

A Bayesian framework for robotic programming

O. Lebeltel*, J. Diard*, P. Bessière* and E. Mazer†

**Laboratoire LEIBNIZ - CNRS
46, avenue Félix Viallet, 38031 Grenoble, FRANCE*

*†Laboratoire GRAVIR - CNRS
INRIA Rhône-Alpes, ZIRST 38030 Montbonnot, FRANCE*

Abstract. We propose an original method for programming robots based on bayesian inference and learning. This method formally deals with problems of uncertainty and incomplete information that are inherent to the field. Indeed, the principal difficulties of robot programming comes from the unavoidable incompleteness of the models used. We present the formalism for describing a robotic task as well as the resolution methods. This formalism is inspired by the theory of probability, suggested by the physicist E T Jaynes: “Probability as Logic”[1]. Learning and maximum entropy principle translate incompleteness into uncertainty. Bayesian inference offers a formal framework for reasoning with this uncertainty. The main contribution of this paper is the definition of a generic system of robotic programming and its experimental application. We illustrate it by programming a surveillance task with a mobile robot: the Khepera. In order to do this, we use generic programming resources called “descriptions”. We show how to define and use these resources in an incremental way (reactive behaviors, sensor fusion, situation recognition and sequences of behaviors) within a systematic and unified framework.

INTRODUCTION

Anyone who ever had to program a real robot in a physical environment eventually had to face problems due to uncertainties. Sensor values are “noisy”, motor commands consequences are never quite the ones expected, models are “erroneous”... In robotics, dealing with uncertainties is inevitable.

There is quite a lot of experimental work involving programming robots to act under uncertainty, based on Bayesian inference. In robotics, the uncertainty topic is either related to calibration [2] or to planning problems [3]. Bayesian techniques are used in POMDP (Partially Observable Markov Decision Processes) to plan complex paths in partially known environments [4], in BDA (Bayesian Decision Analysis) for sensor planning problem [5]. HMM (Hidden Markov Models) and Monte Carlo methods are used for localization, planning of complex tasks and recognizing situations [6, 7, 8, 9]. These works effectively use the Bayesian approach for accomplishing robot tasks, but they do not present a structured programming paradigm as the current paper does.

The paper is organized as follows. Section 2 deals with basic definitions and notations. Section 3 presents our method for robotic programming using a very simple example. Section 4 shows various instances of bayesian programs: simple reactive behaviours: instances of behaviour combinations: sensor fusion: and a combination of all these programs to achieve a patrolling task. Finally, we conclude with a discussion summing up the principles of our programming method. More details on this approach can be

found in [10, 11].

BASIC CONCEPTS

Following the works of Cox [12] and Jaynes [1], we base our inference on two basic rules:

- The *conjunction rule*, which gives the probability of a conjunction of propositions:

$$P(a \wedge b|\pi) = P(a|\pi)P(b|a\pi) = P(b|\pi)P(a|b\pi). \quad (1)$$

- The *normalization rule*, which states that the sum of the probabilities of a and $\neg a$ is one:

$$P(a|\pi) + P(\neg a|\pi) = 1. \quad (2)$$

For notational convenience, we define the *discrete variable* X as being a set E_X of k_X logical propositions $[X = x_i]$ such that these propositions are mutually exclusive ($[X = x_i] \wedge [X = x_j]$ is false unless $i = j$) and exhaustive (at least one of the proposition $[X = x_i]$ is true). When introducing variables, we will merely give the domain E_X of possible values for that variable, along with its cardinal k_X . The conjunction $X \otimes Y$ (or simply XY) of two variables X and Y then corresponds to the set of $k_X k_Y$ propositions $[X = x_i] \wedge [Y = y_j]$. XY corresponds to a set of mutually exclusive and exhaustive logical propositions: as such, it is a new variable. The two rules 1 and 2, when applied to variables, become

$$P(X \otimes Y|\pi) = P(X|\pi)P(Y|X\pi) = P(Y|\pi)P(X|Y\pi), \quad (3)$$

and

$$\sum_X P(X|\pi) = 1. \quad (4)$$

From these two rules we derive the marginalization rule, which allows easier derivations:

