shown in Algorithm 3.1. This is the well-known Metropolis-Hasting algorithm [Tie96]. The proposal probability $q(.)$ is called the *dynamics* of the Markov chain.

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

Algorithm 3.1 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

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 [ZZT00] that earns the approach its name *Data-Driven* MCMC method (DDMCMC).

Moving object hypothesis generation

To make the proposals more informative, we take advantage of the detection module in section 2.2.3 which can help to identify moving parts of dynamic objects. Combined with suitable object models, all possible object hypotheses are generated at location of these detected motion evidences. The principle is that we want to keep the detection rate high and accept false alarms to cover as many potential moving objects as possible. These rough hypotheses provide initial proposals for the MCMC sampler (Algorithm 3.1) that performs a finer search over the spatio-temporal space to find the most likely trajectories of moving objects with a maximum of posterior probability.

The goal of the first step is to build a set of dynamic segments corresponding to potential moving objects. Note that objects can be divided into several parts so that several segments might be related to the same object.

Figure 3.9 illustrates our detection process. In the figure, the bottom image describes a situation when the host vehicle moving along the street seeing two cars

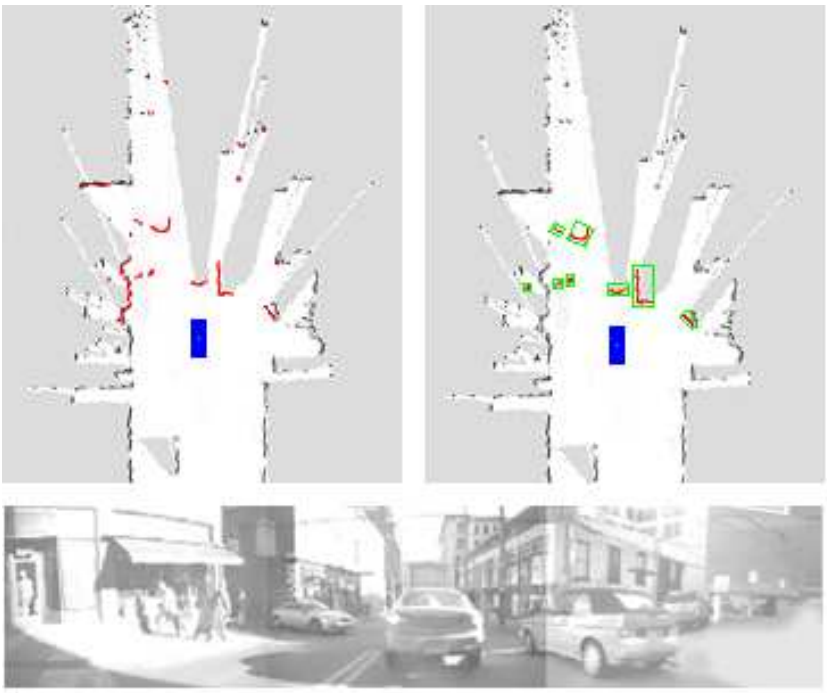


FIGURE 3.9 – Object detection based on occupancy grid.

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.

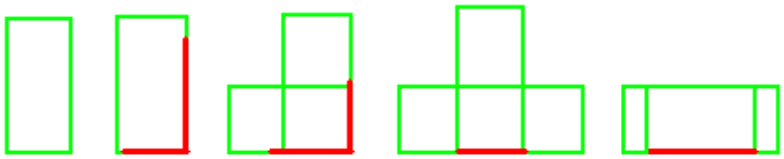


FIGURE 3.10 – Illustration of fitting object box model (green) to L-shape and I-shape segments (red). The last two shapes show that using box model helps connecting discontinued segments.

FIGURE 3.10 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 3.1).

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. 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 3.11 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

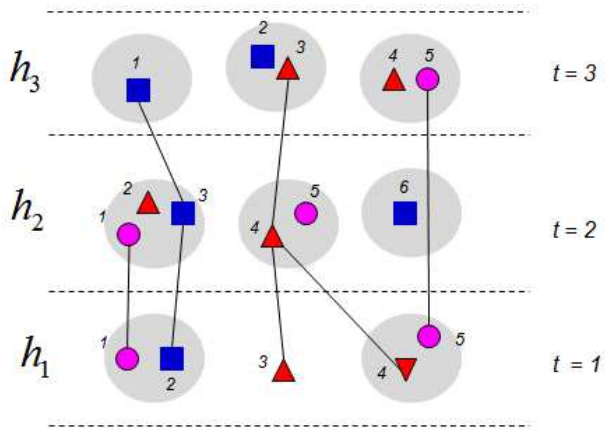


FIGURE 3.11 – Example of a neighborhood graph of moving object hypotheses generated over three frames.

nodes are displayed inside gray circles. Parent-child nodes are connected by segments.

Markov chain dynamics

The Markov chain dynamics correspond to sampling the proposal distribution $q(\omega'|\omega_{n-1})$ described in Algorithm 3.1 (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}.

FIGURE 3.12 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 3.2.4). 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

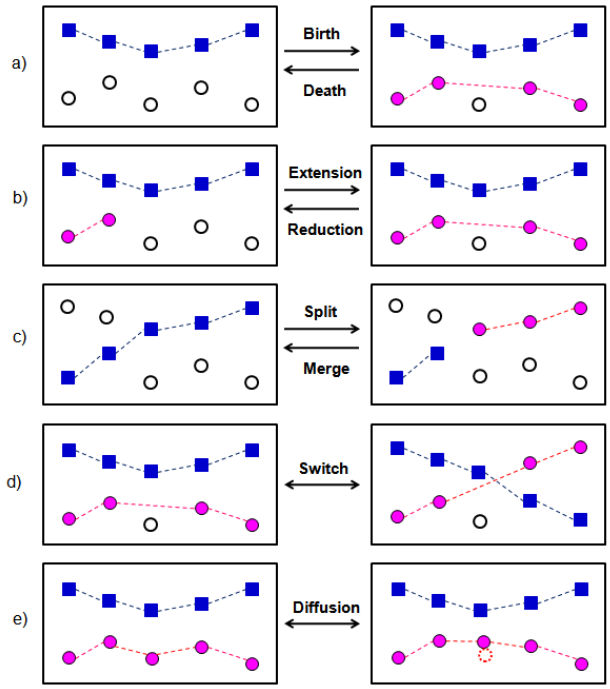


FIGURE 3.12 – Dynamics moves of the Markov chain.

that the entire solution space can be explored if the MCMC sampler constructed this way is run with unlimited resources.

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 (3.10). 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 (3.4). 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 MCMC approach is that, we only need 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}.

3.3 Adaptative Interactive Multiple Models

3.3.1 Introduction

In previous section, we develop 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. One of the prior information for each track of ω is $P_T(\tau_k)$. 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. 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 $(x, y, \theta, \dot{\theta}, v, a)$ 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).

In this section, we detail our second contribution [33, 6] on DATMO. We present a method to automatically model the interactions between the different motion models instead of defining them *a priori* and also to guide us in the choice of motion models. The main advantage of this method is to design multiple motions models that are very close to the real motions of tracked objects. Actually, the motion models of objects tracked in section 3.2 have been choosen using this method. In next subsection, we detail the principle of our approach and explain how we use it in the framework of multiple objects tracking. In the last subsection, we describe how we can use this principle to find the pertinent model for an IMM.

Actually, TPM model changes in displacement of moving objects. So, this set of changes of displacement could be seen as a set of behaviors. For instance, when moving in an environment, some objects will have similar behaviors (e.g pedestrians crossing a parking or cars moving in a specific direction). These similar behaviors will cause the same changes of displacements. To obtain a classification of behaviors of objects, we use automatically online adapted TPM to characterize typical displacements of objects. Afterwards, this classification is used to specify a precise TPM for a given object and so to obtain more adapted modelization of typical displacements of this object.

3.3.2 Principle

Algorithm 3.2 TPM adaptation

Adaptation of TPM($T^1 = (z^1_1, ..., z^1_{length(1)}), \ldots, T^N = (z^N_1, ..., z^N_{length(N)})$)
for all Trajectory T^i **do**
  $S_i \leftarrow [\,]$
  /* Obtaining of μ_1,...$\mu_{length(i)}$ using IMM */
  for all observation z^i_j in T^i **do**
    $\{\mu_j,\ P(X)\} \leftarrow$ IMM(z^i_j)
    $S_i \leftarrow S_i \cup [\mu_j]$
  end for
  /* Determination of the most likely sequence of models MLSM */
  MLSM $\leftarrow Viterbi(S_i)$
  /* Quantification of most likely transitions between models */
  for all Couple ($SMP_k,\ SMP_{k+1}$) in MLSM **do**
    $i \leftarrow$ MLSM$_k$
    $j \leftarrow$ MLSM$_{k+1}$
    $F_{ij} = F_{ij} + 1$
  end for
end for
/* Update of TPM */
TPM $\leftarrow$ Normalization(F)
Return TPM

The principle of the method is the following. For a given number N of trajectories [2], we build sequences of associated motion models probabilities. And then, using these motion models probabilities, the TPM is adapted and reused in the IMM filters for the next estimations. The TPM is initially chosen to be uniform. In more details, algorithm 3.2, given in pseudo-code, is the algorithm defined to compute one adaptation of the TPM. An adaptation of the TPM is done after a given number N of trajectories obtained from past tracked objects, to update TPM using a window on trajectories.

The algorithm given in pseudo-code is the algorithm developped to determine the TPM of an IMM. This algorithm uses a predefined IMM (i.e., a set of motion models) with an uniform TPM and a collected set of N trajectories of past objects.

