Steps towards safe navigation in open and dynamic environments

Christian Laugier, Stephane Petti, Dizan Alejandro Vasquez Govea, Manuel Yguel, Thierry Fraichard, Olivier Aycard

Steps Towards Safe Navigation in Open and Dynamic Environments

C. Laugier, S. Petti, D. Vasquez, M. Yguel, Th. Fraichard & O. Aycard
INRIA Rhône-Alpes & GRAVIR Lab. (CNRS, INPG, UJF)
http://emotion.inrialpes.fr

***Abstract*— Autonomous navigation in open and dynamic environments is an important challenge, requiring to solve several difficult research problems located on the cutting edge of the state of the art. Bassically, these problems can be classified into three main categories: SLAM in dynamic environments; Detection, characterization, and behavior prediction of the potential moving obstacles; On-line motion planning and safe navigation decision based on world state predictions. This paper addresses some aspects of these problems and presents our latest approaches and results. The solutions we have implemented are mainly based on the followings paradigms: *Characterization and motion prediction* of the observed moving entities using bayesian programming; *On-line goal-oriented navigation decisions* using the Partial Motion Planning (PMP) paradigm.**

I. Introduction

A. Outline of the problem

To some extent, autonomous navigation for robotic systems placed in stationary environments is no longer a problem. The challenge now is autonomous navigation in open and dynamic environments, *i.e.* environments containing moving objects (potential obstacles) whose future behaviour is unknown. Taking into account these characteristics requires to solve several difficult research problems at the cutting edge of the state of the art. Basically, these problems can be classified into three main categories:

- Simultaneous Localisation and Mapping (SLAM) in dynamic environments;
- Detection, tracking, identification and future behaviour prediction of the moving obstacles;
- On-line motion planning and safe navigation

In such a framework, the system has to continuously characterize the fixed and moving objects that can be observed both with on-board or off-board sensors. As far as the moving objects are concerned, the system has to deal with problems such as interpreting appearances, disappearances, and temporary occlusions of rapidly manoeuvring objects. It also has to reason about their future behaviour (and consequently to make predictions). From the autonomous navigation point of view, this means that the system has to face a double constraint: constraint on the response time available to compute a safe motion (which is clearly a function of the dynamicity of the environment), and a constraint on the temporal validity of the motion planned (which is a function of the validity duration of the predictions). In other words, one needs to be able to plan motion fast, but one does not need to plan motion very far in the future.

This paper addresses some aspects of the previous problem, and presents our latest approaches and results. The solutions we have implemented rely on the following modules:

- *Scene interpretation and short-term motion prediction* for the moving obstacles, using the new concept of Bayesian Occupancy Filters and an efficient wavelet-based representation of the related occupancy grids;
- *Medium-term motion and behaviour prediction* for the observed moving objects, using motion pattern learning and Hidden Markov Models;
- *On-line goal-oriented navigation decision* using the Partial Motion Planning (PMP) paradigm.

B. Case Study: The Automated Valet Parking

One possible (and very relevant) target application for the techniques presented in this paper is that of the Automated Valet Parking (AVP). The robotic system considered is a "smart" car operating autonomously in a "smart" city parking. Both the car and the parking are equipped with sensors providing them with information about the world. Let us imagine the following scenario: you drive your car and leave it at the entrance of a given parking. From then on, it operates autonomously and go park itself. As soon as the car enters the parking, the car on-board intelligent system connects to the parking's own intelligent system and request a free parking place. The parking then confirms the availability of a parking space and provides the car with a model of the parking and an itinerary to the empty place. From then on, the car, using information obtained from both its own sensors and the parking sensory equipment, go park itself.

From an architecture point of view, the AVP scenario involves two "intelligent" systems communicating with one another: the car on-board system and the parking off-board system. As mentioned earlier, it is assumed that both systems are equipped with sensors providing them with information about the environment considered. While the car sensors will provide it with a local view of its surroundings, it can be expected that the parking sensors will provide the car with an overall view of what is going on in the whole parking.

To address the AVP scenario, we have devised a scheme relying upon the following functionalities (split between the car and the parking)

Parking abilities:

- Parking monitoring: at any time, the parking should know which places are free or not (and by whose car they are occupied).

- Route planning: the parking should be able to provide the car with a model of the parking premises along with the best itinerary to reach a given place.
- Moving objects monitoring: any moving objects (vehicles, pedestrians, etc.) should be monitored and tracked. The parking should be able to provide information such as position, speed and expected trajectory (*i.e.* future behaviour). Expected trajectories can come from different clues: typical movements of the different kinds of moving objects, learnt from previous observation, or knowledge of a planned trajectory.
- Car localisation: given its moving objects' monitoring functionality, the parking can provide the car with its current state in the parking premises.

Car abilities:

- Localisation: the car should be able to maintain an estimate of its localisation in the parking. It can be the result of a data fusion between parking information and on-board localisation.
- Environment modelling: the car on-board sensor are primarily used to build a model of the surroundings of the car. This local model should be enriched using the global information provided by the parking (in particular, the information concerning the moving objects' future behaviour).
- Automated driving: given the parking model and the route to the goal, the car should be able to determine its future course of action so as to reach its goal efficiently and safely.

One can notice that some of the functionalities mentioned above are somewhat redundant (in particular when dealing with sensing data). This property has intentionally been chosen in order to increase the robustness and the efficiency of the system:

- Fusion of data from multiple source increase overall accuracy.
- Using several data source increase fault tolerance.
- By correlating different inputs, it is possible to diagnose if an input is failing or becoming unreliable.

C. Outline of the paper

In this paper, we focus on two of the functionalities mentioned in the previous section: *Motion prediction of the observed moving objects* and *on-line goal-oriented navigation decisions*. The paper is organized in three main sections. The section II describes how we have solved the problem of interpreting and representing the dynamic environment of the robot using the "Bayesian Occupancy Filtering"(BOF) approach; this approach relies on a local world-state bayesian interpretation scheme, including a short-term motion prediction mechanism. The section III deals with the problem of the prediction of the most likely behaviors of some observed objects executing "intentional motions"; the proposed solution relies on the use of a motion pattern learning mechanism and of a hierarchical Hidden Markov Model. The section IV deals with the problem of planning safe motions in a reconstructed dynamic environment; the proposed paradigm (called "Partial Motion Planning", or PMP) takes into account (at each iteration step) both the time constraints and the current model of the future state of the robot environment.

II. Scene interpretation and short-term motion prediction

A. Overview of the problem

The problem addressed in this section concerns the interpretation of the observed dynamic scene in terms of *potential moving obstacles*, i.e. obstacles which may generate a collision in the near future with the robot). The objective is to be able to correctly interpret the dynamic scene in the presence of noisy or missing data, and to be as robust as possible to temporary or partial occlusions. Our approach for solving this problem is based on the new concept of *Bayesian Occupancy Filtering (BOF)* [1], where the robot environment is represented using a *4-dimensional occupancy grid*, i.e. an occupancy grid which includes the velocity dimension.

The *occupancy grids* [2], [3] framework is a classical way to describe the environment of a mobile robot. It has extensively been used for static indoor mapping [4] using a 2-dimensional grid. More recently, occupancy grids have been adapted to track multiple moving objects [5]. However, a major drawback of these approaches, is that a moving object may be lost due to occlusion effects.

The BOF approach avoid this problem for short temporary occlusions (e.g. a few seconds), by combining two complementary phases in a recursive loop: the *estimation phase* which estimate the occupancy probability of each cell of the 4-dimensional grid, using recursively the set of sensor observations; the *prediction phase* which estimate an a priori model of the grid occupancy at time $k+1$, using a "dynamic model" and the latest estimation of the grid state (figure 2 illustrates). This approach has been developed using the *Bayesian Programming* Framework [6], [7], [8]; it is described in the next sections.

However, large scale environments can hardly been processed in real-time because of the intrinsic complexity of the related inferences and numerical computations (see section II-F). The section II-G presents the outline of the WOG model ("Wavelet Occupancy Grid") we are developing for trying to meet the required efficiency property.

B. Estimation of the occupancy probability

The *estimation phase* consists in estimating, at each time step, the occupancy probability of each cell of the 4-dimensional grid. This estimation is performed using recursively the set of "observations" (i.e. pre-processed sensors data) provided by the sensors at each time step. These observations are represented by a list of detected objects, along with their associated positions and velocities in the reference frame of the processed sensor (several sensors may be used in parallel). In practice, this set of observations could also contain two types of false measurements : the *false alarms*, i.e. when the sensor detects a non existing object; the *missed detection*, i.e. when the sensor does not detect an existing object.

Solving the related estimation problem is done by using the available instantaneous information about the environment

state (i.e. the current observations and grid state). A sketch of the algorithm is given below using our *Bayesian Programming* Framework [6], [7], [8]; a more complete description of the method (which includes a "sensor data fusion" step) can be found in [1].

(i) Choosing the relevant variables and decomposition.

- O_{x,y,v_x,v_y}: The occupancy of the cell (x, y, v_x, v_y) at time t: occupied or not. This variable is indexed by a 4-dimensional index that represents a position and a speed relative to the vehicle.
- $\mathcal{Z}$: The sensor observation set; one observation is denoted Z_s; the number of observation is denoted S.