$$\sum_X P(X \otimes Y|\pi) = P(Y|\pi).$$

Given a set of n variables $\{X_1, X_2, \dots, X_n\}$, a *question* is defined as a partition of this set in three subsets ξ_S , ξ_K and ξ_U , denoting the sets of searched, known and unknown variables. Let *Searched*, *Known* and *Unknown* be the conjunctions of the variables in ξ_S , ξ_K and ξ_U . Given the joint distribution $P(X_1 X_2 \dots X_n|\pi) = P(\text{Searched} \otimes \text{Known} \otimes \text{Unknown}|\pi)$, it is possible to compute the probability distribution $P(\text{Searched}|\text{Known} \otimes \pi)$, using the following derivation:

$$\begin{aligned} & P(\text{Searched}|\text{Known} \otimes \pi) \\ &= \sum_{\text{Unknown}} P(\text{Searched} \otimes \text{Unknown}|\text{Known} \otimes \pi) \\ &= \frac{\sum_{\text{Unknown}} P(\text{Searched} \otimes \text{Unknown} \otimes \text{Known}|\pi)}{P(\text{Known}|\pi)} \end{aligned}$$

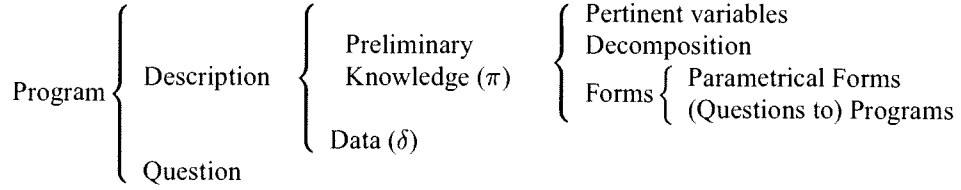


FIGURE 1. Structure of a bayesian program.

$$\begin{aligned}
 &= \frac{\sum_{Unknown} P(Searched \otimes Unknown \otimes Known | \pi)}{\sum_{Searched, Unknown} P(Searched \otimes Unknown \otimes Known | \pi)} \\
 &= \frac{1}{Z} \times \sum_{Unknown} P(Searched \otimes Unknown \otimes Known | \pi),
 \end{aligned}$$

where Z is a normalization constant. Answering a “question” consists in deciding a value for the variable *Searched* according to the distribution $P(Searched | Known \otimes \pi)$. Different decision policies are possible, in our programming system 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. Exact inference has been proved to be \mathcal{NP} -hard [13]; however, we developed an inference engine which proceeds in two phases, the first consisting of symbolic simplifications which reduce the complexity of the considered sums, and the second which computes an approximation of the distributions.

METHOD

Our programming method relies on the fact that, given the joint distribution, one can answer any question. A robotic task can be seen as two components:

- 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, \dots, X_n , given a set of experimental data δ and preliminary knowledge π . This joint distribution is denoted $P(X_1 X_2 \dots X_n | \delta \pi)$.
- A *procedural* component, which consists of using the previously defined description with a question.

These two components, along with their sub components, form a structure we always apply when programming a robotic task: this structure can be seen Figure 1.

We now detail each of these two phases, using a very simple example. In this example, our goal is to program a light following reactive behaviour. Suppose we have, on a robot, a sensory variable θ obtained from low level sensors, giving information about the light source orientation relative to the robot. Suppose we control the robot using one motor variable, V_{rot} , the rotation speed of the robot (the translation speed, V_{trans} , is set to a constant for this program). A reactive behaviour is simply a relation between the motor

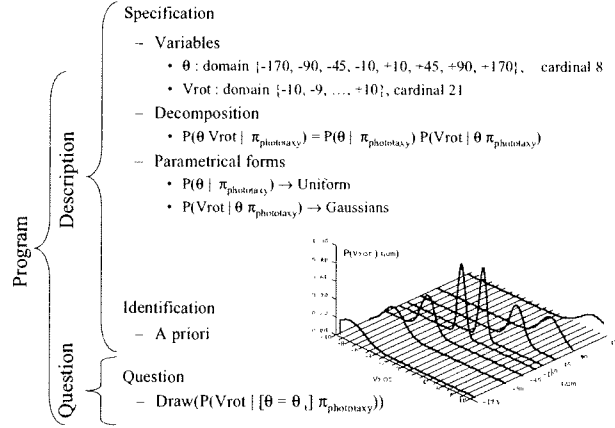


FIGURE 2. An example of a program. It shows both the program structure our method defines, and an example where the robot follows light. The plot shown represents the probability distributions $P(Vrot|\theta \otimes \pi_{phototaxy})$, one for each value of θ , that were defined a priori.

command at time t and the sensory state at the same time step t . A program to perform the light following task is shown Figure 2.