2. A trajectory is defined as the sequence of observations (provided by a detector) of a past object. The observations composing the i^{th} trajectory are noted : $T^i = (z^i_1, ..., z^i_{length(i)})$ where z^i_j is j^{th} observation of the i^{th} trajectory

The algorithm delivers as an output a new estimation of the TPM. Our algorithm is composed of a main loop over these N trajectories (*cf.* lines 3-19). Inside this loop, the processing on each trajectory is divided in 3 steps :

1. **Computation of distribution on motion models using IMM ; (*cf.* lines 3-8)**
   For each observation, the position of the object is estimated by the IMM filter. So, the distribution μ_j on motion models is computed (line 6). So, we obtain for a given trajectory a sequence S_n of distributions on models (line 7).
2. **Determination of the most likely sequence of transitions between motion models ; (*cf.* lines 9-10)**
   During this step, we determine the most likely sequence of transitions between motion models knowing a given trajectory (i.e., a given sequence of K observations). More formally, we are interested in determining the most probable sequence $\mu_0\ \mu_1...\mu_K$ knowing the trajectory $z_0\ z_1...z_K$ that maximizes

   $$\max\{P(\mu_0\ \mu_1...\mu_K \mid z_0\ z_1...z_K)\} \tag{3.11}$$

   Using bayes rule and taking into account that the TPM is markovian, we have :

   $$P(\ \mu_0\ \mu_1...\ \mu_K \mid z_0\ z_1...z_K) = \alpha P(\mu_0) \prod_{k=1}^{K} P(\mu_k|\ \mu_{k-1})\ P(z_k \mid \mu_k)$$

   where $P(\mu_k|\ \mu_{k-1})$ is the TPM and $P(z_k \mid \mu_k)$ is the likelihood of an observation knowing the motion model. The number of possible sequence of motion models is equal to KM^2 with M the number of motion models and K the length of the sequence of observations.
   To find the sequence $\mu_0\ \mu_1...\mu_K$ which maximizes equation 3.11, we use the Viterbi algorithm [For73].
3. **Quantification of the most probable transitions between models. (lines 11-17)**
   The last step of the algorithm consists in quantifying the number of

transitions between models using the previous most likely sequence of transitions between models (lines 11 to 17). For this, we use an intermediate matrix named matrix of frequencies. This matrix models the number of transitions from one model to an other. We note F this matrix and F_{ij} gives us the number of transitions from model i to model j. The update of this matrix of frequencies F is done by counting over the most likely sequence of transitions between models : for each transition in this sequence, the corresponding element in the matrix is incremented (line 15).

At the end of the algorithm, after processing the set of N trajectories, an estimation of the TPM is obtained by normalizing the matrix F (line 21 and 22).

Collecting the set of trajectories

To determine the TPM, we need to collect a set of trajectories. In section 3.2, we define a complete framework for DATMO. In this framework, we used IMM filters to track objects. A description of the motion models used for these IMM filters is given in section 3.3.1 and before adaptation we consider that the TPM for each IMM filter is uniform. For each tracked object, we memorize its complete trajectory and when it leaves the field of view of the laser scanner, we collect its trajectory for adaptation of TPM.

3.3.3 Continuous and online adaptation of TPM

One of the advantage of our method is to determine the TPM associated with a given IMM and a set of trajectories. In this section, we describe how we can use this principle to continuously and online adapt the TPM. The basic idea is the following one : we periodically collect the set of trajectories of objects that have left the environment and use them to update the TPM. The principle of the method is illustrated in figure 3.13. Regarding the framework described in section 3.2, at each timeframe, we obtain for each object the corresponding sequence of observations. We split the set of sequence of observations in two subsets : objects still present in the environment and objects that have just left the environment. For objects that are still present in the environment, we use the current estimate of TPM to continue to track them since they leave the environment. So we only use sequence of observations of objects that have left the environment to update the

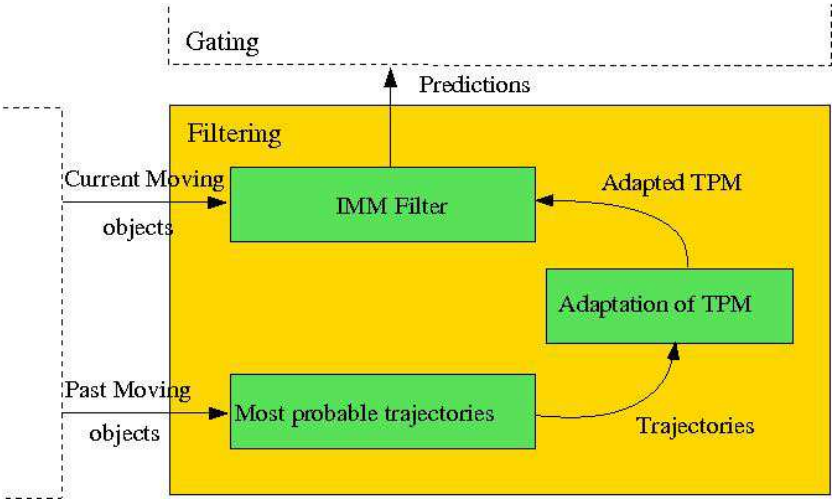


FIGURE 3.13 – Use of adaptative IMM for multi objects tracking

TPM. After collecting a set of N trajectories, we decide to update the TPM and to use this new TPM to track future objects. The main advantage of this method is to continuously and online adapt the TPM.

3.3.4 Use of adaptation to define the set of motion models for IMM

Introduction

To use our adaptative method, we have to define the set of motion models composing the IMM filter. The choice of these models and their interactions *via* TPM is critical to obtain an efficient filtering. The adapted TPM directly depends on the definition of these models. In this section, we present how results of adaptation could be used to check if the set of choosen motion models is pertinent or not.

First of all, we define a set of initial motion models. Using these motion models, we propose a method to redefine the parameters of the models and the set of models based on the results of adaptation.

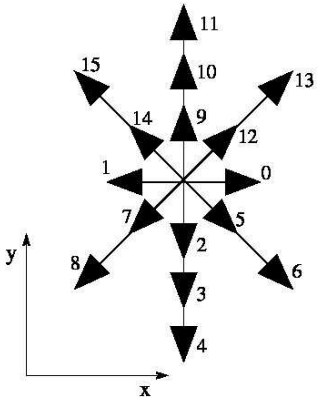


FIGURE 3.14 – Set of initial motion models

Initial choice of set of motion models

The choice of the set of filters composing an IMM is essentially based on two criteria : speed and orientation of motion models. As we are concerned with tracking vehicles or pedestrian mostly on road environments, we choose to have 8 directions and 2 speeds. Moreover, we decide to have 3 translational speeds in the direction of the ego vehicle and only 1 translational speed in the direction perpendicular to the direction of the ego vehicle. The 16 defined motion models are illustrated in figure 3.14.

Redefinition of motion models after adaptation

Analyzing adapted TPM, some rules can be defined to modify the initial set of motion models and as a consequence improve the quality of tracking. For instance, an error on the choice of speed could be detected for a set M of m motion models defining the same direction but with different speeds, if we have after adaptation almost all the transitions for only a subset of M to itself. If we look at the TPM, we will see one or several "picks" for a given direction. For instance, in figure 3.15, on the diagonal (corresponding to motion models 7,8, 12 and 13 on figure 3.14), we see 3 "picks" corresponding to transition of a given motion model to itself. Intuitively, the solution is to redefine speeds of M taking as reference speed, motion models with the most important transitions corresponding to the "picks" in TPM.

Formally, let $\{V_1, V_2, ..., V_m\}$ be the set of speeds increasingly ordered for a

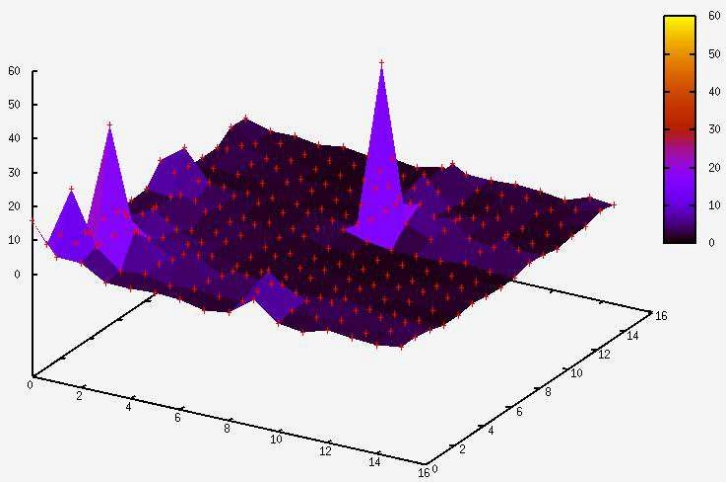


FIGURE 3.15 – Example of adapted TPM with a wrong initial set of motion models

given direction. If there exists i with $i \in [1, m]$ as $TPM(i, i) > G$ et $TPM(j, j) < \epsilon$ and $TPM(i, j) < \epsilon,\ \forall j \neq i$, with G a high value and ϵ a negligeable value, so $V_1 = V_i - \Delta$, $V_M = V_i + \Delta$ with $\Delta < \frac{|V_i - V_{i-1}| + |V_i - V_{i+1}|}{2}$ and $V_2, .., V_{m-1}$ defined to obtain a regular discretization on $[V_1, .., V_m]$.

An error on orientations will be detected in the same way by too important transitions for a given direction and negligeable transitions for neighboring directions. We will also see one or several "picks" for a given direction. The solution is also to discretize orientations for motion models which transitions are negligeable.

Formally, let $\{\theta_1, \theta_2, ..., \theta_M\}$ be the set of defined orientations in different models for a given orientation. If there exists i with $i \in [1, m]$ as $TPM(i, i) > G$ et $TPM(j, j) < \epsilon$ $TPM(i, j) < \epsilon,\ \forall j \neq i$, with G a high value and ϵ a negligeable value, so $\theta_1 = \theta_i - \Delta$, $\theta_m = \theta_i + \Delta$ avec $\Delta < \frac{|\theta_i - \theta_{i-1}| + |\theta_i - \theta_{i+1}|}{2}$ et $\theta_2, .., \theta_{m-1}$ defined to obtain a regular discretization on $[\theta_1, ..., \theta_m]$.

3.3.5 Interacting Multiple Models based Classification of Moving Objects

In this section, we detail how we improved our adaptive method to track objects by adding a classification module [17]. The intuitive idea is to consider

the set of changes of displacement as a set of behaviors. The goal is in a first stage to compute TPM classes modeling typical object behaviors. The second stage aims to identify which behavior and thus at which TPM class it belongs. This identification permits afterwards to assign a specif TPM to moving objects, modeling in a better way the tracked object behavior.

In the first part, we present the general principle. The class computation is exposed in a second part. In the third part, we explain how a class is assigned to each track. The way the classification module output is used in the multiple object tracking is shown in the fourth part. Finally a conclusion is given.

