Hierarchies of probabilistic models of navigation: the Bayesian Map and the Abstraction operator

Julien Diard, Pierre Bessiere, Emmanuel Mazer

Proceedings of the 2004 IEEE
International Conference on Robotics & Automation
New Orleans, LA • April 2004

Hierarchies of probabilistic models of navigation: the Bayesian Map and the Abstraction operator

Julien Diard, Pierre Bessière and Emmanuel Mazer
Laboratoire GRAVIR / IMAG – CNRS
INRIA Rhône-Alpes, 655 avenue de l'Europe
38330 Montbonnot Saint Martin FRANCE
Julien.Diard@free.fr

Abstract— **This paper presents a new method for probabilistic modeling of space, called the *Bayesian Map* formalism. It offers a generalization of some common approaches found in the literature, as it does not constrain the dependency structure of the probabilistic model. The formalism allows incremental building of hierarchies of models, by the use of the *Abstraction* operator. In the resulting hierarchy, localization in the high level model is based on probabilistic competition of the lower level models. Experimental results validate the concept, and hint at its usefulness for large scale scenarios.**

I. Introduction

In robotics, modeling the environment that a robot has to face in a navigation task is a crucial problem, that has received a lot of attention in the community. The most promising approaches rely on the probability calculus, especially for its capacity to handle incomplete models and uncertain information. These approaches include – but are far from limited to – Kalman Filters [1], Markov Localization models [2], (Partially or Fully) Observable Markov Decision Processes [3], and Hidden Markov Models [4]. We will here assume that the reader has some familiarity with these approaches.

In this domain of probabilistic modeling for robotics, hierarchical solutions are currently flourishing. The more active domain in this regard is decision theoretic planning: one can find variants of MDPs that accommodate hierarchies or that select automatically the partition of the state-space (see for instance [5], [6], or browse through the references in [7]). More exceptionally, one can find hierarchical POMDPs, as in [7], which is arguably the work that bears the most resemblance to the one presented here, although we do not use reward functions in this work. The current work can also be related to Thrun's object mapping paradigm [8], in particular concerning the aim of transferring some of the knowledge the programmer has about the task, to the robot. Some hierarchical approaches outside of the MDP community include Hierarchical HMMs and their variants (see [9] and references therein), which, unfortunately, rely on the notion of final state of the automata. Another class of approaches relies on the extraction of a graph from a probabilistic model, like for example a Markov Localization model [10], or a MDP [11]. Using such deterministic notions is inconvenient in a purely probabilistic approach, as we are pursuing here. Indeed, the current work uses probabilities in all layers of the hierarchy of representations, allowing us to propagate and handle uncertainties in a uniform and formally coherent manner.

Moreover, the main philosophy used by all the previous approaches is to try to extract, from a very complex but intractable model, a hierarchy of smaller models. Of course, *automatically* selecting the relevant decomposition of a problem into sub-problems is quite a challenge – this challenge being far from restricted to the domain of navigation for robots facing uncertainties.

We pursue here an alternate route, investigating how, starting from a set of simple models, one can combine them for building more complex models. The goal of this paper is therefore to present a new formalism for building models of the space in which a robot has to navigate (the *Bayesian Map* model), and a method for combining such maps together in a hierarchical manner (the *Abstraction operator*).

This formalism allows for a new representation of space, in which the final program is built upon many imbricate models, each of them deeply rooted into lower level sensorimotor relationships. Such hierarchies of sensorimotor models seem relevant to biologically inspired models, as it appears that no single metric model can account alone for large scale navigation capacities of animals (see [12], [13]).

We will also argue that our approach draws away from the usual characteristics of the common models of space (Section II-C), and that it is also more general than these models (Sections III and VI). For brevity, this paper will discuss neither of the learning methods that can be included into Bayesian Maps (mapping process), nor of another operator for merging Bayesian Maps (the Superposition operator). Preliminary work about these issues and all the details missing in the current paper can be found in Diard's Ph.D. thesis [14].

The rest of this paper is organized as follows. Section II presents the Bayesian Robot Programming methodology, and discusses some of its characteristics. Sections III and IV will quickly define our notion of Bayesian Map, and the Abstraction operator, respectively. The paper concludes on the presentation of experimental results, Section V.