The **specification phase** has three components:

- **Variables** The programmer specifies which variables are relevant for the task. In our running example, we have one sensory variable, θ , which can take 8 different values, from -170 (light source behind, slightly on the left of the robot), to +170 (light source behind, slightly on the right of the robot). We also have one motor command, $Vrot$, with 21 possible values, from -10 (turn on the left with maximum rotation speed) to +10 (turn on the right with maximum rotation speed).
- **Decomposition of the joint distribution** The second specification step consists in giving a decomposition of $P(\theta \otimes Vrot | \Delta \otimes \pi_{phototaxy})$, as a product of simpler terms. Using the rule 3, several decompositions are mathematically correct, we choose

$$P(\theta \otimes Vrot | \Delta \otimes \pi_{phototaxy}) = P(\theta | \Delta \otimes \pi_{phototaxy}) P(Vrot | \theta \otimes \Delta \otimes \pi_{phototaxy}).$$

We could further simplify some terms appearing in the decomposition, using conditional independence hypotheses (see Section for some examples).

- **Parametrical forms** To be able to compute the joint distribution, we finally need to assign parametrical forms to each term appearing in the decomposition:

$$\begin{aligned} P(\theta | \Delta \otimes \pi_{phototaxy}) &\equiv Uniform, \\ P(Vrot | \theta \otimes \Delta \otimes \pi_{phototaxy}) &\equiv G_{\mu(\theta), \sigma(\theta)}(Vrot). \end{aligned}$$

We have no a priori knowledge about the general orientation of the light source, relative to the robot. Therefore we assign a uniform distribution to $P(\theta | \Delta \otimes$

$\pi_{phototaxy}$). In addition, we assume a single rotation speed must be preferred for each sensory situation. Hence, $P(Vrot|\theta \otimes \Delta \otimes \pi_{phototaxy})$ has to be unimodal. However, the confidence in this choice may vary with the situation; this leads to assigning gaussian parametrical forms to this term¹.

This completes the specification phase.

In the **identification phase**, the programmer has to assess the values of the free parameters. In simple cases, the programmer may do it himself, by writing a function or table that stores these parameters. The light following program shown in Figure 2 was obtained this way (see the gaussians in the plot): we call this method *a priori programming*. However, it is often easier to justify parameters when they have been computed by a learning algorithm. In our example, since we only have mean values and standard deviations to set, this learning phase is simple. Using a joystick, we pilot the robot to follow the light. Every tenth of a second, we record experimental data $\langle \theta_t, vrot_t \rangle$, where θ_t is computed from the low level sensors at time t , and $vrot_t$ is the motor command given by the user at the same time t . Given a set Δ_{follow} of such data, computing the mean values and standard deviations of the gaussian distributions associated with the $P(Vrot|\theta \otimes \Delta_{follow} \otimes \pi_{phototaxy})$ term is straightforward.

The description being now completed, we can have the robot play back the knowledge it has been given, by a question. In this case, the robot should answer the following question:

$$Draw(P(Vrot|[\theta = \theta_t] \otimes \Delta_{follow} \otimes \pi_{phototaxy})).$$

EXPERIMENT

The goal of this section is to program a robot so that it patrols its environment, goes back to its base when asked to, or when its batteries get low. When patrolling, the robot will wander aimlessly, while avoiding obstacles. The base of the robot will be a recess in the environment, with a strong light source over it, so that the homing behaviour can be obtained by combining obstacle avoidance with light following. This program will be built incrementally: we will first describe the low level reactive behaviours relevant to the task (obstacle avoidance, light following, homing). Then, we will define two sensor models, one for accurately sensing the light source position, the other for deciding if the robot is at its base. Next comes the patrolling layer, where we relate high level sensory information (orders from the user for example) with high level motor commands (choice of behaviour). The final program integrates together all these building blocks.

¹ This is actually a discrete approximation of a gaussian form:

$$P(\{V = v_i\}|\Delta\pi) = \int_{v_i-\epsilon}^{v_i+\epsilon} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(v-\mu)^2}{2\sigma^2}} dv$$

Since the domain for variables are finite, we also have to normalize afterwards.