Principle

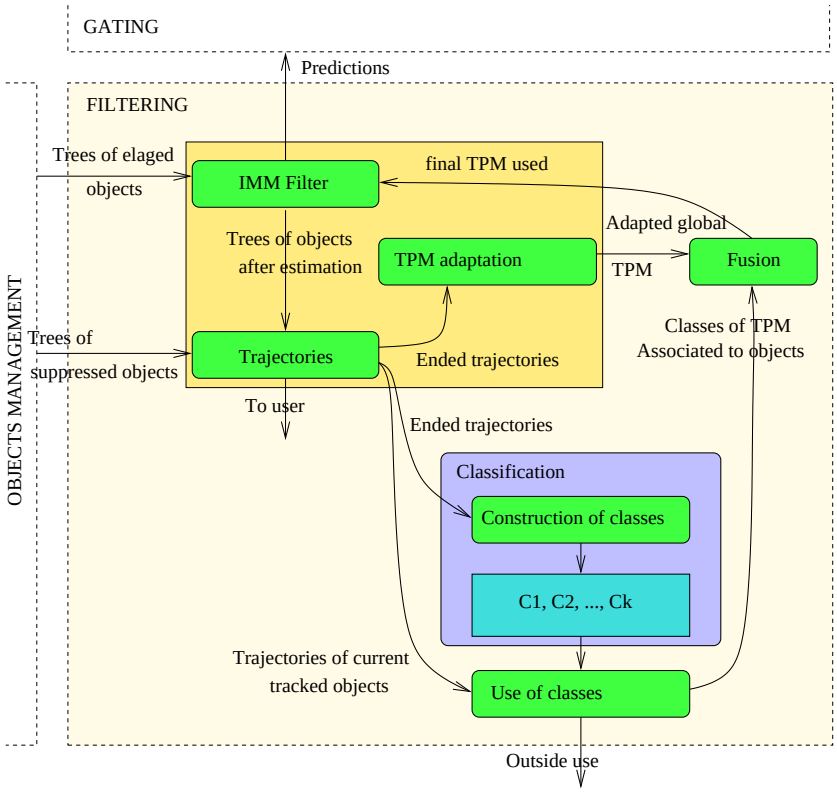


FIGURE 3.16 – General schema showing the classification use in the filtering process

The method's general principle is illustrated in the figure 3.16. The filtering part is again represented with yellow boxes as defined in the previous section.

To specify for each track a TPM using a computed set of TPM classes, we add three blocs at our adaptive filtering and the algorithm 3.3 gives the corresponding algorithm :

Algorithm 3.3 Filtering using the classification module

1: /* Filtering takes as input current and past moving objects trajectory */
2: **Filtering**($T^{1:I}_{current}$, $T^{1:J}_{past}$)
3: /* Compute the TPM corresponding to each trajectory */
4: **for all** i from 1 to I **do**
5: $TPM^{i}_{current} \leftarrow$ computeTPM($T^{i}_{current}$)
6: **end for**
7: **for all** j from 1 to J **do**
8: $TPM^{j}_{past} \leftarrow$ computeTPM(T^{j}_{past})
9: **end for**
10: /* Class building */
11: $C_1, .., C_k \leftarrow$ CEM($TPM^{1:J}_{past}$)
12: /* Class utilisation */
13: **for all** i from 1 to I **do**
14: **if** $Traj_{1:I} > L$ **then**
15: $TPM^{c(i)}_{current} \leftarrow$ MostProbableClass($TPM^{i}_{current}$)
16: **end if**
17: **end for**
18: /* TPM Fusion */
19: /* Global TPM adaptation */
20: $TPM_g \leftarrow$ **TPM_Adaptation**($T^{1:J}_{past}$)
21: **for all** i from 1 to I **do**
22: **if** TPM_i exist **then**
23: Return $\alpha(TPM^{c(i)}_{current} + TPM_g)$ in IMM
24: **else**
25: Return TPM_g in IMM
26: **end if**
27: **end for**

1. a classification component (lines 3 to 11 of the algorithm 3.3) is added in order to take as an input the same trajectories used in the adaptation method *i.e* the most probable trajectories of past moving objects. This component is shown in blue on the figure 3.16. It computes a set of TPM classes, $C_1, C_2, \ldots, C_K$, which are stored in memory.

   For each trajectory, we compute the distribution over motion models as a

sequence. This sequence is then used to compute a frequency matrix which is normalized to obtain a TPM. The process is the same as the one used in the adaptation process but in this case the obtained TPM is local to each past moving object and not global. In this way, the computation for J moving object gives J TPM called *local* TPM but only one TPM called *global* is updated during the adaptation process.

The purpose is then to compute classes using the set of J local TPM. The class computation is performed using Classification Expectation Maximization (CEM) [CG91] in which classes are represented by multidimensional Gaussians. This classification algorithm have been chosen for its Gaussian modelization and its iterative mode of action permitting to obtain an output after any iteration. To apply CEM, the local TPM are simply transformed in vector. Thus the $M \times M$ matrix are transformed in M^2 length vectors. By this way, classes are modeled by Gaussians of mean and covariance matrix with respectively a size of M^2 and $M^2 \times M^2$.

In practice the classes computation is made every J achieved moving objects as per the adaptation and it is performed using the whole history of achieved trajectory.

Furthermore, it is necessary to define the number of classes to use the CEM algorithm. Therefore this number depends on the user and the application.

2. In a class usage component (lines 12 to 17) where each current moving object is assigned to a class. The aim is to compute the most probable class for a given moving object and thus identify the object behavior in order to assign it a more specific TPM, in order to enhance the next predictions. In this part, only trajectories with a significant length L are considered in order to obtain coherent classification.
3. The last component (lines 18 to 27) combines the TPM obtained by adaptation and the one computed using the classification. It permits to specify the used TPM while keeping the advantages of the adaptive TPM *i.e* the global behavior of object and the robustness to behavior variation. The result will then by used for each moving object for the next filtering step.

   Nevertheless, as a specific TPM is only computed for trajectories with a length above L, it is necessary to use only the global TPM for smaller

trajectories. A trajectory with a length above L will take advantages of both specific and global TPM. As both TPM are important and have complementaries advantages, the fusion is performed by adding and normalizing the two associated matrix.

3.4 Conclusion

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 section 2.2 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.

In this chapter, our second contribution is a set of methods to guide us in the definition of dynamic models and to classify objects based on their trajectory.

Firstly, we propose a method of adaptative filtering based on IMM, which enables to automatically and online adapt transitions between a predefined set of dynamic models. Taking into account past moving objects trajectories, the Transition Probability Matrix is online and constantly updated, so transitions between motion models fit better and better to real motions of moving objects and as a consequence it improves quality of tracking. Nevertheless, it is still necessary to define the set of motion models.

To overcome this difficulty, in a second step, we analyze results of adaptation of TPM in order to validate the initial set of motion models. Actually, we experimentally showed that this adaptation enables to check pertinence of defined motion models and detect useless motion models and models that should be redefined. So, we used this method to define the set of motion models in our framework. Finally, this method provides a tool to define the set of motion models.

We extend this method with a behavior based classification module. Actually, TPM models changes in displacement of past moving objects. So, these set of changes of displacement could be seen as a set of behaviors. So we propose a method to classify trajectories of moving objects using the corresponding TPM. Afterwards, this classification is used to specify a precise TPM for a given object and so to obtain more adapted modelization of typical displacements of this object.

Chapitre 4

Applications

4.1 Introduction

This chapter details how the previous contributions on SLAM and DATMO have been integrated and validated on some experimental platforms. This activity is an important part of our research work, because it enables us to make the link between our theoretical contributions and real applications. We describe our contributions on several projects : national project puvame, european IP PReVENT, NAVLAB public dataset, 3D Mapping with an industrial partner and european project Intersafe2. For each projet, we briefly summarize the goal of the project and present the experimental platforms. Afterwards, we explain which contributions have been used and how they have been integrated. Finally, we give some experimental results.

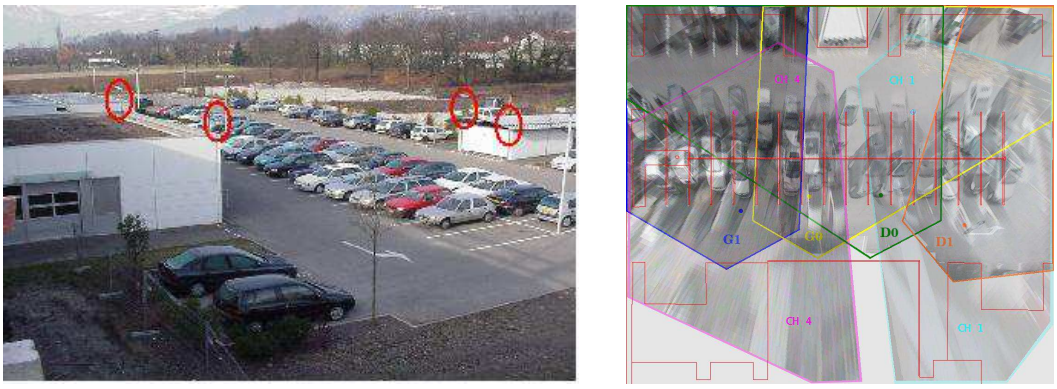
4.2 National project PUVAME

4.2.1 Project description

The national project PUVAME [1] [38] was created to generate solutions to avoid collisions between VRU [2] and Bus in urban traffic. The project started on october 2003 and ended in april 2006 and had 6 partners.

1. http ://emotion.inrialpes.fr/puvame/
2. Vulnerable Road Users

4.2.2 Experimental Platform



(a) (b)

FIGURE 4.1 – (a) Location of the cameras on the parking ; (b) Field-of-view of the cameras projected on the ground.

The experimental setup used to evaluate the PUVAME system is composed of 2 distinctive parts :

- the ParkView platform that is used to simulate an intersection or a bus stop. The ParkView platform is composed of a set of six off-board analog cameras, installed in a car-park setup such as their field-of-view partially overlap (see figure 4.1) ;
- and a cycab vehicule (figure 1.1) used to simulate a bus. It is connected to the ParkView platform by a wireless connection : we can send it informations about possible collisions and collect odometry data. In this project, sensors embedded on the cycab were not used.

4.2.3 Experimental Results

In the framework of this project, 2 kinds of work have been performed : the first one on sensor data fusion of offboard camera [37] and the second one on tracking of pedestrians [35, 33].

Sensor Data Fusion

In cooperation with David Raulo [37], we model the environment perceived by the set of offboard camera with an occupancy grid. This occupancy grid has a size of 150m $\times$ 50m and each cell has a size of 50cms. A partner of the project was in charge of delivering a detector of pedestrians in image. We use this information to build a sensor model using information provided by these detectors.

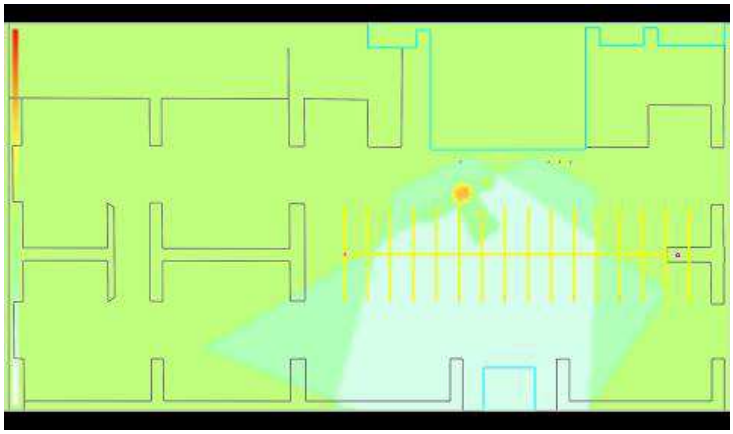


FIGURE 4.2 – The resulting probability that the cells are occupied after the inference process with two cameras.