If we make the reasonable assumption that all the observations of the sensors are independent when knowing the occupancy state of a cell, we can choose to apply the following decomposition of the related joint distribution :

$$P(O_{x,y,v_x,v_y}, \mathcal{Z}) = P(O_{x,y,v_x,v_y}) \times \prod_{s=1}^{S} P(Z_s \,|O_{x,y,v_x,v_y}). \tag{1}$$

(ii) Assigning the parametric forms.

According to our knowledge of the problem to be solved, we can assign the following parametric forms to each of the terms of the previous decomposition:

- $P(O_{x,y,v_x,v_y})$ represents the *a priori* information on the occupancy of each cell. If available, a *prior* distribution could be used to specify it. Otherwise, a uniform distribution has to be selected. The next section will show how the prior distribution may be obtained from previous estimations.
- The shape of $P(Z_s \,|O_{x,y,v_x,v_y})$ is given by the *sensor model*. Its goal is to model the sensor response knowing the cell state. Details about this model can be found in [3]; an example is given for a telemetric sensor in Fig. 1. Such models can easily been built for any kind of sensor.

(iii) Solution of the problem.

It is now possible to ask the *Bayesian question* corresponding to the searched solution (i.e. the searched probability distribution). Since the problem to solve consists in finding a good estimate of the cell occupancy, the question can be stated as follows :

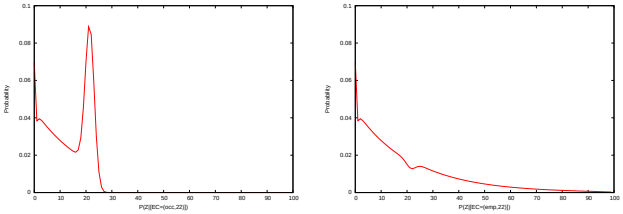
$$P(O_{x,y,v_x,v_y} \mid \mathcal{Z})? \tag{2}$$

The result of the related Bayesian inference[1] can be written as follows :

$$P(O_{x,y,v_x,v_y} \mid \mathcal{Z}\ ) \propto \prod_{s=1}^{S} P(Z_s \mid O_{x,y,v_x,v_y}\ ). \tag{3}$$

C. The Bayesian Occupancy Filter

We are now interested in taking into account the sensor observation history, in order to be able to make more robust estimations in changing environments (i.e. in order to be able to process temporary objects occlusions and detection problems). A classical way to solve this problem is to make use of Bayes filters [9]. Basically, the goal of such a filtering process is to recursively estimate the probability distribution $P(O^k_{x,y,v_x,v_y} \mid Z^k)$, known as the *posterior* distribution. In general, this estimation is done in two stages: a *prediction* stage whose purpose is to compute an *a priori* estimate of the target's state known as the *prior* distribution; an *estimation* stage whose purpose is to compute the *posterior* distribution, using this *a priori* estimate and the current measurement of the sensor. Exact solutions to this recursive propagation of the posterior density do exist in a restrictive set of cases (such as the Kalman filter [10][11] when linear functions and gaussian distributions are used).



(a) Sensor model for occupied cells

(b) Sensor model for empty cells

Fig. 1. Example of one-dimensional sensor models. The sensor is a laser-range finder located in $x = 0$ and detecting an object at $x = 22$. The following property holds : $P(Z|[O_x = occ]) = P(Z|[O_x = emp])$ for $x > z$, which implies that $P(O_x|Z)$ is unknown when the cell is behind the obstacle.

Our approach for solving this problem, is based on the new concept of *Bayesian Occupancy Filter (BOF)*; Fig. 2 shows the related estimation loop. This approach is derived from the general bayesian filtering paradigm; it provides a powerfull formalism for solving difficult estimation problems expressed in our 4-dimensional occupancy grid representation.

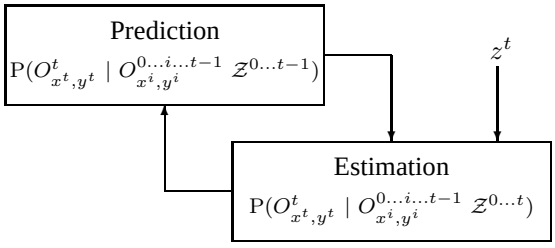


Fig. 2. Bayesian Occupancy Filter as a recursive loop.

The basic idea underlying the conception of the BOF is to make use of the velocities measured in the past, for predicting the near future and propagating this information through time. Indeed, knowing the current velocity of a mobile, it is possible to predict its future motion under the hypothesis of a constant velocity for the next time step (in practice, possible velocity changes will generate several possible future positions).

A complete presentation of the BOF can be found in [1]. In the sequel, we will just give an overview of the approach under the following simplifying assumptions (for clarity reasons) : use of a single sensor and constant velocity for the observed moving objects. A consequence of this last assumption, is that

[1]This result is computed using our $ProBT$ inference engine, currently commercialized by our spin-off company $Probayes$.

we can deal with a single "antecedent cell" when evaluating the occupancy state of a given cell.

(i) Choosing the relevant variables and decomposition.

- $O^t_{x,y}$: Occupancy of the cell $c = (x, y, v_x, v_y)$ at time t, occupied or not.
- $O^{t-\delta t}_{x_p,y_p}$: Occupancy of the cell which is the antecedent of $O^t_{x,y}$, occupied or not. In this model, $x_p = x - v_x\delta t$ and $y_p = y - vy\delta t$, since the velocity is constant.
- Z_s : A sensor observation.

Under the previous simplifying assumptions, the following decomposition of the joint distribution determined by these variables can be expressed as follow:

$$P(O^{t-\delta t}_{x_p,y_p}, O^t_{x,y}, Z_s) = \left(\begin{array}{l} P(O^{t-\delta t}_{x_p,y_p}) \\ \times P(O^t_{x,y}|O^{t-\delta t}_{x_p,y_p}) P(Z_s|O^t_{x,y}) \end{array} \right). \quad (4)$$

(ii) Assigning the parametric forms.

- $P(O^{t-\delta t}_{x_p,y_p})$ is the *prior* for the future occupancy state of the cell $c = (x, y, v_x, v_y)$. For each cell c such as the antecedent (x_p, y_p, v_x, v_y) is out of the current grid, this prior is the probability that a new object enters in the monitored space; since we usually have no real information about such an event, this probability is represented by a uniform distribution.
- $P(O^t_{x,y}|O^{t-\delta t}_{x_p,y_p})$ is related to the very simple "dynamic model" we are using in this case. It is defined as a transition matrix $\begin{bmatrix} 1-\epsilon & \epsilon \\ \epsilon & 1-\epsilon \end{bmatrix}$, which allows the system to take in account the fact that the null acceleration hypothesis is an approximation; in this matrix, ϵ is a parameter representing the probability that the object in $c = (x_p, y_p, v_x, v_y)$ does not follow the null acceleration model.
- $P(Z_s|O^t_{x,y})$ is the sensor model (see section II-B).

(iii) Solution of the problem.

Similarly to the estimation process described in the section II-B, the solution of the problem to be solved by the BOF can be defined by the following Bayesian question : $P(O^t_{x,y}|Z_s)$?. Answering this question (i.e. computing the related probability distribution) is achieved using our inference engine; this inference involves a marginalization sum over $O^{t-\delta t}_{x_p,y_p}$.

D. Experimental results

This approach has been implemented and tested on our experimental platform : the Cycab vehicle equiped with a laser sensor. Fig. 3 shows some resulting grid estimations, for an environment containing two stationary objects and an object moving from the left to the right at a velocity of $0.8\ m/s$; in this example, the robot is not moving.

Fig. 3b depicts the occupancy probability of each cell corresponding to a null relative velocity (*i.e.* $c = [x, y, 0, 0]$). As expected, two areas with high occupancy probabilities are visible; these probability values depends on several factors attached to the sensor model : the probability of true detections, the probability of false alarms, and the sensor accuracy. Since the measured speed for the third obstacle is not null, any area of high occupancy probability corresponding to this observation is only represented in the related slices of the grid (i.e. the slice corresponding to $c = [x, y, 0, 0.8]$ in this case, see Fig. 3c).

It should be noticed that the cells located outside of the sensor field of view, or the cells "hidden" by one of the three sensor observations (i.e. the cells located behind the three detected obstacles) cannot be observed; consequently, nothing really consistent can be said about these cells, and the system has given an occupancy probability value of 0.5 for these cells.

Fig. 4 shows a sequence of successive prediction and estimation results given by the BOF. The experimental scenario involves a stationary obstacle and the Cycab moving forward at a velocity of $2.0\ m/s$. The obstacle is detected using the laser sensor; it finally goes out of the sensor field of view (see Fig. 4-d1), since the Cycab is moving forward. It should be noticed that the prediction step allows to infer knowledge about the current occupancy state of the cycab environment, even if the object is no longer observed by the sensor; this is the situation depicted by fig 4-d3, where an area of high occupancy probability still exists when the object is going out of the sensor field of view. In some sense, our prediction step can be seen as a "short-term memory", which allows to combine in an evolutive way past and current observations.