The robot and its variables

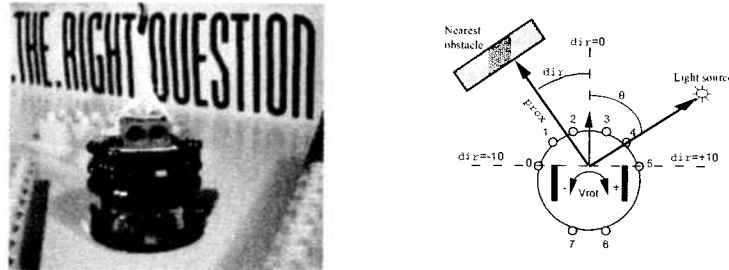


FIGURE 3. On the left, a picture of the Khepera. On the right, a schema of the Khepera with some sensory and motor variables.

The Khepera (see Figure 3) is a miniature mobile robot built by the EPFL (Ecole Polytechnique Fédérale de Lausanne) and produced by K-Team. The Khepera is a mobile robot with two wheels, is 57 mm in diameter and 29 mm tall, for a total weight of 80 g, in the basic configuration. It is equipped with eight light sensors (6 in front and 2 behind) having values ranging from 0 to 511, values decreasing with increasing light (variables $Lm0$ to $Lm7$). From these light sensors, we can derive a variable θ , that corresponds to the bearing of the most powerful light source of the environment. These eight sensors can also be used as infrared proximity sensors, with values from 0 to 1023 as a decreasing function of the obstacle distance (variables $Px0$ to $Px7$). From the six front proximeters, we derive two variables, Dir and $Prox$ (with domains $\{-10, -9, \dots, +10\}$ and $\{0, 1, \dots, 15\}$, respectively), that roughly correspond to the direction and proximity of the nearest obstacle. The Khepera also has rough odometry capabilities. The robot is piloted using the rotation and translation speeds, using variables $Vrot$ and $Vtrans$. In the following, $Vtrans$ is set to a constant, unless noted otherwise.

Behaviour combination

In this section, we want to program a homing behaviour for the robot, by combining light following and obstacle avoidance. We first describe these two components using two descriptions, then we write a program that combines them.

Light following behaviour

This description has already been defined Figure 2. Let us just recall here that the preliminary knowledge corresponding is $\pi_{phototaxy}$.

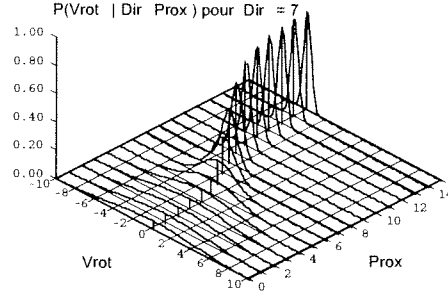


FIGURE 4. The gaussians for the distribution $P(Vrot|[Dir = 7]Prox\pi_{avoid})$. $[Dir = 7]$ corresponds to an object on the right side of the robot (approx. 45°). When $Prox$ is high (obstacle near), $Vrot$ takes negative values with high confidence (turn to the left). However, when $Prox$ is 0 (no obstacle sensed), $Vrot$ is not constrained much, and the resulting law gets close to a uniform distribution.

Obstacle avoidance behaviour

The obstacle avoidance behaviour will simply consist of controlling the rotation speed of the robot ($Vrot$), with two sensory variables Dir and $Prox$, describing the direction and proximity to the nearest obstacle. The preliminary knowledge associated with this behaviour description is the following:

- **Variables** $Vrot$, Dir and $Prox$.
- **Decomposition of the joint distribution**

$$P(Dir Prox Vrot | \Delta \pi_{avoid}) = P(Dir Prox | \Delta \pi_{avoid}) P(Vrot | Dir Prox \Delta \pi_{avoid})$$

- **Parametrical forms**

$$P(Dir Prox | \Delta \pi_{avoid}) \equiv Uniform,$$

$$P(Vrot | Dir Prox \Delta \otimes \pi_{avoid}) \equiv G_{\mu(Dir, Prox), \sigma(Dir, Prox)}(Vrot).$$