Figure 4.2 shows the same pedestrian seen by two cameras. The red area corresponds to the most probable position of the pedestrian : this area is the result of the fusion of the two yellow areas given the two cameras. The 3 green areas around the pedestrian correspond to the fusion between the occluded area of one camera with the free area of the other one. The area seen as free by the two cameras has a very low probability of occupancy. The 4 areas seen as free by one camera and out of the field of view of the second camera have a low probability of occupancy.

Afterwards, a method for objects extraction from the grid has been implemented [32].

Tracking of pedestrians

To validate our work on tracking with adaptative IMM (see section 3.3), experiments on the ParkView platform have been carried out. In these experiments,

a pedestrian moving in the car park is detected by the set of offboard camera. The pedestrian describes in this way approximatively hundred trajectories to meet the needs of our experiment.

Using these trajectories, our adaptive method is used to compute estimates and the TPM of the IMM is updated for each ten trajectories.

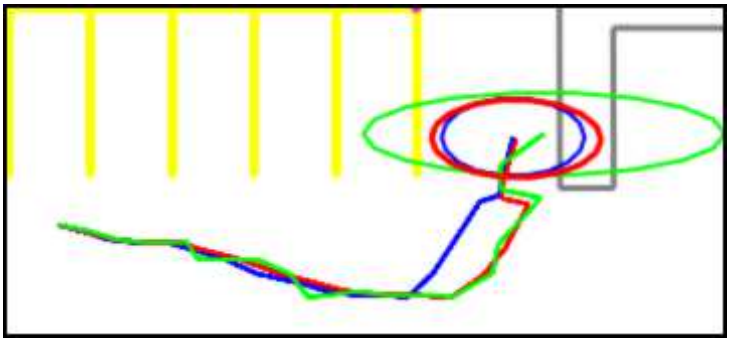


FIGURE 4.3 – Tracking result after 50 trajectories (5 online re-adaptation of the TPM)

To illustrate the effectiveness of our method, traces of tracking with and without adaptation of the TPM are showed in figure 4.3. In these figures, the green (lightest) line corresponds to the trajectory composed by observations (considered as the ground truth), the blue(darkest) line is the trajectory described by estimates computed without adaptation of the TPM and red line corresponds to the trajectory obtained with estimates computed using our method. The ellipses at the end of the trajectories give indications on the size of uncertainty on the final position and so the estimates' shape.

In figure 4.3, the pedestrian has achieved fifty random trajectories. Here, the tracking performed by our method (red trajectory) is significantly improved after five re-estimations while without adaptation, computed estimates are far from observations during pedestrian's motion changes.

Also, as adaptation is continuous using on-line data, even if pedestrian trajectories vary because of changes in car park configuration, for instance if cars exit the car park, the TPM is automatically readapted to fit this variation. Thus the computed estimations are always better than using an *a priori* TPM, or a learned TPM with a finite set of trajectories since our method is robust to pedestrian behavior changes.

4.3 European IP PReVENT-ProFusion

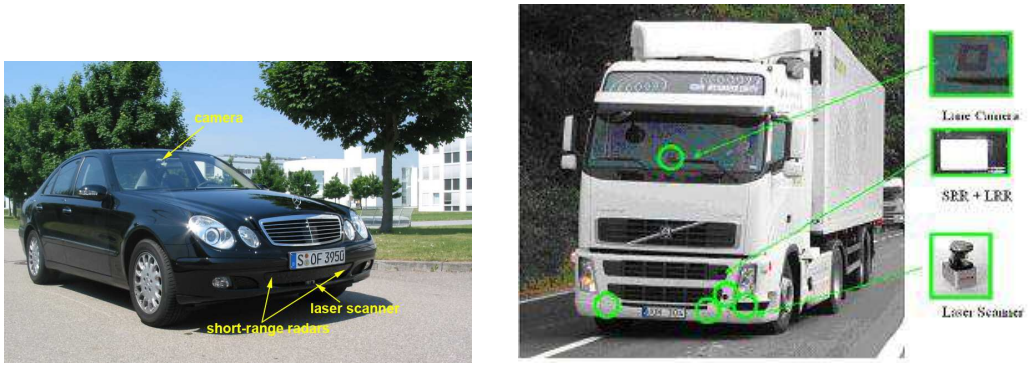
4.3.1 Project description

In the framework of the european IP PReVENT [3], we were involved in the project ProFusion [39] [31] which was in charge of designing and developping a generic architecture for Perception Solutions for automotive applications. We developped a solution based on occupancy grid and integrate this solution on two demonstrators : a Daimler demonstrator and Volvo Truck demonstrator.

The project started in february 2004 and ended in august 2008 and had 50 partners.

4.3.2 Demonstrators

Daimler demonstrator



(a) The Daimler demonstrator car (b) The Volvo Truck demonstrator

FIGURE 4.4 – The PReVENT demonstrators

The Daimler demonstrator car is equipped with a camera, two short range radar sensors and a laser scanner (Figure 4.4(a)). The radar sensor is with a maximum range of 30m and a field of view of 80°. The data of radar are processed and it delivers a list of moving objects. The maximum range of laser sensor is 80m with a field of view of 160°and a horizontal resolution of 1°. The laser data are not

3. http ://www.prevent-ip.org/

processed. In addition, vehicle odometry information such as velocity and yaw rate are provided by the vehicle sensors. Images from camera are for visualization purpose.

Volvo Truck demonstrator

The test vehicle platform is based on a Volvo FH12 420 Globetrotter truck (figure 4.4(b)). The main components of the perception system are (1) a laser scanner, mounted in the front left corner of the truck. This sensor has a field of view of 210˚and a range of 80 meters, (2) a lane camera and vision system, (3) a long range radar (LRR) system. This sensor has a field of view of 12˚and a range of 200 meters and (4) a short-range radar (SRR) system.

Moreover, data of each sensor are processed, and each sensor delivers a list of moving objects present in the environment.

4.3.3 Experimental Results

Daimler demonstrator

The SLAM and moving object detection contributions [29] detailed in section 2.2 were integrated on this demonstrator. The grid has a size 200m $\times$ 40m and each cell has a size of 20 cms. Confirmation of detection of moving objects by laser scanner is performed with the two short range radars. Regarding Tracking of Moving Objects, we integrated our contribution on adaptative IMM (section 3.3 associated with MHT algorithm [Rei79, CH96a] for Data Association[4]. A description of this work could be found in [7].

The detection and tracking results are shown in Figure 4.5. The images in the first row represent online maps and objects moving in the vicinity of the vehicle are detected and tracked. The current vehicle location is represented by blue box along with its trajectories after correction from the odometry. The red points are current laser measurements that are identified as belonging to dynamic objects. Green boxes indicate detected and tracked moving objects with corresponding tracks displayed in different colors. Information on velocities is displayed next to detected objects if available. The second row are images for visual references

4. Our contribution detailed in section 3.2 was not available at this time.

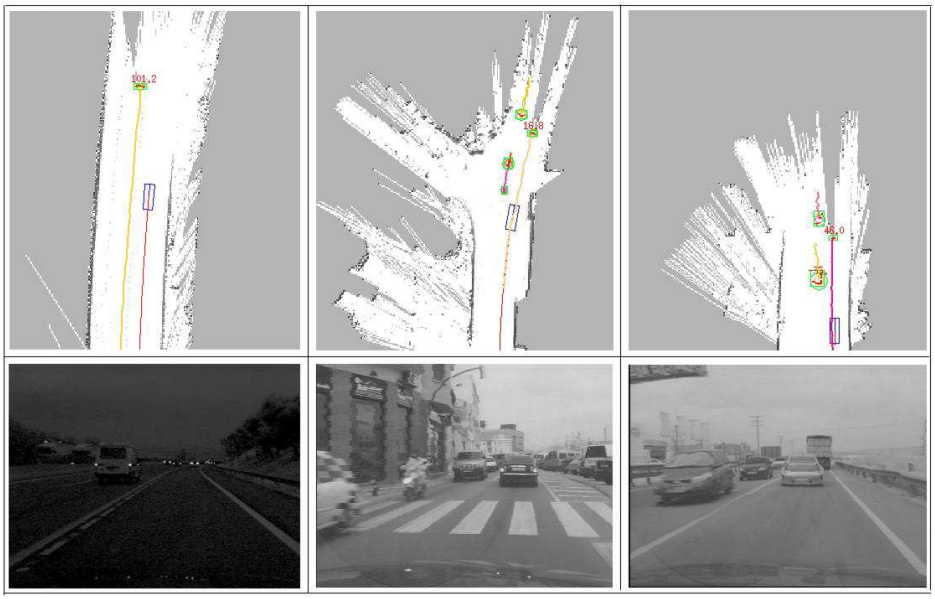


FIGURE 4.5 – Experimental results show that our algorithm can successfully perform both SLAM and DATMO in real time for different environments

to corresponding situations. More results and videos can be found at http://emotion.inrialpes.fr/~tdvu/videos/.

Moreover, our solution has been validated in complex crash and non-crash scenarios and compared with Daimler solution [8]. To conduct the experiments, we built up a comprehensive database that consists of short sequences of measurements recorded during predefined driving maneuvers. To measure the quality, we counted the false alarms that occurred in non-crash scenarios and the missed alarms in case a collision was not detected by the application. As a general result it can be stated that a reliable collision detection is achieved with both perception modules. Whereas Module of Daimler enables a lower false alarm rate, the crash detection rate of our module is very high (98.1%) in urban areas. A description of this work could be found in [8].

Volvo Truck demonstrator

In cooperation with Ruben Garcia [24], we model the environment perceived by the set of sensors with an occupancy grid. We built a sensor model for each

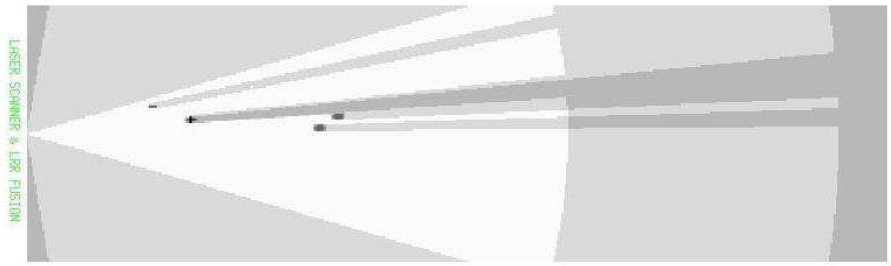


FIGURE 4.6 – The sensors' fusion between long range radar and laser.

sensor in a way similar to what we did in puvame [37]. We also used an adaptative IMM associated with MHT algorithm [Rei79, CH96a] for Data Association, similarly to what we did on the Daimler demonstrator.