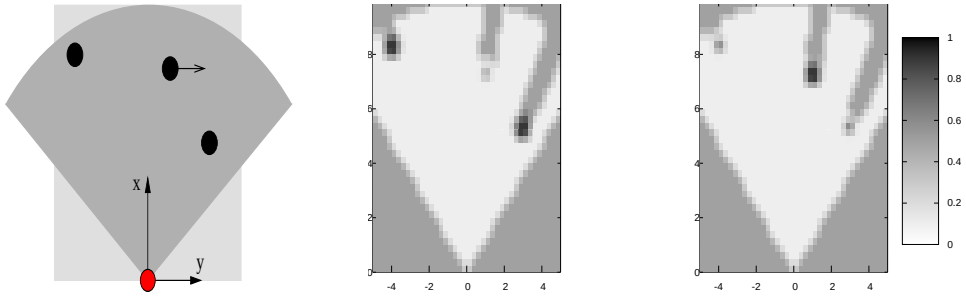
E. BOF based collision avoidance

In [1], we have shown how to avoid a mobile obstacle by combining the occupancy grid result given by the BOF, with a *danger probability* term computed using a "time to collision" criteria. In this approach, each cell O^k_{x,y,v_x,v_y} of the grid is characterized by two probability distributions : the "occupancy" term $P(O^k_{x,y,v_x,v_y} \mid \mathcal{Z}^k)$ and the "danger" term $P(D^k_{x,y,v_x,v_y} \mid O^k_{x,y,v_x,v_y})$. Using this model, it becomes possible to tune the velocity controls of the Cycab, according to the results of a combination of the two previous criteria.

This approach has experimentally been validated using the following scenario : the Cycab is moving forward; a pedestrian is moving from right to left, and during a small period of time the pedestrian is temporarily hidden by a parked car. Fig 5 shows some snapshots of the experiment : the Cycab brakes to avoid the pedestrian which is temporarily hidden by the parked car, then it accelerates as soon as the pedestrian has crossed the road.

F. Discussion and performances

Thanks to the previous assumptions, both the prediction step and the estimation step complexities *increases linearly with the number of cells* of the grid. This make the approach tractable in real situations involving reasonable grid sizes. This is the case for the experimental scenarios described in this section (e.g. size of the sensed space of $10\ m\ m$ with a resolution of $0.5\ m$, longitudinal velocity of $-3\ m.s^{-1}$ to $1\ m.s^{-1}$ with a resolution of $0.4\ m.s^{-1}$, lateral velocity of $-3.2\ m.s^{-1}$ to $3.2\ m.s^{-1}$ with a resolution of $0.4\ m.s^{-1}$). Using such a grid of 64.000 cells, the computation time for both prediction and estimation steps is about $100\ ms$ on a $1\ GHz$ computer. This



a)
$z_1 = (8.3, -4, 0, 0)$
$z_2 = (5, 3, 0, 0)$
$z_3 = (7.3, 1.9, 0, 0.8)$

b)
$P([O_C = occ] \,|\, \mathcal{Z})$
$[c = (x, y, 0, 0)]$

c)
$P([O_C = occ] \,|\, \mathcal{Z})$
$[c = (x, y, 0, 0.8)]$

Fig. 3. Example of a grid estimation using a single laser sensor located at $(0, 0)$, in a scene including two stationary objects and an object moving from the left to the right at a velocity of $0.8\ m/s$. In this example, the robot is not moving. The occupancy probability value of each cell is represented by a gray level (see the colors correspondences on the right side of the figure). (a) The sensed environment and the three instantaneous sensor observations expressed in the $(x, y, \dot{x}, \dot{y})$ space. (b) Occupancy probability in a 2-dimensional slice of the grid corresponding to a null relative velocity (*i.e.* $c = [x, y, 0, 0]$). (c) Occupancy probability in a 2-dimensional slice of the grid corresponding to a relative horizontal velocity 0.8 (*i.e.* $c = [x, y, 0, 0.8]$).

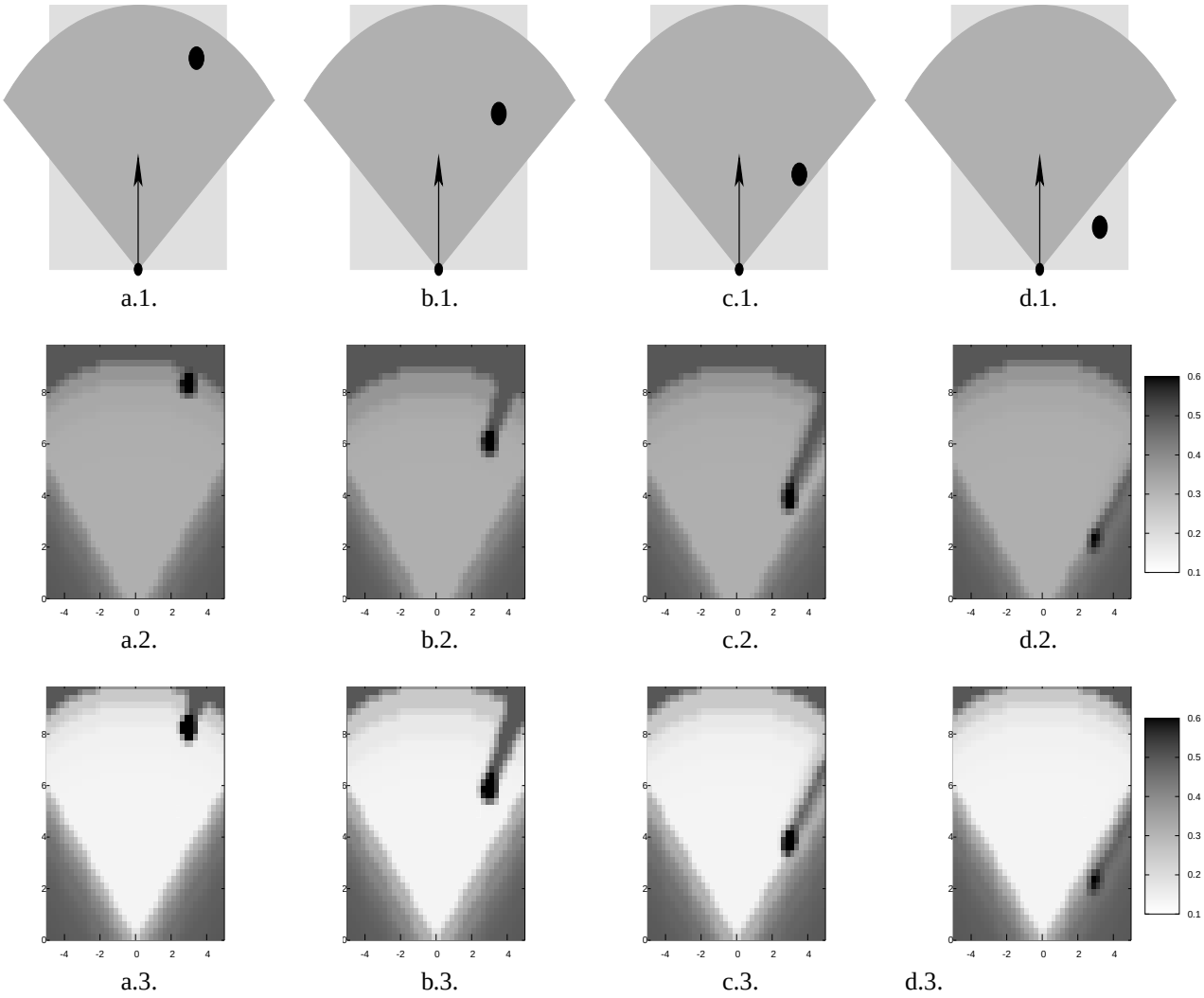


Fig. 4. A short sequence of a dynamic scene involving a stationary obstacle and the Cycab moving forward at a velocity of $2.0\ m/s$. The second and the third row respectively show the results of the prediction and of the estimation steps of the BOF, in a 2-dimensional slice of the grid corresponding to a relative speed of $\dot{x} = -2.0$, i.e. $P([O^t_{x\ y\ [\dot{x}=-2.0]\ [\dot{y}=0.0]} = occ]\ )$.

Fig. 5. Snapshots of the experimental pedestrian avoidance scenario.

is fast enough to control the CyCab at a maximum speed of 2 $m.s^{-1}$.

However, this grid size is not fine enough for some other large scale applications involving higher speeds. In this case, the number of cells increases quickly, and the required computational time becomes too high for satisfying the real-time constraint.

In order to try to solve this problem, we are working in two directions. The first one consists in developing a dedicated hardware exploiting the *highly parallel* structure of the BOF algorithm[2]. The second approach consists in using a multi-resolution framework for representing the 4-dimensional grid and the related processing models. The outline of this approach is described in the next section.

G. Wavelet-based model for the BOF (the WOG model)

(i) Overview of the problem.

The goal of this new model is to provide a "coarse-to-fine" representation of the underlying BOF model, in order to be able to optimize the processing of large homogeneous regions, and to refine the model only when this is necessary. We have chosen to make use of the wavelet framework [12] [13], which allows the processing of large sets of data including non-linearities (as it is the case for our dynamic maps). Pai and Reissell [14] have shown that wavelets could be used for representing 3D static maps for path planning. Sinopoli [15] has extended this approach for solving a global path-planning problem, while using traditional 3D Occupancy Grids (OG) for local navigation.

Our approach, called *"Wavelet Occupancy Grid"* (WOG), can be seen as a tight combination of wavelet and OG representations, allowing us to perform OG *updates in the wavelet space*, and later on to make "prediction inferences" within the wavelet space (i.e. to fully implement the BOF paradigm in this multi-resolution formalism).

(ii) The WOG model.

The first objective is to develop a wavelet-based model for 2-dimensional OG (i.e. without representing velocities). At this step of the development, we will only consider the random variables $O^t_{x,y}$ and Z^t (defined above). Since each occupancy variable ($O_{x,y}$) is binary, its probability distribution is completely defined by $P([O^t_{x,y} = occ])$. Then, we consider now the occupancy function $p^t(x,y) = P([O^t_{x,y} = occ])$ which allows to capture the space homogeneity.