II. Bayesian Robot Programming

The work we present here is based on BRP, a Bayesian Robot Programming methodology. We summarize it here, but

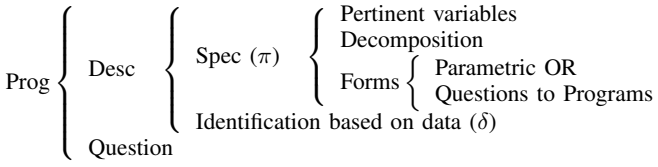

Fig. 1. Structure of a Bayesian Robotic Program.

still invite the reader to refer to [15] for all the details.

A. Definition

In the BRP formalism, a bayesian robotic program is a structure (see Fig. 1) made of two components.

The first is a *declarative* component, where the user defines a **description**. The purpose of a description is to specify a method to compute a joint distribution over a set of relevant variables $\{X_1, X_2, \ldots, X_n\}$, given a set of experimental data δ and preliminary knowledge π. This joint distribution is denoted $P(X_1\ X_2\ \ldots\ X_n \mid \delta\ \pi)$. To specify this distribution, the programmer first lists the pertinent variables (and defines their domains), then decomposes the joint distribution as a product of simpler terms (possibly stating conditional independence hypotheses so as to simplify the model and/or the computations), and finally, assigns forms to each term of the selected product (these forms can be parametric forms, or recursive questions to other bayesian programs). If there are free parameters in the parametric forms, they have to be assessed. They can be given by the programmer (*a priori* programming) or computed on the basis of a learning mechanism defined by the programmer and some experimental data δ.

The second component is of a *procedural* nature, and consists of using the previously defined description with a **question**, *i.e.* computing a probability distribution of the form $P(Searched \mid Known)$. Answering a "question" consists in deciding a value for the variable $Searched$ according to $P(Searched \mid Known)$. Different decision policies are possible, in our robotic experiments we usually choose to draw a value at random according to that distribution. It is well known that general Bayesian inference is a very difficult problem, which may be practically intractable. But, as this paper is mainly concerned with modeling issues, we will assume that the inference problems are solved and implemented in an efficient manner by the programmer.

B. Example

Since the BRP formalism is only based on the inference rules needed for probability calculus, it is very general. Indeed, a very wide class of probabilistic models found in the literature can be rewritten as BRP programs, as is shown in [16]. For example, we can rewrite the Markov Localization model into the BRP formalism. The ML model is basically a Hidden Markov Model with an additional action variable. Recall that a HMM is basically the decomposition $P(O_t\ S_t\ S_{t-1}) = P(S_{t-1})P(S_t \mid S_{t-1})P(O_t \mid S_t)$, where O_t is a perception variable, S_t and S_{t-1} are location variables at time t and $t-1$. Starting from this structure, the action variable A_t is used to refine the transition model $P(S_t \mid S_{t-1})$ into $P(S_t \mid A_t\ S_{t-1})$. The resulting BRP model for Markov Localization is shown Fig. 2.

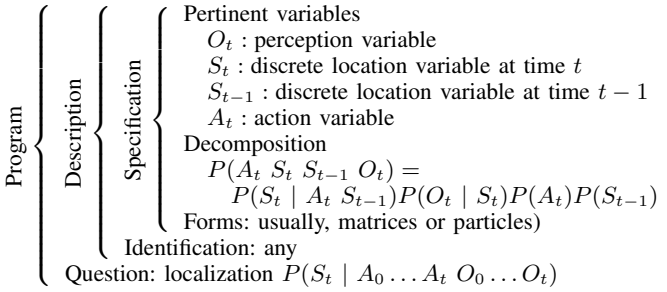

Fig. 2. The Markov Localization definition expressed in the BRP formalism.

C. BRP vs. other models

Let us now develop some remarks that arise from the comparison between the use of the BRP formalism and some aspects of the more common models of the representation of space (see Section I). In particular, we now focus on solving navigation tasks using BRP programs.

The first remark relies on the fact that, in BRP, a form appearing in a description c^1 can be a question to another description c^2. This allows the programmer to decompose a robotic program into sub-programs, as in structured computer programming. Therefore, the first step for solving a navigation task is to imagine, or to copy from living beings (see [12], [13]), intermediary levels of descriptions or skills, that are relevant. This is somewhat different from most probabilistic models of space, that only rely on one level of description, *i.e.* that try to represent the environment using only one type of features. Forms being questions to other descriptions is a key feature of our Abstraction operator (see Section IV).