The figure 4.6 illustrates the process of sensor fusion between long range radar (LRR) and laser. The ego vehicle is located on the left. The white zone corresponds to the fusion between the free zones of both fields of view of the sensors. The only object detected by both sensors has a high probability of occupancy and the area behind this object corresponds to occluded area. Other objects (only detected by one sensor) have lower probabilities of occupancy than the object detected by both sensors.

4.4 NAVLAB

4.4.1 Project description

In this part, we present how we use the public dataset NAVLAB [WDG+04] obtained on the CMU demonstrator (Figure 4.7) to test and validate our contributions on model-based tracking. This dataset was collected using a moving vehicle driven in real-life traffics.

4.4.2 CMU Demonstrator

A laserscanner is mounted on the moving vehicle. The maximum laser range of the scanner is 80m with the horizontal resolution of 0.5°. We only use laser data and odometry vehicle motion information such as translational and rotational

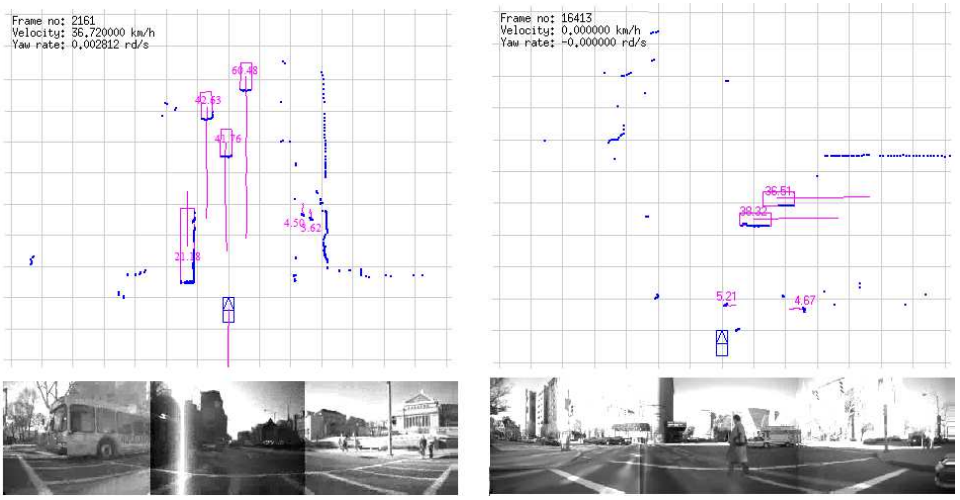
FIGURE 4.7 – Navlab testbed.

velocity (speed and yaw rate) are computed and provided by internal sensors. Images from camera are only for visualization purpose.

4.4.3 Experimental Results

On this demonstrator, we test our approach for SLAM with moving objects detection (section 2.2) and DATMO (section 3.2). The grid has a size of 25m $\times$ 20m and each cell has a size of 20cms. A complete description of this work could be found in [23].

Figure 4.8(a) shows an example of our detection and tracking algorithm in action. In the ego-vehicle's view, the detected moving objects and their trajectories are shown in pink color with current laser scan is in blue color. Moving objects in the situation include a bus moving in the opposite direction on the left, three cars moving ahead and two pedestrians walking on the left pavement. Figure 4.8(b) shows an example of our detection and tracking algorithm when an occlusion occurs. Even if only a part of the second car is detected by the laser, we are able to track this occluded car. With initial evaluations, the MCMC detection and tracking outperforms the detection and tracking using MHT in our previous work (see section 4.3) in terms of a higher detection rate and less false alarms.



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

(b) Example of tracking with occlusion.

FIGURE 4.8 – Moving object detection and tracking in action.

4.5 3D Mapping

4.5.1 Project description and Demonstrator

In 2008, we had a cooperation with an industrial partner to perform 3D Mapping using a laser scanner. We had a dataset of 6 Millions of laser data collected with a vehicle moving in an urban environment. The localization of the vehicle was provided. A camera was also mounted on the vehicle and laser data were synchronized with images. So, each laser data had 6 dimensions : x, y and z (for position) and RGB component for color.

The problem was to propose a model of this 6D dataset with high precision and significant reduction of size. We build a multiscale gaussian map of 700 meters $\times$ 200 meters $\times$ 50 meters. This map was composed of 3 scales : cells have a size of 20 centimeters, 3.2 meters and 12.8 meters.

4.5.2 Experimental Results

With this data set, we have experimented the use of the color as an extra clustering dimension. In this case, the points are gathered into a same cluster if they are close in terms of location (ie, they produce the smallest possible

FIGURE 4.9 – 3D representation of 3D laser data and color

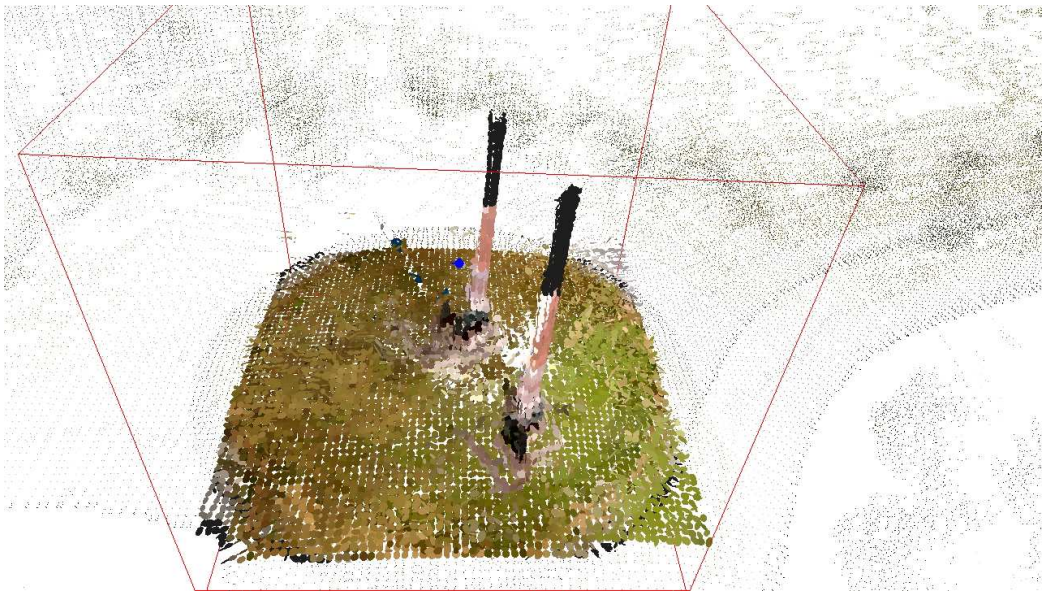


FIGURE 4.10 – focus of the 3D representation of the roundabout

ellipsoid in the 3D space) and in terms of color (they make the smallest ellipsoid in the RGB color space). With that clustering, we were able to obtained a reduce representation of less than 0.1% of the original data but nonetheless accurate (see figure 4.9). Moreover, One can see in figure 4.10 that a significative distinction is made between the object in the environment based upon the color. In particular vegetation is well separated from artificial structure.

4.6 European Project Intersafe2

4.6.1 Project description

The INTERSAFE-2 european project [5] aims to develop and demonstrate a Cooperative Intersection Safety System (CISS) that is able to significantly reduce injury and fatal accidents at intersections. In this project, we are in charge of developping the perception module for the Volkswagen Demonstrator.

The project started in june 2008 and will end in june 2011 and has 11 partners.

4.6.2 Demonstrator

Figure 4.11 illustrates the chosen sensor set and coverage area of the Volkswagen demonstrator car. The sensor set-up includes sensors which are already in serial cars available, namely front ACC radar and rear-looking radar for lane change support. These sensors are accompanied by a stereo camera system to the front with high field of view of about 60˚. A scanning laser with a field of view of about 160˚and dedicated radar sensors directed to +90˚and -90˚respectively are foreseen for measuring the objects coming from the side. These sensors are able to measure position, velocity and some geometrical parameters of the relevant objects at intersections.

4.6.3 Experimental Results

On this demonstrator, we test our approach for SLAM with moving objects detection (section 2.2) and and improvement on our contribution on DATMO (section 3.2) is under investigation. The grid has a size of 25m $\times$ 20m and each cell

5. http ://www.intersafe-2.eu/public/

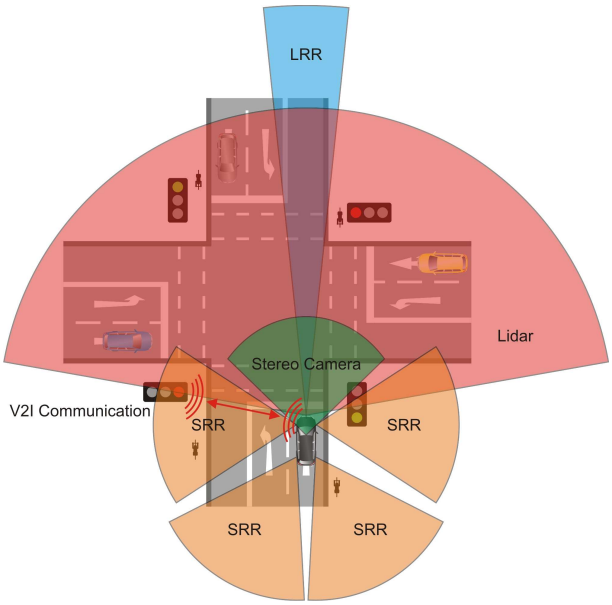


FIGURE 4.11 – Sensors installed on the demonstrator vehicle

has a size of 20cms. A complete description of this work could be found in [22]. Moreover, fusion between stereovision and laserscanner is under investigation (see section 5.2.3).

In Figure 4.12, the first column shows a left turn scenario where the demonstrator car is turning left and a cyclist is going to right in front of the vehicle. This cyclist is detected and tracked. In the second column we have a scenario where a moving vehicle is coming from opposite direction and demonstrator is crossing a vehicle on its right. In both of the cases, precise trajectories of the demonstrator are achieved and local maps around the vehicle are constructed consistently.

4.7 Conclusion

In this chapter, we give an overview of the way our theoretical contributions have been integrated on some industrial experimental platforms. We also summarize the results obtained.