The used wavelet model.

Basically, linking OG and wavelet representations, leads to project a huge function representing the OG into a wavelet vector space. In our approach, we have used the 2D Haar model [13], built upon two types of basis functions: the "scale" and "detail" functions, where the scaling mother function is defined as follows :

$$\Phi(x,y) = 1 \text{ for } (x,y) \in [0,1]^2, \text{ zero elsewhere}$$

Then, the Haar basis at scale s is defined as the union of the set of "scaled" functions $\{\Phi^{sij} | (i,j) \in \mathbb{Z}^2\}$ and the set of "details" functions $\{\Psi^{lij}_M | l = 0, \ldots, s; (i,j) \in \mathbb{Z}^2\}$, where :

$$\Phi^{lij} \quad = \quad 2^{-l}\Phi(2^{-l}x - i, 2^{-l}y - j) \tag{5}$$

and the type M can take three values (01, 10 or 11) corresponding to one of the three mother wavelets for horizontal, vertical and diagonal differencing.

Each t-uplet (s,i,j) defines a *wavelet square* at scale s and an offset (i,j). Thanks to the Haar model property, the projection of the occupancy function over a basis vector function Φ^{sij} is given by the following scalar product of Φ^{sij} with the occupancy function :

$$< p(x,y) \mid \Phi^{sij} >= \int_{x,y \in \mathbb{R}^2} p(x,y)\Phi^{sij}(x,y)d_{x,y} \tag{6}$$

Logarithmic form for OG updates.

In the sequel, we will omit the index (x,y) associated to the OG cells (for simplifying the notations). A standard update in OG is defined by the following equation :

$$p([O^t = occ]) = \frac{p([O^{t-1} = occ])p(z^t|[O^{t-1} = occ])}{\sum_{o^t} p(o^t)p(z^t|o^t)} \tag{7}$$

Let $p^t(occ) = p([O^t = occ])$ and $p^t(emp) = p([O^t = emp])$; then we can write $p^t(occ) = 1 - p^t(emp)$, and we can summarize this property using a single coefficient q^t :

$$q^t \quad = \quad p^t(occ)/p^t(emp) \tag{8}$$

Substituting eq. 7 into eq. 8 leads to the elimination of the marginalisation term; then, we can write the following

[2]thanks to the hypothesis that each cell is independent, the state of each cell can be computed independently.

recursive formulation :

$$q^t = \frac{p(z^{t-1}|occ)}{p(z^{t-1}|emp)} q^{t-1} = q^0 \prod_{i=0}^{t-1} \frac{p(z^i|occ)}{p(z^i|emp)} \quad (9)$$

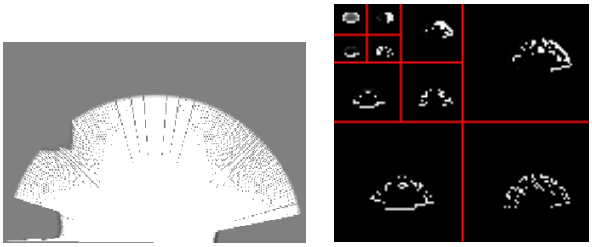
where q_{t-1} is recursively evaluated.

Such updating operations are clearly non-linear. In order to be able to perform the update operations using only sums, we have decided to make use of a logarithmic form of the model :

$$odds^t = odds^0 + \sum_{i=0}^{t-1} \log(\frac{p(z^i|occ)}{p(z^i|emp)}) \quad (10)$$

where $odds^t = log(q^t)$.

Let $Obs(z^{t-1})$ be the term $\log(\frac{p(z^{t-1}|occ)}{p(z^{t-1}|emp)})$ in the eq. (10). $Obs(z^{t-1})$ represents the "observation" term of the update, and $odds(t-1)$ is the a priori term. Then, updating a WOG can be done by adding the wavelet transform of observation terms to the wavelet representation of the map.



(a) (b)

Fig. 6. Mapping obtained by a laser range-finder : (a) the obtained OG model; (b) the related three first "detail spaces" of the WOG model and the complementary scaled space. The density of non-zero coefficients decreases drastically in wavelet space.

Multi-scale semantic of logarithmic OG.

In the case of the Haar wavelet basis (eq. 5), $\Phi_s^{(i,j)}$ has a constant value (k) over the domain[3] (s,i,j), and is equal to zero elsewhere. Then the integral of the scalar product is a sum over the cells enclosed in the domain (s,i,j); this sum can be re-written as follows [16] :

$$< \log(\frac{p^t(x,y)}{1-p^t(x,y)}) \mid \Phi^{sij} >$$

$$= \int_{x,y \in \mathbb{R}^2} \log(\frac{p^t(x,y)}{1-p^t(x,y)}) \Phi^{sij}(x,y) d_{x,y} \quad (11)$$

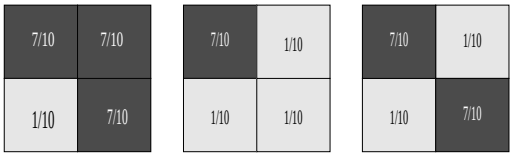
$$= k \log(\frac{\Pi_{c \in (s,i,j)} p([O_c = occ])}{\Pi_{c \in (s,i,j)} p([O_c = emp])}) \quad (12)$$

$$= k \log(q_s) \quad (13)$$

where c is the index of a cell in the square domain. $k \log(q_s)$ is the log of the ratio of two geometric means (cells are "all occupied" and cells are "all empty") which leads us to a continuous semantics. Let define *full* as the event "every subcell is occupied" then:

$$\Pi_{c \in (s,i,j)} p([O_c = occ]) = p(\bigwedge_{c \in (s,i,j)} [O_c = occ]) = p(full)$$

Let define *open* as the event "every subcell is empty".



(a) (b) (c)

Fig. 7. Three configurations of a subsquare domain at scale 1/2: the probability that the square domain 7(a) is fully "occupied" is 0.0343; its probability of being fully "empty" is 0.0243. The occupancy ratio related to the "occupied" and "empty" properties is $P(open|open \vee full) = 0.58$; this means that the related square domain is considered as "occupied". The occupancy ratio in 7(b) is 0.003; it is of 0.06 in 7(c).

q_s leads immediately to the conditional probabilities:

$$P(open|open \vee full) = q_s/(1+q_s) \quad (14)$$

$$P(full|open \vee full) = 1/(1+q_s) \quad (15)$$

which express a continuous ratio between the two events *full* and *open* (Fig. 7).

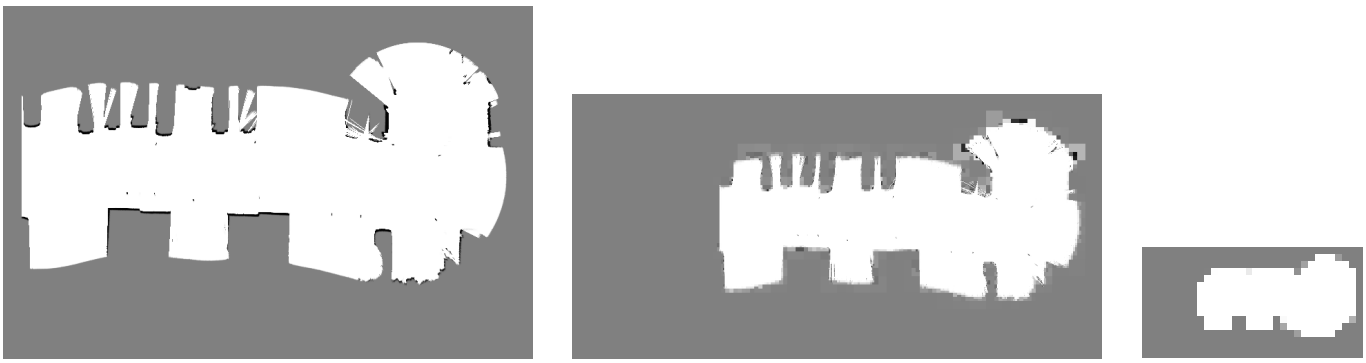
The multi-scale information which can be computed by this approach, is directly derived from these two basic cases. Consequently, only 2 relevant events can be considered for a square domain (s,i,j) containing n subcells (i.e. where 2^n possibilities can be defined by the binary vector representing the possible occupancy of these n subcells), Fig. 8(c) illustrates. Fig. 7 shows the information which can be derived from the previous multi-scale occupancy model. The next step will be to exploit this concise information for elaborating multi-scale algorithms.

(iii) WOG implementation and experimental results.

This approach has been implemented and tested on the Cycab equipped with a laser sensor an with our SLAM method [17]. In the experiments, the size of the finer scale is equal to $6.25cm$, and we have used 5 different scales (where the size is multiplied by 2 for each new scale); thus, the size of the coarsest cells is equal to $1\ m$. The size of the square window which is used for constructing the WOG model, is chosen in order to contain the whole sensor field of view. The content of this window is updated at each laser scan (see Fig. 6; then, the Haar wavelet transform is applied to this sensory data, and incrementally added to the current WOG model. The chosen compression threshold is a compromise between data fitting and sparsity (wavelet terms which are lower than the threshold are removed).