The second remark is that the first step when designing a BRP description is the choice of variables. When dealing with the representation of space, one usually selects a perception variable, an action variable, and a location (or state) variable. Therefore, the programmer has to choose a set of locations that are *relevant for solving the task at hand, in the class of environments the robot will likely face*. The choice of the nature of these locations (metric or topologic, or dense or sparse, for instance) should come *as a consequence of these considerations*. This, again, somewhat differs from existing approaches, where the choice of model (Markov Localization or Kalman Filter, for instance), is rather a choice of a *dependency structure* or *form definition*, that implies properties on the choice of variables (Kalman Filters are well suited to continuous variables, for instance). In contrast, in the Bayesian Map formalism, we will not put constraints on the choice of decomposition or forms: the programmer will have all latitude left for choosing the semantic of the location variable that

solves his navigation task (the constraints on the choice of variables will merely be syntactic).

The third and final remark is that, in BRP, the description phase is considered independent of the utilization phase. This contrasts with most probabilistic models, where the terms appearing in the decomposition are usually chosen for a particular inference. For example, action or transition models, which can be difficult to assess when the variables are not chosen well, are still very common because they are easily integrated into the location estimation. In our Bayesian Map formalism, we will constraint what maps are used for (the questions), but not how the knowledge necessary for using the map is structured (the decomposition).

III. Bayesian Maps

A Bayesian Map c is a description that defines a joint distribution $P(P\ L_t\ L_{t'}\ A \mid c)$, where:

- P is a perception variable (the robot reads its values from physical sensors or lower level variables),
- L_t is a *location* variable at time t,
- $L_{t'}$ is a variable having the same domain than L_t, but at time t' (without loss of generality, let us assume $t' > t$),
- and A is an action variable (the robot writes commands on this variable).

For simplicity, we will assume here that all these variables have finite domains.

The choice of decomposition is not constrained: any probabilistic dependency structure can therefore be chosen here. Finally, the definition of forms and the learning mechanism (if any) are not constrained, either.

For a Bayesian Map to be useable in practice, we need the description to be rich enough to generate *behaviors*. We call *elementary behavior* any question of the form $P(A^i \mid X)$, where A^i is a subset of A, and X a subset of the other variables of the map (*i.e.*, not in A^i). A behavior can be not elementary, for example if it is a sequence of elementary behaviors, or, in more general terms, if it is based on elementary behaviors and some other knowledge (which need not be expressed in terms of maps).

For a Bayesian Map to be interesting, we will also require that it generates *several* behaviors – otherwise, defining just a single behavior instead of a map is enough. Such a map is therefore a resource, based on a location variable relevant enough to solve a class of tasks: this internal model of the world can be reified.

A "guide" one can use to "make sure" that a given map will generate useful behaviors, is to check if the map answers in a relevant manner the three questions $P(L_t \mid P)$ (localization), $P(L_{t'} \mid A\ L_t)$ (prediction) and $P(A \mid L_t\ L_{t'})$ (control).

By "relevant manner", we mean that these distributions have to be informative, in the sense that their entropy is "far enough" of its maximum (*i. e.* the distribution is different from a uniform distribution). This constraint is not formally well defined, but it seems intuitive to focus on these three questions. Indeed, the skills of localization, prediction and control are well identified in the literature as means to generate behaviors.

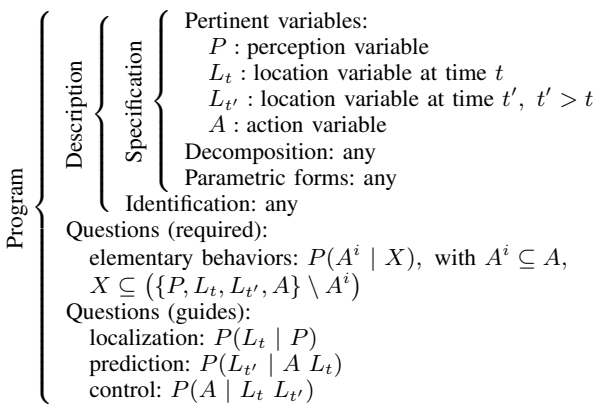

Fig. 3. The Bayesian Map model definition expressed in the BRP formalism.