This preliminary knowledge, π_{avoid} , is very similar to $\pi_{phototax}$: in both cases, it consists of a product of a uniform distribution $P(S)$ over the sensory variables S and of the direct control law of the form $P(M|S)$, with M the motor variables. The free parameters of the gaussians can be obtained by experimentation (data Δ obtained by joysticking the robot), however, for the π_{avoid} case, we choose to define them a priori, therefore the Δ variable can be omitted. For example, the specification of the standard deviations express constraints on $Vrot$: far from obstacles, the rotation speed need not be constrained (large standard deviation), and near obstacles, the rotation speed need to follow closely the order (small standard deviation around the mean value). We show an example of the gaussians for the case $[Dir = 7]$, Figure 4.

Homing behaviour

We now have two descriptions that can control the rotation speed of the robot. The goal of the next description π_{homing} is to combine them. This description concerns all

the variables appearing in $\pi_{phototaxy}$ and π_{avoid} , and a new variable, H . This variable indicates what behaviour should be used: when $[H = p]$, we do phototaxy, when $[H = a]$, we do obstacle avoidance. The choice between these two modes depends on the proximity of the nearest obstacle only (variable $Prox$) : the nearer this obstacle, the higher the probability of doing obstacle avoidance. We translate all these choices in our formalism:

- **Variables** $Dir, Prox, \theta, H, Vrot$.
- **Decomposition of the joint distribution**

$$P(DirProx\theta HVrot|\Delta\pi_{homing}) = P(DirProx\theta|\pi_{homing})P(H|Prox\pi_{homing})P(Vrot|HDirProx\theta\pi_{homing}).$$

- **Parametrical forms**

$$\begin{aligned} P(DirProx\theta|\pi_{homing}) &\equiv Uniform, \\ P(H|Prox\pi_{homing}) &\equiv Table, \\ P(Vrot|[H = p]DirProx\theta\pi_{homing}) &\equiv P(Vrot|\theta\pi_{phototaxy}), \\ P(Vrot|[H = a]DirProx\theta\pi_{homing}) &\equiv P(Vrot|DirProx\pi_{avoid}). \end{aligned}$$

The tables associated with $P(H|Prox\pi_{homing})$ are defined a priori, so that when $Prox$ is minimum, the probability of doing phototaxy is maximum, and when $Prox$ is maximum, the probability of doing obstacle avoidance is maximum. The last term makes the link between the homing program and its two resources, via two questions to the $\pi_{phototaxy}$ and π_{avoid} descriptions.

All the terms are specified, no learning is needed.

The question we ask, and its resolution, are:

$$\begin{aligned} P(Vrot|[Dir = d_t][Prox = p_t][Lum = l_t]\pi_{homing}) = \\ P([H = a][Prox = p_t]\pi_{homing})P(Vrot|[Dir = d_t][Prox = p_t]\pi_{avoid}) \\ + P([H = p][Prox = p_t]\pi_{homing})P(Vrot|[Lum = l_t]\pi_{phototaxy}), \end{aligned}$$

which means that the resulting command is a weighted combination of motor commands given by the obstacle avoidance and light following programs, and not just a all-or-nothing kind of combination.

Sensor fusion

The goal of this section is to provide an effective way of computing the variable θ , in two steps. we first build, for each light sensor, a description π_{lmi} modelling the sensor's response to a given light source (direction and distance to the robot known). Then, we merge all these models associated with each sensor, in a description π_{SF} (SF for sensor fusion), to compute the light source position relative to the robot.

Sensor model

For a given sensor i , we decide to consider only three variables: Lmi , the response of the sensor, θ and $Dist$, the angle and distance of the light source relative to the robot (see Figure 3). The domains of these variables are $\{0, 1, \dots, 511\}$, $\{-180, -170, \dots, +170\}$ and $\{0, 1, \dots, 25\}$, respectively. The decomposition we choose for the joint distribution is the following:

$$P(Lmi\theta Dist|\Delta\pi_{lmi}) = P(\theta Dist|\Delta\pi_{lmi})P(Lmi|\theta Dist\Delta\pi_{lmi}).$$

It is our modelization choice to assign a uniform distribution to $P(\theta Dist|\Delta\pi_{lmi})$. This choice represents our ignorance about the direction of the light source in the environment. Finally, gaussian forms are associated to the $P(Lmi|\theta Dist\Delta\pi_{lmi})$ term.