Fig. 8 shows the results obtained on the car park of INRIA. These experimental results clearly shows that we have obtained a significant reduction of the size of the model (about 80% relatively to the OG model), and that the interesting details are still represented (such as the beacons represented by

[3]With value: $k = 2^{-s}$.



(a) (b) (c)

Fig. 8. Maps of the car park of INRIA. (a) The OG model contains $393,126$ cells, while the WOG model contains $78,742$ cells. (b) The OG model reconstructed from the previous WOG model; it can be seen that significant details such as beacons have been captured by the WOG representation (the shapes of maps (a) and (b) are very close). (c) The empty space is pretty well approximated at the coarser scale.

dark dots in Fig. 8(b)). It should be noticed that the coarser model give a quite good representation of the empty space (see Fig. 8(c)); this model could be used for path planning, and refined when necessary using the wavelet model. In the previous experiments, the map building has been done in real-time.

One of the major issue of our approach is now, to combine bayesian occupancy filter and object tracking such as the informations about the object motions could be transmitted to the high level object motion prediction module. One of the possibilities, we explore now is to extract coherent zones of the BOF in term of occupancy and velocity and that we call them objects. Our hope is that these zones would appear hierarchically through the scales in the wavelet representation, and so woubld be easy to find.

III. Medium-term motion prediction

The work presented in this section focuses on motion prediction for objects which are able to execute trajectories as a result of an internal motion planning process or decision mechanism (*e.g.* persons, animals and robots). It is assumed that such plans are made *with the intention* to reach a specific goal, thus the name *intentional motion* which will be used hereafter to designate this kind of motion.

Assuming that the object's decision mechanism as well as all the relevant variables at every time step (*e.g.* internal state, sensorial input, etc.) are known, predicting its trajectory consists in replicating the planning process in order to find the intended trajectory. However, this assumption is not realistic. Neither the planning model nor the variables are known or observable (what is the decision mechanism of a human being?) and they must be inferred from observed motion before performing prediction. This leads to the following decomposition of the problem:

- *Learning*. Construct a plan representation based on observations.
- *Prediction*. Use the representation obtained during learning to estimate future states on the basis of present knowledge.

Thus, learning consists in observing a given environment in order to construct a representation of every possible plan for it. But, how long should we observe the environment in order to construct such a "plan library"? Given the enormous number of possible plans for all but the simplest environments, there is not a simple answer. This raises an important problem of existing learning techniques (e.g. [18], [19]): the use of a "learn then predict" approach, meaning that the system goes through a learning stage where it is presented with a set of observations (an example dataset) from which it construct its plan models. Then, the plan library is "frozen" and the system goes into the prediction stage.

The problem with this approach is that it makes the assumption that all possible plans are included in the example dataset, which, as we have shown, is a difficult condition to meet. Our work addresses the problem by proposing a "learn and predict" approach which is able to learn in an incremental fashion (*ie* by continuously refining its knowledge on the basis of new observations used for prediction). To the extent of our knowledge, this is the first intentional motion prediction technique in the literature to have this property.

Learning techniques used by the "learn then predict" approaches are very diverse. For example in [20] plans are modeled as series of straight motion segments which are clustered together. In [18] and [21], typical behaviors are learned by clustering whole trajectories. In [22] Bui proposes Abstract Hidden Markov Models as a way to represent plans as hierarchies of probabilistic sub-plans or policies. Although the approach does not define an automatized learning mechanism, this has been done in [19] by using the Expectation-Maximization algorithm.

In this section, we present our "learn and predict" approach which models plans as Hidden Markov Models (HMM)[23] augmented with a variable which indicates the goal that the plan intends to reach The learning algorithm is composed of two modules: in the first one, the Grow When Required algorithm (GWR) [24] is used to estimate both the set of states in the model and the observation probabilities. The second module identifies goals and then uses a Maximum-Likelihood

criterion to update the transition probability of the model. As mentioned above, the technique determines the number of goals and states in the model, thus learning the structure of the underlying HMM.

A. Proposed Approach

The proposed unsupervised learning algorithm, constructs plan representations by observing the motion of objects (e.g. pedestrians, vehicles, etc.) moving in a given environment.

The input of the learning algorithm is a continuous stream of observations $z_t = \{z_1, z_2, \cdots\}$ gathered through a tracking system. In order to keep notation simple, we will assume that no more than one object is observed at the same time, noting that the approach is easily generalizable to the multi-object case. It will also be assumed that the tracking system can determine when the object has stopped or exited the environment.

Every observation $z_t = (x_t, y_t, \eta_t)$ returned by the tracker consists of an estimate of the object's position[4] at time t and a binary variable η_t which indicates whether the object has reached the end of its trajectory ($\eta = 1$) or not ($\eta = 0$). A trajectory ends when the object stops moving or exits the environment.

Learning will consist in estimating the parameters of the modified HMM which will be presented in the following section.

1) Plan Modeling: The model used in this approach is defined by three components: 1) a set of relevant *variables* which define a joint probability distribution (JPD), 2) a *decomposition* of the JPD which is obtained by applying Bayes rule in order to reflect conditional independence assumptions and 3) the *parametric forms* used to represent each of the terms which appear in the decomposition. They are called *parametric* because they include parameters which may be adjusted either manually or through an automatic parameter estimation (*ie* learning) mechanism.

Relevant Variables:

- $N \in \mathbb{N}$: The total number of discrete states in the model. These states correspond to positions in the environment.
- $q_t \in [1, N]$: The object's state at time t.
- $q_{t-1} \in [1, N]$: The object's state at time $t-1$.
- $z_t \in \mathbb{R}^2$: The object's state estimation returned by the sensor at time t. (*ie* the observation variable).
- $G \in \mathbb{N}$: The total number of goals in the model. The goals correspond to specific *places* in the environment (*ie* it may correspond to many discrete states).
- $\gamma \in [1, G]$: The place that the object intends to reach (*ie* its goal).

Decomposition: The Joint Probability Distribution is decomposed as follows:

$$p(q_t, q_{t-1}, z_t, \gamma) = p(q_{t-1})p(\gamma)p(q_t \mid \gamma, q_{t-1})p(z_t \mid q_t) \quad (16)$$

This decomposition implies two hypothesis: a) knowing the state, subsequent observations are independent of each other and b) knowing the intended goal, the present state depends only in the past state (*ie* knowing the goal, the system becomes a first order Markov process and behaves like a conventional HMM).

The parametric forms of the probabilities that compose the JPD are presented in table I.

- $p(q_{t-1})$: Uniform $\mathcal{U}_N = \frac{1}{N}$.
- $p(\gamma)$: Uniform $\mathcal{U}_G = \frac{1}{G}$.
- $p(q_t \mid \gamma, q_{t-1})$: Table.
- $p(z_t \mid [q_t = i])$: Gaussian $\mathcal{G}(\mu_i, \sigma_i)$.

TABLE I

PARAMETRIC FORMS OF THE PROBABILITY DISTRIBUTIONS.

2) Parameter Learning: Learning the model's parameters consists in finding the values of the conditional probabilities shown in table I. Our approach splits the problem in three tasks:

1) *State GWR*. The observation probability $p(z_t \mid q_t)$ and the number of states N are estimated using the Grow When Required algorithm (§III-B).
2) *Goal GWR*. Another instance of GWR is used to estimate the number of goals G as well as their position.
3) *Transition Counting*. The Viterbi algorithm [25] is used to perform a maximum likelihood (ML) estimation of the transition probability $p(q_t \mid \gamma, q_{t-1})$. This estimation uses the outputs of tasks 1 and 2 (§III-D).

B. Learning observation probabilities and number of states

The observation probability for a given state $p(z_t \mid [q_t = i])$ is defined as a gaussian. Therefore, the learning algorithm should estimate the mean value μ_i and standard deviation σ_i for the N states.

This rises the question of the "correct" value for N, which is an important question. The state space is continuous, when it is mapped to a finite set of discrete values an error is introduced in the representation. The number of states allows to trade off accuracy and computational efficiency. By incrementing the value of N the approximation error – also known as distortion – is reduced at the expense of additional calculation burden.

There is another way of reducing the distortion: discrete states may be placed in such a way that the mean distance between them and observed data is minimized. This is known as Vector Quantization [26].

Our approach uses the Grow When Required (GWR) [24] algorithm to perform vector quantization in order to estimate the number of discrete states of the model as well as the mean values and standard deviation of the observation probabilities. This algorithm has been chosen between many different approaches existent in the literature [26], [27], [28], [29] due to its following properties:

- It is fast. The costliest operation is of $O(N)$. This can be further optimized by using a hierarchical structure like an r-tree[30].
- The number of states is not fixed. New states are added and deleted as observations arrive.

[4]Higher-dimensional observations (*ie* (x_t, y_t, x'_t, y'_t)) may also be used as input by the algorithm.

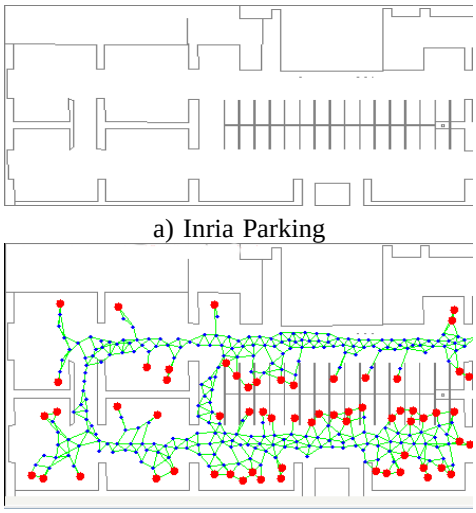
- It is incremental. This makes it suitable to process continuous streams of observations.