Checking that the answers to these questions are informative is a first step to evaluate the quality of a Bayesian Map with respect to solving a given task.

Fig. 3 is a summary of the definition of the Bayesian Map formalism.

A. Generality of the Bayesian Map formalism

We now invite the reader to verify that the Markov Localization model is indeed a special case of the Bayesian Map model by comparing Fig. 2 and Fig. 3. Recall that Kalman Filters and Particle Filters are special cases of Markov Localization, as they add hypotheses over the choice of dependency structure made by the Markov Localization model. This implies that Kalman Filters and Particle Filters also are special cases of Bayesian Maps.

Bayesian Maps can therefore accommodate many different forms, depending on the needs or information at hand: for example, one Bayesian Map can be structured like a real valued Kalman Filter for tracking the angle and distance to some feature when it is available. If that feature is not present, or in cases where the linearity hypotheses fail, we can use another Bayesian Map, which need not be a Kalman Filter (for example, based on a symbolic variable).

IV. Abstraction of Bayesian Maps

Having defined the Bayesian Map concept, we now turn to defining operators for putting Bayesian Maps together. The one we present here is called the Abstraction of maps, it is defined Fig. 4, and commented in the rest of this section.

As stressed above, in a Bayesian Map, the semantics of the location variable can be very diverse. The main idea behind the abstraction operator is to build *a Bayesian Map c whose different locations are other Bayesian Maps* $c^1, c^2, \ldots, c^n$. The location variable of the abstract map will therefore take n possible symbolic values, one for each underlying map c^i. Each of these maps will be "nested" in the higher level abstract map, which justifies the use of the term "hierarchy" in our work. Recall that Bayesian Maps are designed for generating behaviors. Let us note $a^1, a^2, \ldots, a^k$ the k behaviors defined

Program

Description

Specification

Pertinent variables:

$P = \bigwedge_{i=1}^{n} \left(P^i \wedge L_t^i \wedge L_{t'}^i \wedge A^i\right)$

$L_t : \mathcal{D}_{L_t} = \{c^1, c^2, \ldots, c^n\}, k_{L_t} = n$

$L_{t'} : \mathcal{D}_{L_{t'}} = \{c^1, c^2, \ldots, c^n\}, k_{L_{t'}} = n$

$A : \mathcal{D}_A = \{a_1, a_2, \ldots, a_k\}, k_A = k$

Decomposition:

$P(P\ L_t\ L_{t'}\ A) =$

$P(L_t) \prod_{i=1}^{n} P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid L_t)$

$P(L_{t'}) P(A \mid L_t\ L_{t'})$

Parametric forms:

$P(L_t) =$ Uniform

$P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid [L_t = c]) = \begin{cases} \text{if } c = c^i \text{ then } P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid c^i) \\ \text{else Uniform} \end{cases}$

$P(L_{t'}) =$ Uniform

$P(A \mid L_t\ L_{t'}) =$ Table

Identification:

a priori programming or learning of $P(A \mid L_t\ L_{t'})$

Questions:

$$\begin{aligned} P(L_t \mid P) &= \tfrac{1}{Z_1} \prod_{i=1}^{n} P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid L_t) \\ P(L_{t'} \mid A\ L_t) &= \tfrac{1}{Z_2} P(A \mid L_t\ L_{t'}) \\ P(A \mid L_t\ L_{t'}) &= P(A \mid L_t\ L_{t'}). \end{aligned}$$

Fig. 4. The abstraction operator definition expressed as a Bayesian Map.

in the n underlying maps. In the abstract map, these behaviors can be used for linking the locations c^i. The action variable of the abstract map will therefore take k possible symbolic values, one for each behavior of the underlying maps. In order to build an abstract map having n locations, the programmer will have to have previously defined n lower level maps, which generate k behaviors. The numbers n and k are therefore small, and so the abstract map deals with a small internal space, having retained of each underlying map only a symbol, and having "forgotten" all their details. This justifies the use of the name "abstraction" for this operator. But this "summary mechanism" has yet to be described: that is what the perception variable P of the abstract map will be used for, as it will be the list of all the variables appearing in the underlying maps: $P = P^1, L_t^1, L_{t'}^1, A^1, \ldots, P^n, L_t^n, L_{t'}^n, A^n$.