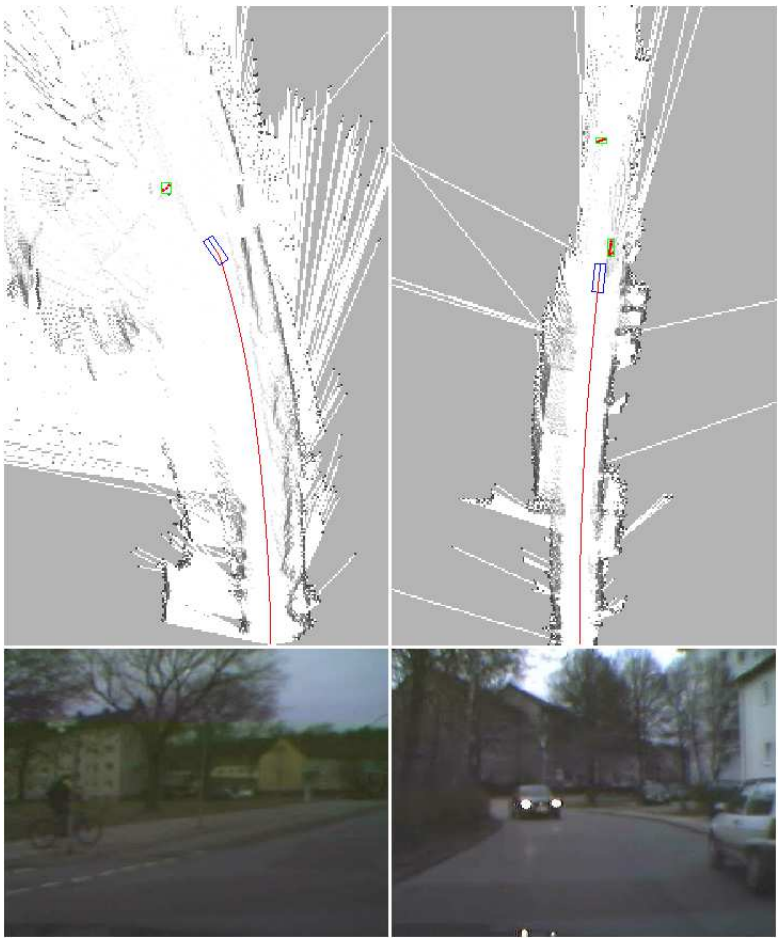


FIGURE 4.12 – Experimental results show that the algorithm can successfully perform both SLAM and moving objects detection in real time for different environments.

Chapitre 5

Conclusion and Perspectives

5.1 Conclusion

In this document, we have studied core tasks of the vehicle perception problem including Simultaneous Localization And 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 our main contributions in what follows.

In Chapter 2, we described a grid-based algorithm to solve SLAM with detection of moving objects. In the literature, SLAM algorithms have reached 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 in the framework of european projects PReVENT (section 4.3) and Intersafe2 (section 4.6) and with the NAVLAB datasets (section 4.4) 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.

Occupancy grids are considered as the mainstream tool to map dynamic outdoor environment while filtering out moving obstacles. However, it is well known that occupancy grids suffer from a precision problem. This problem is often discarded. For this reason, we propose to represent data in a given cell by one or more gaussians. An other important problem with grids is the size of the grid when representing large 2D or 3D environments. Moreover, most cells are empty or unknown. So, we propose to represent only cells that are considered as occupied using a sparse grid. Finally, we introduce a multiscale representation. Results on simulated 3D data (section 2.3) and on real 3D+color data in cooperation with an industrial partner (section 4.5) shows the interest of the method.

Chapter 3 follows the results in Chapter 2 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 2, moving objects are identified irrespective of their 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 approaches 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 the 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 2 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 from NAVLAB datasets (introduced in section 4.4) and evaluations showed promising results.

Regarding tracking, we started by proposing a method to automatically adapt the TPM [1] associated with a set of motion models. In the litterature, to use IMM [2], it is necessary to define a set of motion models which depends on the application and should model all the possible motions of an object. Such a definition could be difficult or impossible if the set of possible motions is varied. Secondly, we explained how to use this automatic adaptation as a help to define the set of motion models. Actually, analyzing the adapted TPM enables us with some rules to modify or validate the set of initial motion models. Finally, we extended this contribution by the definition of a method of classification enabling us to classify the differents behaviors that could appear in a given environment and secondly to reuse this classification to define a very specific TPM for each class of behavior of moving objects.

These contributions have been used in the framework of the european project PReVENT (section 4.3) and the national project PUVAME (section 4.2) with very good results.

To summarize, the thesis is aimed at developing an efficient perception system for intelligent 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. The obtained results will open a wide range of potential applications as well as serve as a basis for pursuing the dream of building autonomous intelligent vehicles.

1. Transition Probability Matrix
2. Interacting Multiple Models

5.2 Perspectives

In this section, we give some perspectives of the work presented in this document. Our main perspective in the futur is to design and develop generic modules and solutions for perception tasks and to apply them to different domains of applications. The 3 first perspectives are short term perspectives related to SLAM and DATMO done in the framework of current PhD Thesis [1, 2, 3]. These PhD Thesis are done in the framework of the european project Interactive (see section A.7 and http ://www.interactive-ip.eu) which goal is to build generic modules for perception for automotive applications. These PhD Thesis will bring new tools for our main perspective.

We conclude with 2 long term perspectives that constitute some extensions of researches described in this document. The first perspective is related to trajectory estimation and prediction of moving objects. The goal of this perspective is to have a "higher level" understanding of the environment to make the link with the navigation task and the decision layer of a typical architecture of an intelligent vehicle. The second one is an extension of this work in the frame of ambiant intelligence to extend our research on perception.

5.2.1 Environment Representation and Interpretation of Environment

Representation of environment constitutes a key point to solve the SLAM problem. In the PhD Thesis of Manuel Yguel [4], we proposed an interesting solution to map large scale 2D and 3D environments. This representation has the main advantages of being very precise and multiscale. But this representation has several limits : first of all, updating of this representation is not real-time, so we are not able to use it in applications where real time constraints are important. Secondly, at the moment, no localization is provided with this representation. Finally, we have not defined how to manage moving objects with this representation.

On the other side, typical representation of roads environments designed by Navteq [3], for instance, are a kind of topological maps where nodes represent junctions between several roads and arcs are represented by polylines that connect

3. http ://www.navteq.com

these junctions. This simple representation is very efficient for coarse localization with GPS and motion measurements. But precise localization is impossible with this kind of representation and sensors.

We are working on combination of these two representations. The topological map could be used to provide a coarse localization of the vehicle. On fine scale, we propose to use a gaussian local map and laser scanner to provide a precise localization (similarly to section 2.2).

An other important drawback is that in our current environment representation, there is any semantical information about the environment. In a perception module, some high level information about environment are necessary. At the moment, our perception module only provides a very limited understanding of the environment : only informations about detection and tracking of moving objects are available. We have no information about static part of the environment. For instance, we would like to extract road border from a grid representation to use them to improve the quality of localization at coarser level : for instance, provide lateral localization as road lane positionning. The same kind of information could be used to determine the border between free and occupied space in local environment. This border is crucial to determine the free space in order to provide information about where the vehicle can go in a short term.

This work is currently under investigation, in the framework of the PhD Thesis of Asma Azeem [2], in cooperation with Navteq in the framework of the european project Interactive.

5.2.2 Frontal object perception

The objective of this module is to detect and track every relevant obstacle in the front area of the ego vehicle including stationary and moving objects and provide information about these objects (ex : position and dynamic parameters, etc.). Sensor data fusion and advanced filtering techniques should be taken into account in order to obtain a more reliable perception result and provide additional information not directly observed from sensor (ex : estimation of object velocity from laser scanner data).

In this work, we plan to extend the PhD Thesis of Trung-Dung Vu [5] on DATMO on several aspects :

- The model-based tracking has proved to be very robust but at the moment due to its high computational demand it is not real time. The idea is to put more constraints on the hypothesis space according to our knowledge of related applications ;
- Moreover, at the moment, any information is provided on static objects. For instance, we would like to know if there are, in the environment, some static objects that could move in the future like a car that is parked ;
- As this module is mainly concerned with detection and tracking of static and dynamic objects, the map of the environment will only be used to localize ego vehicle. To improve the precision and have more compact representation, our idea is to perform localization and detection of moving objects on a sliding window of observations.
- As described in detail in section 5.2.3, we plan to use fusion between vision, radar and laser to improve the robustness of objects detection and tracking. Basically, the idea is to check detection by laser with radar to improve robustness and to use vision to quickly obtain class of objects for the model-based tracking.

This work is currently under investigation, in the framework of the PhD Thesis of Omar Chavez [1], in cooperation with Daimler (http ://www.daimler.com/) and TRW (http ://www.trw.com) (for fusion between radar and vision) in the framework of the european project Interactive.

5.2.3 Sensor Data Fusion

Sensor Data Fusion plays an important role in the conception and realization of a perception module. Actually, there is no "perfect" sensor and each sensor has its own physical properties. To solve this problem, an intelligent vehicle is usually equipped with several kind of sensors. Data provided by these sensors are fusioned to improve the quality of perception. Most of works described in this document only use laser scanner.

In the framework of european project PReVENT-ProFusion, some studies have been done on fusion on Daimler [7] and Vtech [24] demonstrators : we performed fusion to improve robustness on estimation of moving objects attributes (position, speed...). Fusion was performed after tracking. In this case, fusion techniques were designed independently of the sensor processing. The main

advantage is the genericity of sensor data fusion techniques designed but the drawback is that some informations provided by each sensor at earlier stage have been suppressed during sensor processing.

To avoid this problem, we are developping techniques to perform fusion after detection phase using complementary informations provided by vision and laser scanner. Actually, laser scanner processing provides very good rate of detections of moving objects, but determination of class of objects is sometimes difficult due to ambiguities in laser data. This is especially the case between cars and buses/trucks which have a very similar box model when we only observe one corner of the box model. In this case, we need to have informations about the length of one side to decide between a car and a truck/bus. On the contrary, vision processing provides lower rate of detections of moving objects, but determination of class of moving objects is easier. So, we plan to integrate classification provided by vision in the process of model-based tracking with laserscanner. Combining data of vision and laser, we should quickly know the class of a given moving object. This information will improve the quality of DATMO as we will be able to use a box model as soon as a moving object will enter in the field of view of the laser scanner.

This work is currently under investigation, in the framework of the PhD Thesis of Qadeer Baig [3], in cooperation with Volkswagen and Technical University of Cluj-Napoca for stereovision processing, in the framework of the european project Intersafe2 (see section A.7 and http ://www.intersafe-2.eu).

5.2.4 Trajectory estimation and prediction of moving objects

We are able to estimate the position, speed and class of moving objects. But these information are not enough to understand the futur behavior of moving objects present in the environment. We are interested in knowing their futur trajectories. In the past, we participated to the PhD Thesis of Dizan Vasquez [Vas, 10, 40] related to this topic. This Thesis was built upon a family of approaches which are based on the idea that, for a given environment, moving objects follow typical motion patterns which may be observed consistently. Most of these approaches operate in a learn then predict fashion : first, they learn motion patterns from sensor data, then, they predict further motion on the basis of learned patterns. The basis of this work is on Hidden Markov Model. However, unlike

these approaches, which rely on clustering algorithms to identify motion patterns, this work has set to integrate the whole learning process into an HMM parameter and structure learning algorithm. The approach is based on the idea that a single HMM may represent multiple motion patterns by defining an appropriate graph structure.