The algorithm processes observation on a one by one basis. It produces a graph, where nodes representing discrete states are explicitly linked to their closest neighbors (the graph is a subset of the Delaunay triangulation). Every node i is associated to a vector μ_i known as the centroid.

The application of this structure to estimate the required parameters is straightforward: state information $\{x_t, y_t\}$ contained in each observation is used as an input for a GWR. The resulting set of nodes represents discrete states whose centroids are the mean values of the observation probabilities. The standard deviation σ_i for state i is calculated by averaging the half length of the links emanating from the corresponding node.

Insertion of new states is no longer allowed when $\arg\min \sigma_i$ is less than a given threshold. This restrains the algorithm from discretizing the space below the sensor's precision.

An example of the use of GWR is presented in fig. 9. The environment is the parking lot of the INRIA building. It contains a number of places which may constitute motion goals for a car (*i.e.* parking places and the parking's exit). Fig. 9b presents the state of the GWR structure after processing 500 trajectories.



a) Inria Parking

b) Sample GWR structure.

Fig. 9. Using GWR to find discrete states in the INRIA paking lot. Big circles correspond to goals.

C. Identifying Goals

The problem of automatically identifying the goals that an object seeks to reach using only observation data is very difficult since these goals are often related to information which is not present in this data (*e.g.* the presence of a billboard).

The approach taken here aims to identify goals based on a simple hypothesis: when an object stops moving (or exits the environment) it is because it has reached its goal. This leads to a simple goal identification scheme: every observation z_t having $\eta_t = 1$ (end observation) is processed by a GWR structure which clusters this information together into high-level goals.

The nodes of the resulting GWR graph corresponds to goals[5]. The graph itself may be used to identify the goal that corresponds to a given end-state observation:

$$\gamma = \min \arg_i \|(x_t, y_t) - \mu_i^g\|, \text{for } \eta_t = 1 \tag{17}$$

D. Learning transition probabilities

Transition probabilities are updated once a complete trajectory is available, this means that all non-end observations are stored until an observation having $\eta = 1$ is received, then, expression 17 is used to compute the attained goal g. For every observation in the trajectory z_t, the Viterbi algorithm is used in order to find q_t given the past state $q_{t-1} = i$ (which has been estimated in the previous iteration)[6]:

$$q_t = \max \arg_j \left\{ p([q_t = j] \mid [\gamma = g], [q_{t-1} = i]) p(z_t \mid [q_t = j]) \right\}$$

The obtained values for g, i and j are then used as indices to update a transition count matrix A on a maximum-likelihood criterion:

$$A[g, i, j] \leftarrow A[g, i, j] + 1 \tag{18}$$

If the observation correspond to the first step of a trajectory only the current state is estimated using:

$$q_0 = \max \arg_i \{p(z_0 \mid [q_0 = i])\} \tag{19}$$

Transition probabilities are calculated using:

$$p([q_t = j] \mid [\gamma = g], [q_{t-1} = i]) = \frac{\mathcal{A}[g, i, j]}{\sum_h \mathcal{A}[g, i, h]} \tag{20}$$

Finally, when N or G change due to additions or deletion on the corresponding GWR structures the corresponding columns and rows are simply inserted or deleted accordingly, this is possible due to the fact that we are storing counts instead of probabilities in the transition matrix.

E. Motion Prediction

The model may be used to predict future states in one of two ways:

a) The state may be projected into the future a number k of timesteps. This is useful to predict intermediate positions of the object trajectory.

b) The goal probability may be directly estimated. This gives a prediction of the object final destination.

In order to achieve real-time performance when predicting, we have resorted to approximate inference using a particle filter with a resampling step [31]. The algorithm approximates the belief state by a set of particles $\mathcal{X} = \{\S', \cdots, \S^{\mathcal{M}}\}$. The

Algorithm 1 Particle Filter Algorithm($\mathcal{X}_{t-1}$, z_t)

1: $\hat{\mathcal{X}}_t \leftarrow \mathcal{X}_t \leftarrow \emptyset$
2: **for** every particle $x_{t-1}^m = \{q_{t-1}^m, \gamma^m\}; x_{t-1}^m \in \mathcal{X}_{t-1}$ **do**
3: sample q_t^m from $p(q_t^m \mid \gamma^m, q_{t-1}^m)$
4: $x_t^m = \{q_t^m, \gamma^m\}$
5: $w_t^m = p(z_t \mid q_t^m)$
6: $\hat{\mathcal{X}}_t \leftarrow \hat{\mathcal{X}}_t + < x_t^m, w_t^m >$
7: **end for**
8: **for** $m = 1$ to M **do**
9: draw i with probability $\propto w_t^i$
10: add x_t^i to $\mathcal{X}_t$
11: **end for**

pseudocode of the algorithm is presented in the following listing (adapted from [32]):

Using the filter, we may estimate the probability to reach a particular goal by counting the number of particles having that goal and normalizing by the total number of particles M. Examples of goal prediction are presented in fig. 10.

Fig. 10. Two examples of prediction shown at different moments (time progresses from left to right).

1) Conclusions and future work: We have started to work on a parking environment which constitutes the main testbed for the ParkView project. Experiments are currently being conducted and only preliminar results are available. Thus, our current primary goal is to conclude our experiments in order to measure the performance of our algorithm.

IV. Safe Navigation in Dynamic Environments

When placed in a dynamic environment, an autonomous system must consider the real time constraint that such an environment imposes. Indeed, the system has a limited time only to make a decision about its future course of action otherwise it might be in danger by the sole fact of being passive. The time available depends upon a factor that we will call the *dynamicity* of the environment and which is a function of the system and the moving objects'dynamics.

[5]notations μ_i^s and μ_i^g will be used henceforth in order to distinguish between state and goal GWR

[6]This implies iterating through the domain of q_t, meaning that the update step has cost $O(N)$.

In this context, basing the decision making process on a motion planning technique[7] leaves little hope to fulfil this real-time constraint given the intrinsic time complexity of the motion planning problem [33] (even if using randomised approaches). This certainly explain why so many reactive methods[8] have been developed in the past (*cf* [34], [35], [36], [37], [38], [39], [40], [41], [42], [43] or [44]. However, reactive approaches are confronted with two key issues: the *convergence* and the *safety* issues. As for convergence, their lack of lookahead may prevent the system to ever reach its goal. As for safety, what guarantee is there that the system will never find itself in a dangerous situation eventually yielding to a collision?

Partial Motion Planning (PMP) is the answer we propose to the problem of navigation in dynamic environments. It is especially designed in order to take into account the real-time constraint mentioned above. PMP is a motion planning scheme with an anytime flavor: when the time available is over, PMP returns the best partial motion to the goal computed so far. Like reactive scheme, PMP is also confronted to the convergence and safety issues. At this point, we have decided to focus on the safety issue and to propose a solution relying upon the the concept of *Inevitable Collision States* (ICS) originally introduced in [45]. An ICS is a state such that no matter what the future motion of the system is, it eventually collides with an obstacle. ICS takes into account the dynamics of both the system and the moving obstacles. By computing ICS-free partial motion, the system safety can be guaranteed.

PMP is detailed in section IV-A while section IV-B presents the ICS concept. Finally, an application of PMP to the case of a car-like system in a dynamic environment is presented in section IV-C.

A. Partial Motion Planning

As mentioned earlier, a robotic system cannot in general safely stand still in a dynamic environment (it might be collided by a moving obstacle). It has to plan a motion within a bounded time and then execute it in order to remain safe. The time δ_c available to calculate a new motion is function of the nature and dynamicity of the environment. To take into account the real-time constraint that stems from a dynamic environment, we propose a scheme that calculates partial motions only according to the the following cycle (also depicted in Fig. 11):

PMP Algorithm	
Step1:	*Get model of the future*
Step2:	*Build tree of partial motions towards the goal*
Step3:	*When δ_c is over, return best partial motion*
Step4:	*Repeat until goal is reached*

Like motion planning, partial motion planning requires a model of the environment, the first step is aimed at getting this model. The required model is provided by the environment modelling and motion prediction functions presented earlier. The periodic iterative PMP scheme proposed in this paper

[7]Wherein a complete motion to the goal is computed a priori.

[8]Wherein only the next action is determined at each time step.

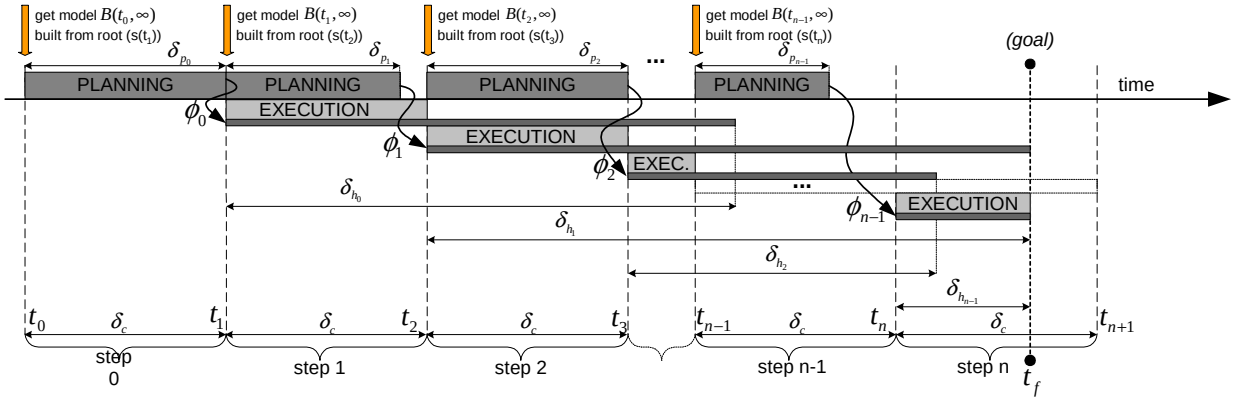


Fig. 11. Partial Motion Planning iterative cycle

accounts for both the planning time constraints and the validity duration of the predictions made.