Given the four variables of the abstract map, we define its joint distribution with the following decomposition:

$$\begin{aligned} &P(P\ L_t\ L_{t'}\ A) \\ &= P(L_t) \prod_{i=1}^{n} P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid L_t) P(L_{t'}) P(A \mid L_t\ L_{t'}). \end{aligned}$$

In this decomposition, $P(L_t)$ and $P(L_{t'})$ are defined as uniform distributions. All the terms of the form $P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid [L_t = c])$ are defined as follows: when $c \neq c^i$, the probabilistic dependency between the variables P^i, L_t^i, $L_{t'}^i$, A^i of the map c^i is supposed unknown, therefore defined by a uniform distribution. Whereas when $c = c^i$, this dependency is exactly what the map c^i defines. Therefore this term is a question to the description c^i, but a question that includes the whole sub-description by asking for the joint distribution it defines. Since the last term, $P(A \mid L_t\ L_{t'})$, only includes symbolic variables that have a small number of values, it makes sense to define it as a table, which can be easily *a priori* programmed or learned experimentally.

The abstract Bayesian Map is now fully defined, and, given n underlying maps, can be automatically built. The last step is to verify that it generates useful behaviors. We will examine the guide questions of localization, prediction and control.

The localization question leads to the following inference (derivation omitted): $P(L_t \mid P) \propto \prod_{i=1}^{n} P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid L_t)$. The interpretation of this result will be explained with an example, Section V. The derivations for solving the prediction $P(L_{t'} \mid A\ L_t)$ and control $P(A \mid L_t\ L_{t'})$ questions are also straightforward, and given Fig. 4.

Recall that the final goal of any Bayesian Map is to provide behaviors. In the abstract map, this is done by answering a question like $P(A \mid [L_{t'} = c^i]\ [P = p])$: what is the probability distribution over lower level behaviors, knowing all values p of the variables of the lower level, and knowing that we want to "go to map c^i?" Answering this question thus allows selecting the most relevant underlying behavior to reach a given high level goal. The computation is as follows:

$$\begin{aligned} &P(A \mid L_{t'}\ P) \\ &= \frac{1}{Z} \sum_{L_t} \left(\prod_{i=1}^{n} P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid L_t) \right) P(A \mid L_t\ L_{t'}). \end{aligned}$$

This computation includes the localization question, to weigh the probabilities given by the control model $P(A \mid L_t\ L_{t'})$. In other words, the distribution over the action variable A includes all localization uncertainties. Each underlying model is used, even when the robot is located at a physical location that this model is not made for. As a direct consequence, there is no need to *decide* what map the robot is in, or to *switch* from map to map: the computation considers all possibilities and weighs them according to their (localization) probabilities. Therefore the underlying maps need not be "mutually exclusive" in a geographical sense.

V. Experimental validation

We report here an experiment made on the well-known Koala mobile robot platform (K-team company). In order to keep as much control as possible over our experiments and the different effects we observe, we simplify the sensorimotor system and its environment. We only use the 16 proximeters $Px = Px_0 \wedge \ldots \wedge Px_{15}$ of our robot, and keep two degrees of freedom of motor control, via the rotation and translation speed $Vrot$ and $Vtrans$. The environment we use is a 5 m $\times$ 5 m area made of movable planks (see a typical configuration we use Fig. 5). The goal of this experiment is to solve a navigation task: we want the robot to be able to go hide in any corner, as if the empty space in the middle of the area were dangerous.

The first programming step is to analyze this task into sub-tasks. We particularize three situations that are relevant for solving the task: the robot can either be near a wall, and it should follow it in order to reach the nearest corner, or the robot can be in a corner, and it should stop, or finally it could

be in empty space, and should therefore go straight, so as to leave the exposed area as quickly as possible.

A. Low level Bayesian Maps

Given this analysis, the second programming step is to define one Bayesian Map for each of the three situations. They all use the same perception variable $P = Px$ and the same action variable $A = Vrot \wedge Vtrans$.