In the futur, we plan to extend this past work on 2 aspects :

1. taking into account information about environment like road border for instance. This information will play an important role in determining the position that are reachable in the environment ;
2. taking into account information about class of moving objects to improve the quality of prediction. This information is fundamental as futur trajectory is different for different classes of moving objects.

If we provide robust prediction of future trajectories, these informations could be used by the decision layer of a typical intelligent vehicle architecture.

5.2.5 Ambiant Intelligence and Perception

Technological evolution enables to build now computers and components of very small size : for instance, sensors and actuators are present everywhere and can communicate between themself and with different networks. This evolution opens a lot of applications in our everyday life : prevention (fire, accidents) assistance (guidance, distant control)... This evolution is general defined as Ambiant Intelligence.

As sensors are present everywhere, perception tasks play an important role in Ambiant Intelligence. Moreover, Ambiant Intelligence opens new challenges for Perception : perception will be distributed and performed by heterogeneous sensors. To solve these challenges, we have to deal with large scale sensor data fusion and mapping of large environments. Solving these challenges is a natural extension of the perspectives described in section 5.2.1 and 5.2.3.

Other important aspects of Ambiant Intelligence are the Detection and Tracking of people and the recognition and monitoring of their activities. Our contribution on DATMO (section 3.2) could be adapted to track people : we think that model-based tracking has an important potential for multi objects tracking. Moreover, our contribution on behavior based classification of moving objects

(section 3.3.5) could bring new solutions to the problem of recognition of human activities.

The last topic related to our researches and Ambiant Intelligence is what we usually call the ubiquitous robotics. The goal is to integrate mobile robots inside environments composed of sensors and actuators networks. In this case, mobile robots could be considered as mobile sensors units that will cooperate with sensor networks to improve the quality of perception of the environment. For instance, a mobile robot could move to perceive a part of the environment that is not perceived by any other sensor. So, data acquired by sensors present (microphones, cameras...) in the environment are fused with data acquired from embedded sensors in order to improve the quality of service.

Annexe A

Publications

PhD Students

Current PhD Students

[1] Omar Chavez. *Frontal moving objects detection and tracking*. Phd thesis, Grenoble University, UJF, Grenoble, France. since sep. 2010.

[2] Asma Azeem. *Environment representation and interpretation*. Phd thesis, Grenoble University, UJF, Grenoble, France. since sep. 2009.

[3] Qadeer Baig. *Detection Level Fusion between laserscanner and stereovision*. Phd thesis, Grenoble University, UJF. since sep. 2008.

Past PhD Students

[4] Manuel Yguel. *Variational Algorithms for On-line Updating of Dense Maps in Sparse Multi-scale Representations. Application to robot navigation in 3D environments.* Phd thesis, Grenoble University, INPG, Grenoble, France, December 2009.

[5] Trung-Dung Vu. *Vehicle Perception : Localization, Mapping with Detection, Classification and Tracking of Moving Objects*. Phd thesis, Grenoble University, INPG, Grenoble, France, September 2009.

[6] Julien Burlet. *Suivi Multi-Objets Adaptatif : Application à la Classification de Comportements d'objets mobiles*. Phd thesis, Grenoble University, INPG, Grenoble, France, December 2007.

Scientific Journal Articles

[7] TD. Vu, J. Burlet, and O. Aycard. Grid-based localization and local mapping with moving objects detection and tracking. *International Journal on Information Fusion,* Special issue on Intelligent Transport System, *Elsevier*, July 2010. *[pdf]*. Impact factor : 2.21.

[8] S. Pietzsch, TD. Vu, J. Burlet, O. Aycard, T. Hackbarth, N. Appenrodt, J. Dickmann, and B. Radig. Results of a precrash application based on laser scanner and short range radars. *IEEE Transactions on Intelligent Transport Systems*, 10(4) :584–593, 2009. *[pdf]*.

[9] J. Burlet, T. Fraichard, and O. Aycard. Robust navigation using markov models. *International Journal of Advanced Robotic Systems*, 2008. *[pdf]*.

[10] D. Vasquez, T. Fraichard, O. Aycard, and C. Laugier. Intentional motion on-line learning and prediction. *Machine Vision and Applications*, 2008. *[pdf]*.

[11] C. Laugier, D. Vasquez, M. Yguel, T. Fraichard, and O. Aycard. Geometric and bayesian model for safe navigation in dynamic environments. *Intelligent Service Robots*, 2007. *[pdf]*.

[12] M. Yguel, O. Aycard, and C. Laugier. Efficient gpu-based construction of occupancy grids using several laser range finders. *International Journal of Autonomous Vehicles*, 2007. *[pdf]*.

[13] O. Aycard, JF. Mari, and R. Washington. Learning to automatically detect features for mobile robots using second-order hidden markov models. *International Journal of Advanced Robotic Systems*, 2004. *[pdf]*.

Book Chapters

[14] C. Laugier, S. Petti, A. D. Vasquez, M. Yguel, Th. Fraichard, and O. Aycard. Steps Towards Safe Navigation in Open and Dynamic Environments. In C. Laugier and R. Chatila, editors, *Autonomous Navigation in Dynamic Environments*, volume 35 of *Springer Tracts in Advanced Robotics Series*. Springer, 2007. *[pdf]*.

[15] A. D. Vasquez, Th. Fraichard, O. Aycard, and C. Laugier. Intentional Motion Online Learning and Prediction. In P. Corke and S. Sukkarieh, editors, *Field and Service Robotics*, volume 25 of *Springer Tracts in Advanced Robotics Series*. Springer, 2006. *[pdf]*.

[16] O. Aycard, JF. Mari, and R. Schott. *Application of Markov models in robotics. In Probabilistic and Statistical Methods in Computer Science.*, pages 177–205. Kluwer Academic Publishers, January 2001.

Peer-Reviewed Conference Articles

[17] Julien Burlet and Olivier Aycard. Interacting multiple models based classification of moving objects. In *IEEE International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2010.

[18] Qadeer Baig and Olivier Aycard. Low level fusion of laser and mono camera for object detection and classification. In *IEEE International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2010.

[19] Azeem Asma and Aycard Olivier. Multiple pedestrian tracking using viterbi data association. In *IEEE International Conference on Intelligent Vehicles (IV)*, 2010.

[20] Aycard Olivier, Vu Trung-Dung, and Baig Qadeer. An occupancy grid based architecture for advanced driver assistant system. In *Advanced Microsystems for Automotive Applications (AMAA)*, 2010.

[21] Yguel Manuel, Vasquez Dizan, Aycard Olivier, Siegwart Roland, and Laugier Christian. Error-driven refinement of multi-scale gaussian maps application to 3-d multi-scale map building, compression and merging. In *International Symposium on Robotics Research (ISRR)*, 2009.

[22] Qadeer Baig, Trung-Dung Vu, and Olivier Aycard. Online localization and mapping with moving objects detection in dynamic outdoor environments. In *IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, 2009.

[23] Trung-Dung Vu and Olivier Aycard. Lased-based detection and tracking moving object using data-driven markov chain monte carlo. In *IEEE*

International Conference on Robotics and Automation (ICRA), Kobe, Japan, May 2009. *[pdf]*.

[24] Ruben Garcia, Olivier Aycard, Trung-Dung Vu, and Malte Ahrholdt. High level sensor data fusion for automotive applications using occupancy grids. In *IEEE International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2008. *[pdf]*.

[25] S. Pietzsch, O. Aycard, J. Burlet, TD. Vu, T. Hackbarth, N. Appenrodt, J. Dickmann, and B. Radig. Results of a precrash application based on laserscanner and short range radars. In *IEEE International Conference on Intelligent Vehicles (IV)*, 2008. *[pdf]*.

[26] TD. Vu, J. Burlet, and O. Aycard. Grid-based localization and online mapping with moving objects detection and tracking : new results. In *IEEE International Conference on Intelligent Vehicles (IV)*, 2008. *[pdf]*.

[27] TD. Vu, J. Burlet, and O. Aycard. Mapping of environment, detection and tracking of moving objects using occupancy grids. In *Workshop on Intelligent Transport*, 2008. *[pdf]*.

[28] M. Yguel, C. Tay Meng Keat, C. Braillon, C. Laugier, and O. Aycard. Dense mapping for telemetric sensors : efficient algorithms and sparse representation. In *Robotic Science and Systems (RSS)*, 2007. *[pdf]*.

[29] TD. Vu, O. Aycard, and N. Appenrodt. Online localization and mapping with moving objects tracking in dynamic outdoor environments. In *IEEE International Conference on Intelligent Vehicles (IV)*, 2007. *[pdf]*.

[30] N. Floudas, A. Polychronopoulos, O. Aycard, J. Burlet, and M. Ahrholdt. High level sensor data fusion approaches for object recognition in road environment. In *IEEE International Conference on Intelligent Vehicles (IV)*, 2007. *[pdf]*.

[31] SB. Park, F. Tango, O. Aycard, A. Polychronopoulos, U. Scheunert, and T. Tatschke. Profusion2 - sensor data fusion for multiple active safety applications. In *Intelligent Transport System World Congress*, 2006. *[pdf]*.

[32] O. Aycard, A. Spalanzani, J. Burlet, C. Fulgenzi, TD. Vu, D. Raulo, and M. Yguel. Pedestrians tracking using offboard cameras. In *IEEE International Conference on Intelligent Robot and Systems (IROS)*, 2006. *[pdf]*.

[33] J. Burlet, O. Aycard, A. Spalanzani, and C. Laugier. Adaptive interacting multiple models applied on pedestrian tracking in car parks. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2006. *[pdf]*.

[34] M. Yguel, O. Aycard, and C. Laugier. Efficient gpu-based construction of occupancy grids using several laser range finders. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2006. *[pdf]*.

[35] J. Burlet, O. Aycard, A. Spalanzani, and C. Laugier. Pedestrian tracking in car parks : a adaptive interacting multiple models based filtering method. In *IEEE International Conference on Intelligent Transport Systems (ITSC)*, 2006. *[pdf]*.

[36] O. Aycard, A. Spalanzani, J. Burlet, C. Fulgenzi, TD. Vu, D. Raulo, and M. Yguel. Grid based fusion and tracking. In *IEEE International Conference on Intelligent Transport Systems (ITSC)*, 2006. *[pdf]*.

[37] M. Yguel, O. Aycard, D. Raulo, and C. Laugier. Grid based fusion of offboard camera. In *IEEE International Conference on Intelligent Vehicles (IV)*, 2006. *[pdf]*.

[38] O. Aycard. Puvame - new french approach for vulnerable road users safety. In *IEEE International Conference on Intelligent Vehicles (IV)*, 2006. *[pdf]*.