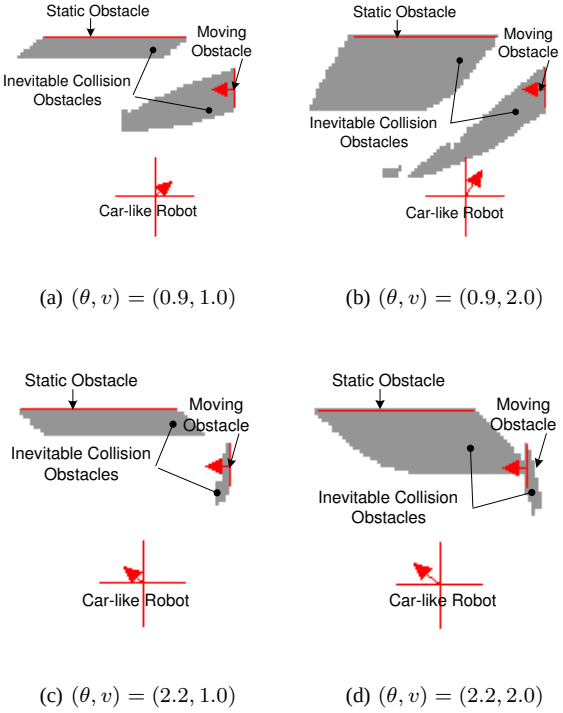
B. Inevitable Collision States

Like every method that computes partial motion only, PMP has to face a safety issue: since PMP has no control over the duration of the partial trajectory that is computed, what guarantee do we have that the robot $\mathcal{A}$ will never end up in a critical situation yielding an inevitable collision? As per [45], an Inevitable Collision State (ICS) is defined as a state s for which no matter the control applied to the system is, there is no trajectory for which the system can avoid a collision in the future. The answer we propose to the safety problem lies then in the very fact that the partial trajectory that is computed is ICS-free. Meaning that, even in the worst case scenario where the duration δ_{h_i} of the partial trajectory is shorter than the cycle time δ_c, $\mathcal{A}$ can always execute one of the existing safe trajectory. The overall safety is guaranteed as long as the initial state is ICS-free (which is something that can be reasonably assumed). Now, determining whether a given state of $\mathcal{A}$ is an ICS or not is a complex task since it requires to consider all possible future trajectories for $\mathcal{A}$. However, it is possible to take advantage of the approximation property demonstrated in [45] in order to compute a conservative approximation of the set of ICS. This is done by considering only a subset $\mathcal{I}$ of the full set of possible future trajectories.

C. Case Study

In this section we present the application of PMP to the case of a car-like vehicle $\mathcal{A}$ moving on a *planar* surface $\mathcal{W}$ and within a fully observable environment cluttered with stationary and dynamic obstacles. A control of $\mathcal{A}$ is defined by the couple (α, γ) where α is the rear wheel linear acceleration. and γ the steering velocity. The motion of $\mathcal{A}$ is governed by the following differential equations:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v}_r \\ \dot{\xi} \end{bmatrix} = \begin{bmatrix} v_r \cos\theta \\ v_r \sin\theta \\ \frac{\tan\xi v_r}{L} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \alpha + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \gamma \qquad (21)$$



(a) $(\theta, v) = (0.9, 1.0)$ (b) $(\theta, v) = (0.9, 2.0)$

(c) $(\theta, v) = (2.2, 1.0)$ (d) $(\theta, v) = (2.2, 2.0)$

Fig. 12. (θ, v)-slices of the state space of $\mathcal{A}$. Shaded regions are ICS respectively defined for the braking trajectory of control $(\alpha_{min}, \dot{\xi}_{min})$ (top), and all braking trajectories with controls selected from $[(\alpha_{min}, \dot{\xi}_{max}), (\alpha_{min}, 0), (\alpha_{min}, \dot{\xi}_{min})]$ (bottom).

with $\alpha \in [\alpha_{min}, \alpha_{max}]$ (acceleration bounds), $\gamma \in [\gamma_{min}, \gamma_{max}]$ (steering velocity bounds), and $|\xi| \leq \xi_{max}$ (steering angle bounds). L is the wheelbase of $\mathcal{A}$.

For practical reasons, the duration of the trajectories of $\mathcal{I}$ has to be limited to a given *time horizon* that determines the overall level of safety of $\mathcal{A}$. In our case, the subset $\mathcal{I}$ considered in order to compute a conservative approximation of the set of ICS includes the braking trajectories with a constant control selected from $[(\alpha_{min}, \dot{\xi}_{max}), (\alpha_{min}, 0), (\alpha_{min}, \dot{\xi}_{min})]$, and applied over the time necessary for $\mathcal{A}$ to stop.

Fig. 12 depicts the ICS obtained when different set of braking trajectories are considered. Each subfigure represents

a (θ, v)-slice of the full 5D state space of $\mathcal{A}$. In the top subfigures, only the braking trajectory of control $(\alpha_{min}, \dot{\xi}_{min})$ is considered. In the bottom subfigures, the three braking trajectories are considered.

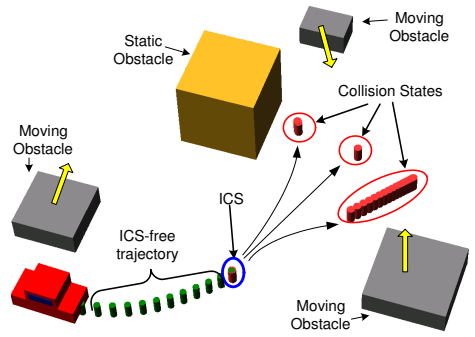


Fig. 13. The state labelled ICS is an ICS since the three braking trajectories issued from it yield collisions.

In PMP, checking whether a state is an ICS or not is carried out by testing if all the braking trajectories yield a collision with one of the moving obstacles. If so, the state is an ICS. In fig. 13), the collision states in red represent the collision that will occur in the future from this state for all trajectories of $\mathcal{I}$. In this case, since all trajectories collide in the future, this state is an ICS. In PMP, every new state is similarly checked to be an ICS or not over $\mathcal{I}$. In case all the trajectories appear to be in collision in the future, this state is an ICS and is not selected.

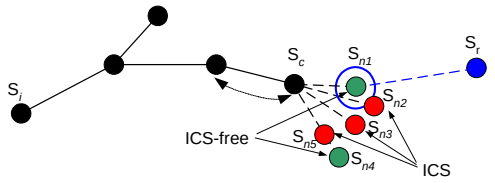


Fig. 14. Search tree construction principle.

The exploration method used is the well known Rapidly-Exploring Random Tree method (RRT) [46]. RRT incrementally builds a tree in the state space of $\mathcal{A}$. The basic principle of RRT is depicted in Fig. 14. A state s_r is randomly selected first. Then, the closest node in the tree, say s_c, is determined. Constant controls selected from $\mathcal{U}_{2D}$=[$(\alpha_{max}, 0)$; $(\alpha_{max}, \dot{\xi}_{max})$; $(\alpha_{max}, \dot{\xi}_{min})$;$(0, \dot{\xi}_{max})$; $(0, 0)$; $(0, \dot{\xi}_{min})$;$(\alpha_{min}, \dot{\xi}_{max})$; $(\alpha_{min}, 0)$; $(\alpha_{min}, \dot{\xi}_{min})$] are then applied to s_c for a duration ϵ, they yield a set of candidate trajectories ending in given states s_{ni}. These candidate trajectories are pruned out: only are kept the trajectories that are collision-free and whose final state is ICS-free (as per property 2, such trajectories are ICS-free). Finally, the trajectory whose final state is closer to s_r is selected and added up to the tree. This process is repeated until the end of the time available where the best partial trajectory extracted from the tree is returned.

In Fig. 15 we can see an example of a navigation from a still starting state (green box) to a still goal state (red box). The environment is cluttered with moving (grey) and static (orange) obstacles. In 15(a) one can observe how the safe partial trajectory (green) is calculated and planned within the time-state space in order to avoid the obstacle moving upward. The states in blue behind the car, is the trajectory, built from partial trajectories from the previous PMP cycles and (ideally) executed by the robot. In 15(b) we can observe that the car was obliged to slow down at the intersection of several obstacles, since no other safe trajectories could be found, before to re-accelerate. In 15(c) the system has planned a partial trajectory that avoids the last static obstacle.