The first map, c^{wall}, describes how to navigate in presence of a single wall, using a location variable $L_t = \theta \wedge Dist$: the phenomenon "wall" is summed up by an angle θ and a distance $Dist$. Therefore, c^{wall} defines $P(Px\ \theta_t\ Dist_t\ \theta_{t'}\ Dist_{t'}\ Vrot\ Vtrans \mid c^{wall})$. We have implemented this map using 12 possible angle values, and 3 different distances. This lead to a compact model, yet accurate enough to solve the sub-tasks we wanted to solve. The dependency structure we choose is (c^{wall} on right hand sides omitted):

$$\begin{aligned} & P(Px\ \theta_t\ Dist_t\ \theta_{t'}\ Dist_{t'}\ Vrot\ Vtrans) \\ = \quad & P(\theta_t\ Dist_t) \prod_i P(Px_i \mid \theta_t\ Dist_t) P(\theta_{t'}\ Dist_{t'}) \\ & P(Vrot \mid \theta_t\ Dist_t\ \theta_{t'}\ Dist_{t'}) \\ & P(Vtrans \mid \theta_t\ Dist_t\ \theta_{t'}\ Dist_{t'}). \end{aligned}$$

$P(\theta_t\ Dist_t)$ and $P(\theta_{t'}\ Dist_{t'})$ are uniform probability distributions. Each term of the form $P(Px_i \mid \theta_t\ Dist_t)$ is a set of Gaussians, that were identified experimentally, by a supervised learning phase: we physically put the robot in all 36 possible situations, and recorded proximeter values so as to compute experimental means and standard deviations. Finally, the two control terms $P(Vrot \mid \theta_t\ Dist_t\ \theta_{t'}\ Dist_{t'})$ and $P(Vtrans \mid \theta_t\ Dist_t\ \theta_{t'}\ Dist_{t'})$ were programmed "by hand": given the current angle and distance, and the angle and distance to be reached, what should be the motor commands?

This map successfully solves navigation tasks like "follow-wall-right", "follow-wall-left", "go-away-from-wall", "stop", using behaviors of the same name. For example, "follow-wall-right" is defined by the probabilistic question $P(Vrot\ Vtrans \mid Px\ [L_{t'} = \langle 90, 1 \rangle])$: compute the distribution on motor variables knowing the sensory input and knowing that the location to reach is $\theta = 90\ ^\circ$, $Dist = 1$ (wall on the right at medium distance).

This map is an instance where a Kalman Filter based Bayesian Map could have been used instead: for example, if we had required more accuracy on the angle and distance to the wall, using continuous variables. The coarse grained set of values we used were actually sufficient for our experiments.

The two other Bayesian Maps we define are the following. 1) c^{corner} describes how to navigate in a corner, using a symbolic location variable that can take 4 values: $FrontLeft$, $FrontRight$, $RearLeft$ and $RearRight$. This is enough for solving tasks like "quit-corner-and-follow-right", "away-from-both-walls", "stop". 2) $c^{empty-space}$describes how to navigate in empty space, *i.e.* when the sensors do not see anything. The behaviors defined here are "straight-ahead" and "stop".

B. Abstract Bayesian Map

Given these three maps, the third and final programming step is to apply the abstraction operator on them. We obtain a map c, whose location variable is $L_t = \{c^{wall}, c^{corner}, c^{empty-space}\}$. The action variable lists the behaviors defined by the low level maps: $A = \{\text{follow-wall-right}, \text{go-away-from-wall}, \ldots\}$. The rest of the abstract map is according to the schema of Fig. 4.

We want here to discuss the localization question. Let us assume that the robot is in empty space: all its sensors read 0. Let us also assume that the robot is currently applying the "straight-ahead" behavior, that sets $Vrot$ and $Vtrans$ near 0 (no rotation) and 40 (fast forward movement), respectively, using sharp Gaussian distributions.