For the identification phase, we only have to define the means and standard deviations of the $P(Lmi|\theta Dist\Delta\pi_{lmi})$ term. This can easily be done, following the manufacturer's characterization of the sensor. Indeed, this characterization describes the response of the sensor to a given stimulus, which is exactly the semantics associated with this term of the decomposition. Since this is done without any experimental data (some calibration could indeed be included at this point), we can drop the Δ variable.

There are several ways of using this description: for instance, the question $P(\theta|Lmi\pi_{lmi})$ corresponds to computing the probability distribution over all possible θ values, knowing the sensor value, but ignoring the distance of the light source. $P(Dist|Lmi\pi_{lmi})$, or $P(Dist\theta|Lmi\pi_{lmi})$ are other possible questions.

Fusion

In order to merge the information given by the eight light sensors about the light source position and distance, we naturally select the variables characterizing the light source (θ and $Dist$, see above), and the eight sensor values ($Lm0$ to $Lm7$, with the same domain as lmi). For our application, we use the following decomposition (we omit the Δ and π_{SF} variables for clarity) :

$$\begin{aligned} &P(\theta Dist Lm0 \dots Lm7) \\ &= P(\theta Dist)P(Lm0|\theta Dist)P(Lm1|Lm0\theta Dist) \dots P(Lm7|Lm6 \dots Lm0\theta Dist) \\ &= P(\theta Dist)P(Lm0|\theta Dist)P(Lm1|\theta Dist) \dots P(Lm7|\theta Dist). \end{aligned}$$

The first equality is simply an application of the product rule, while the second is the translation of a conditional independence hypothesis stating that, *knowing the characteristics of the light source*, we consider the readings on the different sensors independent. Again, we decide to assign a uniform distribution to the $P(\theta Dist|\Delta\pi_{SF})$ term. As for the terms $P(Lmi|\theta Dist\Delta\pi_{SF})$, that concern a single sensor, we can use the description

π_{lmi} , by asking a question to it: formally, we write:

$$P(Lmi|\theta Dist \Delta \pi_{SF}) = P(Lmi|\theta Dist \pi_{SF}) = P(Lmi|\theta Dist \pi_{lmi})^2.$$

There are no free parameters in π_{SF} , therefore there is no identification phase, and we can drop the Δ variable.

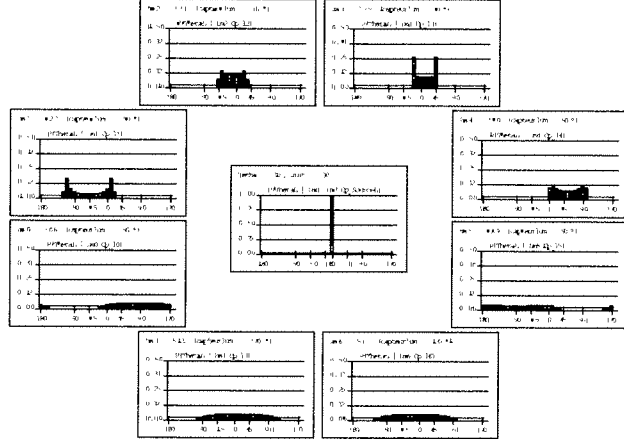


FIGURE 5. Fusion of the eight sensors for a light source placed slightly on the right side of the robot (θ should read 10), and moderately far ($Dist$ is around 10). The eight peripheral plots are the individual sensor model responses $P(\theta|Lmi\pi_{lmi})$, and the central plot is the answer to the $P(\theta|Lm0 \cdots Lm7\pi_{SF})$ question.

We can now use the π_{SF} description in many different ways, for example:

- $P(\theta|Lm0Lm1 \cdots Lm7\pi_{SF})$, if we want to compute the position of the light source relative to the robot. A specific example, showing how well the signal gets enhanced when merging the eight sensor models, is shown Figure 5. We clearly see that, while no individual sensor can measure θ precisely, the fusion of the eight models give a very accurate and certain response. If we decide for a value of θ , according to this distribution, we can obtain at every time step t a value for the variable θ . This is how θ was obtained in the $\pi_{phototaxy}$ and π_{homing} programs previously described.
- In the same way, we can also ask $P(Dist|Lm0Lm1 \cdots Lm7\pi_{SF})$, or $P(Dist\theta|Lm0Lm1 \cdots Lm7\pi_{SF})$.
- Questions like $P(Lm3|\theta Lm2Lm4\pi_{SF})$, for example, allow to predict what should be read on some sensors, knowing what the nearest sensors read. In this fashion, we approach sensor failure diagnosis.