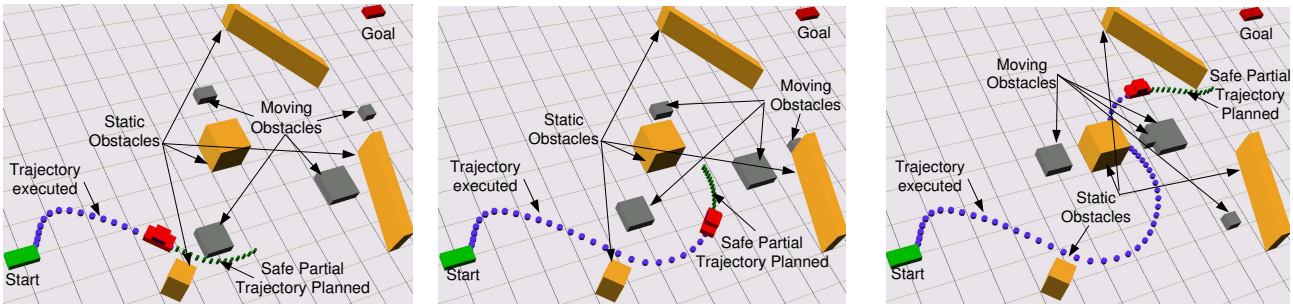
V. Conclusion

This paper addressed the problem of navigating safely in a open and dynamic environment sensed using both on-board and external sensors. After a short presentation of the context and of the related open problems, we focused on two complementary questions: how to interpret and to predict the motions and the behaviors of the sensed moving entities ? how to take appropriate goal-oriented navigation decisions in such a rapidly changing and sensed environment ?

In order to answer these questions, we have proposed an approach including three main complementary functions: (1) Scene interpretation and short-term motion prediction for the sensed potential obstacles, using the "Bayesian Occupancy Filtering" approach (BOF) (2) Medium-term motion and behavior prediction for the observed entities, using motion pattern learning and hierarchical Hidden Markov Models; (3) On-line goal-oriented navigation decision in a dynamic environment, using the "Partial Motion Planning" paradigm (PMP).

The first function (BOF)has experimentally been validated on our experimental vehicle (the Cycab), for avoiding partially observed moving obstacles. A scenario involving the Cycab, a moving pedestrian, and a parked car which temporarily hide the pedestrian to the sensors of the Cycab, has successfully been executed. In this experiment, the avoidance behavior has been obtained by combining the *occupancy probability* and the *danger probability* of each cell of the grid. The second function has experimentally been validated on some indoor data (the INRIA entry hall), using about 1000 tracked human trajectories for the initial learning phase. At the moment, the last function (PMP) has only been experimented in simulation.

Current work mainly deals with three major points: (1) Improvement of the prediction approaches for making it possible to cope with larger environments (such as complex urban traffic situations), while preserving the efficiency property; the current development on the WOG model is an example of this work. (2) Fusion of our current interpretation and prediction paradigms with higher-level information (e.g. GPS maps, moving entities properties, nominal behaviors ...) to better estimate the scene participants behaviors. (3) Integration of the three previous functions, and implementation and test this new navigation system on our experimental platform involving the INRIA car park, several Cycabs, and both inboard and infrastructure sensors.



(a) Avoidance of the first moving obstacle

(b) Braking in front of unavoidable moving obstacles

(c) Avoidance of the last static obstacle

Fig. 15. Results of a 2D safe Partial Motion Planning ($\delta_c = 1s$, $v_{max} = 2.0m/s$, $\xi_{max} = \pi/3rad$, $\dot{\xi}_{max} = 0.2rad/s$, $\alpha_{max} = 0.1m/s^2$)

Acknowledgements.: This work has been partially supported by several national and international projects and funding bodies: European projects *Carsense*, *CyberCars & PreVent/ProFusion*; French Predit projects *Puvame & MobiVip*. The authors would like to thanks Pierre Bessière, Christophe Coué and Cedric Pradalier for their contributions and fruitful discussions.

REFERENCES

[1] C. Coué, C. Pradalier, and Laugier C. Bayesian programming for multi-target tracking: an automotive application. In *Proceedings of the International Conference on Field and Service Robotics*, Lake Yamanaka (JP), July 2003.

[2] H.P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2), 1988.

[3] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer, Special Issue on Autonomous Intelligent Machines*, Juin 1989.

[4] S. Thrun. Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[5] E. Prassler, J. Scholz, and A. Elfes. Tracking multiple moving objects for real-time robot navigation. *Autonomous Robots*, 8(2), 2000.

[6] O. Lebeltel. *Programmation Bayésienne des Robots*. Thèse de doctorat, Institut National Polytechnique de Grenoble, Grenoble, France, Septembre 1999.

[7] O. Lebeltel, P. Bessière, J. Diard, and E. Mazer. Bayesian robot programming. *Autonomous Robots*, 16:49–79, 2004.

[8] C. Coué and P. Bessière. Chasing an elusive target with a mobile robot. In *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems*, Hawai (HI), 2001.

[9] A. H. Jazwinsky. *Stochastic Processes and Filtering Theory*. New York : Academic Press, 1970.

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

[11] G. Welch and G. Bishop. An introduction to the Kalman filter. available at http://www.cs.unc.edu/ welch/kalman/index.html.

[12] I. Daubechies. *Ten Lectures on Wavelets*. Number 61 in CBMS-NSF Series in Applied Mathematics. SIAM Publications, Philadelphia, 1992.

[13] Stéphane Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, San Diego, 1998.

[14] 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.

[15] Bruno Sinopoli, Mario Micheli, Gianluca Donato, and T. John Koo. Vision based navigation for an unmanned aerial vehicle. In *Proceedings of the International Conference on Robotics and Automation*, May 2001.

[16] Manuel Yguel, Olivier Aycard, and Christian Laugier. Internal report: Wavelet occupancy grids: a method for compact map building. Technical report, INRIA, 2005.

[17] C. Pradalier and S. Sekhavat. Simultaneous localization and mapping using the geometric projection filter and correspondence graph matching. *Advanced Robotics*, 2004. To appear.

[18] Maren Bennewitz, Wolfram Burgard, and Sebastian Thrun. Learning motion patterns of persons for mobile service robots. In *Proceedings of the IEEE Int. Conf. On Robotics and Automation*, pages 3601–3606, Washington, USA, 2002.

[19] Sarah Osentoski, Victoria Manfredi, and Sridhar Mahadevan. Learning hierarchical models of activity. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, 2004.

[20] E. Kruse, R. Gusche, and F. M. Wahl. Acquisition of statistical motion patterns in dynamic environments and their application to mobile robot motion planning. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 713–717, Grenoble, France, 1997.

[21] Dizan Vasquez and Thierry Fraichard. Motion prediction for moving objects: a statistical approach. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 3931–3936, New Orleans, LA (US), apr 2004.

[22] H. Bui, S. Venkatesh, and G. West. Policy recognition in the abstract hidden markov models. *Journal of Artificial Intelligence Research*, 17:451–499, 2002.

[23] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition*, pages 267–296, 1990.

[24] Stephen Marsland, Jonathan Shapiro, and Ulrich Nehmzow. A self-organizing network that grows when required. *Neural Networks*, 2002.

[25] Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13(2):260–269, April 1967.

[26] Y. Linde, A. Buzo, and R.M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, COM-28:84–95, 1980.

[27] Teuvo Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 1995. (Second Extended Edition 1997).

[28] M. Martinetz and K. J. Schulten. A "neural-gas" network learns topologies. In T. Kohonen, K. M??kisara, O. Simula, and editors J. Kangas, editors, *Proceedings of International Conference on Articial Neural Networks*, volume I, pages 397–402, North-Holland, Amsterdam, 1991.

[29] G.A. Carpenter, S. Grossberg, and D.B. Rosen. Fuzzy art: An adaptive resonance algorithm for rapid, stable classification of analog patterns. In *Proc. Int. Joint Conf. Neural Networks*, volume II, pages 411–420, Seattle, USA, 1991.

[30] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD Conference*, pages 47–57, 1984.

[31] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, feb 2002.

[32] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic robotics. Unpublished.

[33] J. H. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *Proc. of the IEEE Symp. on the Foundations of Computer Science*, pages 144–154, Portland, OR (US), October 1985.

[34] R.C. Arkin. Motor schema-based mobile robot navigation. *International Journal of Robotics Research*, 8(4):92–112, August 1989.
[35] J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3):278–288, June 1991.
[36] R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1080–1087, 1995.
[37] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. Technical Report IAI-TR-95-13, 1 1995.
[38] M. Khatib. *Sensor-based motion control for mobile robots*. PhD thesis, LAAS-CNRS December, 1996, 1996.
[39] R. Simmons. The curvature velocity method for local obstacle avoidance. In *Proceedings of the International Conference on Robotics and Automation*, pages 3375–3382, Minneapolis (USA), april 1996.
[40] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17(7):760–772, July 1998.
[41] N.Y. Ko and R. Simmons. The lane-curvature method for local obstacle avoidance. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Victoria (Canada), October 1998.
[42] J. Minguez and L. Montano. Nearness diagram navigation (ND): A new real time collision avoidance approach for holonomic and no holonomic mobile robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Takamatsu, Japan, November 2000.
[43] N. Roy and S. Thrun. Motion planning through policy search. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, 2002.
[44] J. Minguez, L. Montano, and J. Santos-Victor. Reactive navigation for non-holonomic robots using the ego kinematic space. In *Proceedings IEEE International Conference on Robotics and Automation*, Washington (US), May 2002.
[45] Thierry Fraichard and Hajime Asama. Inevitable collision states - a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.
[46] S. LaValle and J. Kuffner. Randomized kinodynamic planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 473–479, Detroit (US), May 1999.