Let us consider the probability to be in location $c^{empty-space}$ (with w standing for $wall$, c for $corner$ and e for $empty - space$):

$$\begin{aligned} & P([L_t = c^{empty-space}] \mid P) \\ & \propto \begin{pmatrix} P(P^w\ L_t^w\ L_{t'}^w\ A^w \mid [L_t = c^{empty-space}]) \\ P(P^c\ L_t^c\ L_{t'}^c\ A^c \mid [L_t = c^{empty-space}]) \\ P(P^e\ L_t^e\ L_{t'}^e\ A^e \mid [L_t = c^{empty-space}]) \end{pmatrix}. \end{aligned}$$

Of the three terms of the product, two are uniforms, and one is the joint distribution given by $c^{empty-space}$. That joint distribution gives a very high probability for the current situation, as describing the phenomenon "going straight ahead in empty space" basically amounts to favoring sensory readings of 0 and motor commands near 0 and 40 for $Vrot$ and $Vtrans$, respectively. The situation is quite the opposite for $P([L_t = c^{wall}] \mid P)$: for example, c^{wall} does not favor at all this sensory situation. Indeed, the phenomenon "I am near a wall" is closely related to the fact that the sensors actually sense something. The probability of seeing nothing on the sensors knowing that the robot is near a wall is very low: $P([L_t = c^{wall}] \mid P)$ will be very low. The reasoning is similar for $P([L_t = c^{corner}] \mid P)$.

This computation can thus be interpreted as the *recognition of the most pertinent underlying map* for a given sensorimotor situation. Alternatively, it can be seen as a *measure of the coherence of the values of the variables* of each underlying map, or even as a *Bayesian comparison of the relevance of models*, as assessed by the numerical value of the joint distributions of each lower level model. Since these distributions include (lower level) location and action variables, the maps are not only recognized by sensory patterns, but also by what the robot is currently doing.

The localization question can therefore be used to assess the "validity zones" of the underlying maps, *i.e.* the places of the environment where the hypotheses of each model hold. Experimentally, we have the robot navigate in the environment, and ask at each time step the localization question. We can summarize visually the answer, for example by drawing values for L_t, and report the drawn value on a Cartesian map of the environment. A (simplified but readable) result is shown Fig. 5. As can be seen, the robot correctly recognizes each

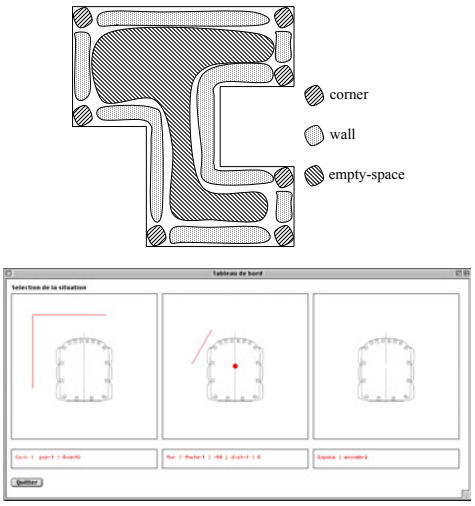

Fig. 5. 2D projection of the estimated "validity zones" of the maps c^{wall}, c^{corner} et $c^{empty-space}$. The bottom part of the figure is a screenshot of the localization module of the abstract map: it shows the "comparison" and competition between the underlying models. The winner is marked by the central dot: in this case, the robot was near a wall.

situation that it has a model for. Let us note that the resulting zones are not contiguous in the environment: for example, all the corners of the environment are associated with the same symbol, namely, c^{corner}. This effect is known as *perceptual aliasing*. But this very simple representation is sufficient for solving the task that was given to the robot: we report here that the behavior "go-hide-in-any-corner" is indeed generated by the abstract map.

A typical trajectory for the robot, starting from the middle of the arena, is to start by going straight ahead. As soon as a couple of forward sensors sense something, the "empty-space" situation is not relevant anymore, and the robot applies the best model it has, depending on the correlation between what the sensors see: if it looks like a wall and moves like a wall, then the probability for the "wall" model is high; on the other hand, if it rather feels like a corner, then the corner model wins the probabilistic competition. Suppose it was near a wall, then it starts to follow it, until a corner is reached. In our first version, the corner model was designed "too independently" of the wall model: the validity zone of the c^{corner} map was too small, and seldom visited by the robot as it passed the corner using the "follow-wall-right" behavior, defined by c^{wall}. The robot would then miss the first corner, and stop at another one. This shows that the decomposition of the task gives independent sub-tasks only as a first approximation. We solved the problem by modifying the "corner" model, so that it would recognize a corner on a typical "follow-wall-right" trajectory.