² This actually requires a little geometric computation in order to shift the individual models to account for the sensor placement on the robot body.

Base recognition

In the same fashion, we can program the robot to recognize its base, which is a small recess in its environment, with a light source over it. This sensory situation is considered typical of “being in the base”, and can be learned experimentally by the robot. The associated description is a sensor model of the base, called π_{base} , and allows to compute values of a boolean variable, $Base$, which is true if the robot recognizes the current sensory situation as the one learned for the base.

Patrolling

The next description, π_{patrol} , is providing the task level control of the robot: the “motor” variable of this layer is B_t (for Behaviour at time t), which can take four possible values: $[B_t = standby]$ when the robot is inactive (at the base), $[B_t = patrol]$ when the robot is randomly patrolling the environment, $[B_t = homing]$ when the robot was asked to go back to standby mode, and $[B_t = henergy]$ when the robot goes back to the base to recharge its batteries. There is one internal variable, B_{t-1} , which memorizes what was the selected behaviour at the previous time step. Finally, there are three binary sensory variables, $Base$, which is obtained via a question to the π_{base} description, $Energy$, which is read from an internal sensor (actually simulated: four possible values: vH, H, L, vL , for very high, high, low and very low), and $Vigil$, which is an order given by the user ($[Vigil = 1]$ asks the robot to go on patrol, $[Vigil = 0]$ asks it to go back to its base and get in standby mode).

As in the sensor model programs, we apply in the decomposition phase what we call “inverse programming” (we omit π_{patrol} for clarity, and Δ , since there is no learning involved):

$$P(B_t B_{t-1} Base Energy Vigil) = P(B_{t-1}) P(B_t | B_{t-1}) P(Base | B_t) P(Energy | B_t) P(Vigil | B_t).$$

These terms may appear counter-intuitive at first, but they actually are very easy to define, and allow to break the complexity of the sensory-motor space. B_{t-1} is treated as a sensory variable given by the same description at the previous time step. The $P(B_{t-1})$ term is thus assigned a uniform distribution. The “diagonal” elements of $P(B_t | B_{t-1})$ describe the stability of each behaviour, and we can specify in the rest some impossible transitions (for instance, if we were inactive at the previous time step, we therefore are at the base, and there is no sense in switching suddenly to any homing behaviour: we set $P([B_t = homing] | [B_{t-1} = standby])$ to 0). The last three terms, $P(Base | B_t)$, $P(Energy | B_t)$ and $P(Vigil | B_t)$, describe what should be read on the sensory variables, given what is the current behaviour.

These terms (except the $P(B_{t-1})$ one), are defined using tables, which are shown Figure 6.

		B_{t-1}						B_{t-1}						B_{t-1}				
		standby	patrol	homing	h-energy			standby	patrol	homing	h-energy			standby	patrol	homing	h-energy	
B_{t-1}	standby	0.9	0.03	0.03	0.03	Base	0	0	0.99	0.99	0.00	Energy	Vt	0.325	0	0.25	0.8	
	patrol	0.0	0.91	0.03	0.03		1	1	0.00	0.01	0.00		L	0.325	0.0	0.25	0.2	
	homing	0	0.01	0.91	0.03		Vigil	0	0.9	0	1		0.5	0	0.25	0.45	0.25	0
	h-energy	0	0.03	0.03	0.91		1	0.1	1	0	0.5		VH	0.0	0.45	0.25	0	

FIGURE 6. The tables defining the terms of the decomposition. Each column sums to one.