[39] T. Tatschke, SB. Park, A. Amditis, A. Polychronopoulos, U. Scheunert, and O. Aycard. Profusion2 - towards a modular, robust and reliable fusion architecture for automotive environment perception. In *Advanced Microsystems for Automotive Applications (AMAA)*, 2006. *[pdf]*.

[40] D. Vasquez, T. Fraichard, O. Aycard, and C. Laugier. On-line intentional motion learning and prediction. In *International Conference on Field and Service Robotics (FSR)*, 2005. *[pdf]*.

[41] M. Yguel, O. Aycard, and C. Laugier. Wavelet occupancy grids : a method for compact map building. In *International Conference on Field and Service Robotics (FSR)*, 2005. *[pdf]*.

[42] J. Burlet, T. Fraichard, and O. Aycard. Robust navigation using markov models. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2005. *[pdf]*.

[43] N. Mansard, O. Aycard, and C. Koike. Hierarchy of behaviours. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2005. *[pdf]*.

[44] C. Laugier, S. Petti, D. Vasquez, M. Yguel, T. Fraichard, and O. Aycard. Steps towards safe navigation in open and dynamic environments. In *IEEE International Conference on Robotics and Automation (ICRA). Workshop on Autonomous Navigation in Dynamic Environments*, 2005. *[pdf]*.

[45] J. Burlet, O. Aycard, and T. Fraichard. Robust motion planning using markov decision processes and quadtree decomposition. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2004. *[pdf]*.

[46] O. Aycard, JF. Mari, and R. Washington. Learning to automatically detect features for mobile robots using second-order hidden markov models. In *IJCAI-Workshop on Reasoning with Uncertainty in Robotics*, 2003. *[pdf]*.

[47] A. Tapus and O. Aycard. Searching a target with a mobile robot. In *International Conference on Control Systems and Computer Science*, 2003. *[pdf]*.

[48] O. Aycard and R. Washington. State identification for planetary rovers : Learning and recognition. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2000. *[pdf]*.

[49] O. Aycard, JF. Mari, and F. Charpillet. Second order hidden markov models for places recognition : New results. In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 1998. *[pdf]*.

[50] O. Aycard, P.Laroche, and F. Charpillet. Mobile robot localization in dynamic environment using place recognition. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1998. *[pdf]*.

[51] P.Morignot, O. Aycard, and F. Charpillet. A pair of heterogeneous agents in a unique vehicle for object motion. In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 508–513, 1997. *[pdf]*.

[52] O. Aycard, F. Charpillet, D. Fohr, and JF. Mari. Place learning and recognition using hidden markov models. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1741–1746, 1997. *[pdf]*.

[53] O. Aycard, F. Charpillet, and JP. Haton. A new approach to design fuzzy controllers for mobile robots navigation. In *IEEE International Symposium*

on Computational Intelligence in Robotics and Automation (CIRA), pages 68–73, 1997. *[pdf]*.

Research Reports

[54] J. Burlet, TD. Vu, and O. Aycard. Grid-based localization and online mapping with moving objects detection and tracking. Technical Report 167687, INRIA-UJF, 2007. *[pdf]*.

Annexe B

Bibliography

[AM79] B.D. Anderson and J.B. Moore. *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, New Jersey, 1979.

[AMGC02] 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*, 50(2), 2002.

[BB95] M. Busch and S. Blackman. Evaluation of imm filtering for an air defence system application. In *Proceedings SPIE Signal Data Process. Small targets*, volume SPIE 2561, pages 435–447, 1995.

[Bla86] S. Blackman. *Multiple Target Tracking with Radar Applications*. Artech House : Dedham, 1986.

[Bla04] S. S. Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 19(1) :5–18, Jan 2004.

[BP99] Samuel Blackman and Robert Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House, Norwood, MA, 1999.

[BP00a] S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House, 2000.

[BS03] Peter Biber and Wolfgang Straßer. The normal distributions transform : A new approach to laser scan matching. In *IEEE/RJS International Conference on Intelligent Robots and Systems*, 2003.

[BSF88] Y. Bar-Shalom and T.E. Fortman. *Tracking and Data Association*. Academic Press, 1988.

[BW92] W. D. Blair and G. A. Watson. IMM algorithm and aperiodic data. In *Proc. SPIE Vol. 1697, p. 83-91, Acquisition, Tracking, and Pointing VI, Michael K. Masten ; Larry A. Stockum ; Eds.*, pages 83–91, November 1992.

[CG91] G. Celeux and G. Govaert. Clustering criteria for discrete data and latent class models. *Journal of Classification*, 8(157-176), 1991.

[CH96a] Ingemar J. Cox and Sunita L. Hingorani. An efficient implementation and evaluation of reid's multiple hypothesis tracking algorithm for visual tracking. *Pattern Recognition, 1994. Vol. 1*, 1, 1996.

[Chr02] Henrik I. Christensen. *Lecture Notes : SLAM Summer School*. 2002.

[DBFT99] Frank Dellaert, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Using the condensation algorithm for robust, vision-based mobile robot localization. *CVPR*, June 1999.

[EF97] 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.

[Elf89a] A. Elfes. *Occupancy grids : a probabilistic framework for robot percpetion and navigation*. PhD thesis, Carnegie Mellon University, 1989.

[FBDT99] 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.

[FBT99] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11, 1999.

[For73] G. David Forney. The viterbi algorithm. *Proceedings of The IEEE*, 61(3) :268–278, 1973.

[FP02] David A. Forsyth and Jean Ponce. *Computer Vision : A Modern Approach*. Prentice Hall, August 2002.

[FVFH90] J. Foley, A. VanDam, S. Feiner, and J. Hughes. *Computer Graphics : Principles and Practice.* Addison-Wesley, 2nd. ed. edition, 1990.

[GH00] Zoubin Ghahramani and Geoffrey E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4) :831–864, 2000.

[GSB05] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. pages 2443–2448, 2005.

[HSB03] D. Hähnel, D. Schulz, and W. Burgard. Mobile robot mapping in populated environments. *Advanced Robotics*, 17(7) :579–598, 2003.

[JL04] Vesselin P. Jilkov and X. Rong Li. Online bayesian estimation of transiton probabilities for markovian jump systems. In *IEEE transaction on signal processing*, volume vol 52, No 6, pages 1620–1630, 2004.

[JLA03] Vesselin P. Jilkov, X. Rong Li, and Donka S. Angelova. Estimation of markovian jump systems with unknown transition probabilities through bayesian sampling. In *NMA '02 : Revised Papers from the 5th International Conference on Numerical Methods and Applications*, pages 307–315, London, UK, 2003. Springer-Verlag.

[Kal60] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 35, Mars 1960.

[LDW91] 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.

[Llo82] S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2) :129–137, Mar 1982.

[LM97a] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 1997.

[LM97b] Feng Lu and Evangelos Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, 18 :249–275, 1997.

[MABSD98] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan. Interacting multiple model methods in target tracking : a survey. *Aerospace and Electronic Systems, IEEE Transactions on*, 34(1) :103–123, 1998.

[MBB+08] 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.

[MLT00] Gérard Medioni, Mi-Suen Lee, and Chi-Keung Tang. *A Computational Framework for Segmentation and Grouping*. Elsevier Science Inc., New York, NY, USA, 2000.

[MP98] O.Drummond M.Miller and A. Perrella. Multiple-Model Filters for Boost-to-Coast Transition of Theater Ballistic Missiles. In *Proceedings SPIE Signal Data Process. Small targets*, volume SPIE 2561, pages 355–376, 1998.

[MTKW02] 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.

[Mur99] Kevin P. Murphy. Bayesian map learning in dynamic environments. *Neural Information Processing Systems (NIPS)*, pages 1015–1021, 1999.

[Ols00] Clark F. Olson. Probabilistic self-localization for mobile robots. *IEEE Transactions on Robotics and Automation*, 16 :55–66, 2000.

[ORS04] 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.

[OUV97] G. Oriolo, G. Ulivi, and M. Vendittelli. Fuzzy maps : a new tool for mobile robot perception and planning. *J. of Robotic Systems*, 14(3) :179–197, 1997.

[PNDW98] D. Pagac, E.M. Nebot, and H. Durrant-Whyte. An evidential approach to map-building for autonomous vehicles. *IEEE Transactions on Robotics and Automation*, 14, 1998.

[PR98] D. K. Pai and L.-M. Reissell. Multiresolution rough terrain motion planning. In *IEEE Transactions on Robotics and Automation*, volume 1, pages 19–33, February 1998.

[PRB03] 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.

[PT08] 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.

[RB05] Nora Ripperda and Claus Brenner. Marker-free registration of terrestrial laser scans using the normal distribution transform. Technical report, Institute of Cartography and Geoinformatics, University of Hannover, Germany, 2005.

[Rei79] Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24, 1979.

[RFDW07] 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.

[RLP03] X. Rong Li and Vesselin P.Jilkov. A survey of maneuvering target tracking-part v : Multiple-model methods. *IEEE Transactions on Aerospace and Electronic Systems*, 2003.

[SBFC01] 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.

[SKF72] Y. Sawaragi, T. Katayama, and S. Fujishige. Sequential state estimation with interrupted observation. *IEEE Trans. Automat. Contr.*, 21(1) :56–71, aug 1972.

[SN05] 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.

[TBF00] 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.

[TBF05] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, September 2005.

[Tho] Lemaire Thomas. *Simultaneous Localisation And Mapping with Monocular Vision*. Phd thesis, Toulouse University, Grenoble, France. dec. 2006.

[Thr02] Sebastian Thrun. Robotic mapping : A survey. *Exploring Artificial Intelligence in the New Millenium*, 2002.

[Tie96] Luke Tierney. Markov chain concepts related to sampling algorithms. *Markov Chain Monte Carlo in Practice*, pages 59–74, 1996.

[Tug82b] J. K. Tugnait82. Adaptive estimation and identification for discrete systems with markov jump parameters. *IEEE Trans. Automat. Contr.*, 27(1) :1054–1065, oct 1982.

[Vas] Dizan Vasquez. *Incremental Learning for Motion Prediction of Pedestrians and Vehicles*. Phd thesis, Grenoble University, INPG, Grenoble, France. feb. 2007.

[VJ01] Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2001.

[Wan04] Chieh-Chih Wang. *Simultaneous Localization, Mapping and Moving Object Tracking*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 2004.

[WDG+04] 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.

[WT02] 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.

[WTT03] 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.

[YCMW06] 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.

[YSSB98] Alex Yahja, Anthony (Tony) Stentz, Sanjiv Singh, and Barry Brummit. Framed-quadtree path planning for mobile robots operating in sparse environments. In *In Proceedings, IEEE Conference on Robotics and Automation, (ICRA)*, Leuven, Belgium, May 1998.

[ZNW08] Tao Zhao, Ramakant Nevatia, and Bo Wu. Segmentation and tracking of multiple humans in crowded environments. *PAMI*, 30(7) :1198–1211, 2008.

[ZZT00] 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.