VI. Conclusion

We have presented the Bayesian Map formalism: it is a generalization of most probabilistic models of space found in the literature. Indeed, it drops the usual constraints on the choice of decomposition, forms, or implementation of the probability distributions. We have also presented the Abstraction operator, for building hierarchies of Bayesian Maps.

The experiments we presented are of course to be regarded only as "proofs of concept". Their simplicity also served didactic purposes. However, these experiments, in our view, are a successful preliminary step toward applying our formalism. Part of the current work is of course aimed at enriching these experiments, in particular with respect to the *scaling up* capacity of the formalism.

Moreover, since each map of the hierarchy is a full probabilistic model it is potentially very rich. Possible computations based on these maps include questions like the prediction question $P(L_{t'} \mid A\ L_t)$, which can form the basis of *planning* processes. Hierarchies of Bayesian Maps are therefore to be placed alongside model based approaches, instead of pure reactive approaches. Exploiting such knowledge by integrating a planning process in our Bayesian Map formalism is also part of the ongoing work.

References

[1] J. Leonard, H. Durrant-Whyte, and I. Cox, "Dynamic map-building for an autonomous mobile robot," *The International Journal of Robotics Research*, vol. 11, no. 4, pp. 286–298, 1992.

[2] S. Thrun, "Probabilistic algorithms in robotics," *AI Magazine*, vol. 21, no. 4, pp. 93–109, 2000.

[3] L. Kaelbling, M. Littman, and A. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[4] L. R. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Englewood Cliffs, New Jersey: Prentice Hall, 1993, ch. Theory and implementation of Hidden Markov Models, pp. 321–389.

[5] M. Hauskrecht, N. Meuleau, L. P. Kaelbling, T. Dean, and C. Boutilier, "Hierarchical solution of Markov decision processes using macro-actions," in *Proceedings of the 14th Conf. on Uncertainty in Artificial Intelligence (UAI-98)*, G. F. Cooper and S. Moral, Eds. San Francisco: Morgan Kaufmann, July, 24–26 1998, pp. 220–229.

[6] T. Lane and L. P. Kaelbling, "Toward hierarchical decomposition for planning in uncertain environments," in *Proceedings of the 2001 IJCAI Workshop on Planning under Uncertainty and Incomplete Information*. Seattle, WA: AAAI Press, August 2001.

[7] J. Pineau and S. Thrun, "An integrated approach to hierarchy and abstraction for POMDPs," Carnegie Mellon University, Technical Report CMU-RI-TR-02-21, August 2002.

[8] S. Thrun, "Robotic mapping: A survey," Carnegie Mellon University, Technical Report CMU-CS-02-111, February 2002.

[9] K. Murphy, "Dynamic bayesian networks: Representation, inference and learning," Ph.D. thesis, University of California, Berkeley, Berkeley, CA, July 2002.

[10] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, no. 1, pp. 21–71, 1998.

[11] T. Lane and L. P. Kaelbling, "Nearly deterministic abstractions of markov decision processes," in *18th Nat. Conf. on Artificial Intelligence*, 2002.

[12] B. J. Kuipers, "The spatial semantic hierarchy," *Artificial Intelligence*, vol. 119, no. 1–2, pp. 191–233, 2000.

[13] O. Trullier, S. Wiener, A. Berthoz, and J.-A. Meyer, "Biologically-based artificial navigation systems: Review and prospects," *Progress in Neurobiology*, vol. 51, pp. 483–544, 1997.

[14] J. Diard, "La carte bayésienne – un modèle probabiliste hiérarchique pour la navigation en robotique mobile," Thèse de doctorat, Institut National Polytechnique de Grenoble, Grenoble, France, Janvier 2003.

[15] O. Lebeltel, P. Bessière, J. Diard, and E. Mazer, "Bayesian robot programming," *Autonomous Robots (in press)*, vol. 16, no. 1, 2004.

[16] J. Diard, P. Bessière, and E. Mazer, "A survey of probabilistic models, using the bayesian programming methodology as a unifying framework," in *The Second Int. Conf. on Computational Intelligence, Robotics and Autonomous Systems (CIRAS)*, Singapore, December 2003.