Integration

The final program glues together all the previous components: it includes all variables, and the decomposition and parametrical forms are as follows:

$$\begin{aligned}
& P(Vrot|B_t|B_{t-1}|Base|Energy|Vigil|H|Dir|Prox|\theta|Lm0 \cdots Lm7|Px0 \cdots Px7|\pi) \\
& = P(Energy|Vigil|Dir|Prox|Lm0 \cdots Lm7|Px0 \cdots Px7|\pi) \\
& \quad P(H|Prox|\pi_{homing}) \\
& \quad P(\theta|Lm0 \cdots Lm7|\pi_{SF}) \\
& \quad P(B_t|B_{t-1}|Base|Energy|Vigil|\pi_{patrol}) \\
& \quad P(Vrot|B_t|H|\theta|Dir|Prox|\pi_{movingmode}).
\end{aligned}$$

The only term not defined here is the last one, which refers to the description $\pi_{movingmode}$, which makes the link between the behaviour to apply and the value of the variable B_t . Please refer to [10, 11] for more details.

DISCUSSION

We have introduced a new formalism for programming robots. Our approach closely implements the Bayesian paradigm and, as a result, follows a clear mathematical framework. It permits programming robots while explicitly taking into account the incompleteness of the models chosen by the programmer.

Our programming method relies on the fact that, given the joint distribution, one can answer many questions. A robotic task results from both a declarative component which specifies a method to compute the joint distribution, and a procedural component which consists of using the previously defined description with a specific question solved by inference.

When programming a robot, the necessity to completely specify the joint distribution may seem coercive and superfluous: for example, in the phototaxy program, defining and using only $P(Vrot|\theta)$ is enough. Generally, the bayesian approaches in robotics do not define anything more than what is needed for the task at hand. In contrast, our method relies on this distinction between what we “know”, and what we want to “do”. This gives the programmer more freedom when defining the preliminary knowledge: some conditional independence hypotheses may help choose toward a certain decomposition, some terms may be chosen because they are easier to assess than others (by learning or by hand), some terms because they can be defined as questions to other descriptions, some terms because they make inference easier, etc. This distinction also gives the

possibility to use a description for other purposes than the first intended. All these aspects have been exemplified in the presented programs.

Hence, a description may be considered as a resource, allowing programming of complex tasks in an incremental manner. We have illustrated both the combination of descriptions (defining a new behaviour as a weighted mixture of simpler ones) and the hierarchical composition of descriptions (similar to calling sub-procedures in classical programming, as some of the parametric forms appearing in a decomposition may be questions addressed to more “basic” descriptions).

Descriptions offer a large capacity of expression to specify models and use them as well. The specification phase compels the programmer to exhaustively express the information he has about the task. Everything that should be known about a given robotics problem is in the description: the synthesis between the preliminary knowledge and the experimental data. There is no hidden knowledge in either the inference program or the decision algorithm. As the description encapsulates all the relevant information, exchanging, sharing or discussing models is easy and rigorous.

REFERENCES

1. E. Jaynes, *Probability theory - The logic of science*, Unfinished book, Note : Publicly available at <http://bayes.wustl.edu>, 1998.
2. R. Bernhardt and S. A. (editors), *Robot Calibration*, Chapman & Hall, 1993.
3. R. Brafman, J.-C. Latombe, Y. Moses, and Y. Shoham, “Applications of a logic of knowledge to motion planning under uncertainty,” *Journal of the ACM*, **44**(5), 1997.
4. L. Kaelbling, M. Littman, and A. Cassandra, “Planning and acting in partially observable stochastic domains,” *Artificial Intelligence*, **101**, 1998.
5. S. Kristensen, “Sensor planning with bayesian decision theory,” *Robotics and Autonomous Systems*, **19**, pp. 273–286, March 1997.
6. O. Aycard, *Architecture de contrôle pour robot mobile en environnement intérieur structuré*. Thèse de doctorat, Université Henri Poincaré, Nancy, France, 1998.
7. S. Thrun, D. Fox, and W. Burgard, “A probabilistic approach to concurrent mapping and localization for mobile robots,” *Machine Learning and Autonomous Robots (joint issue)*, **31/5**, pp. 253–271, 1998.
8. F. Dellaert, D. Fox, W. Burgard, and S. Thrun, “Monte carlo localization for mobile robots,” *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1999.
9. J. Leonard, H. Durrant-Whyte, and I. Cox, “Dynamic map building for an autonomous mobile robot,” *International Journal of Robotics Research*, **11**, August 1992.
10. O. Lebeltel, *Programmation bayésienne des robots*. Thèse de doctorat, Inst. Nat. Polytechnique de Grenoble, Grenoble, France, Octobre 1999.
11. O. Lebeltel, P. Bessière, J. Diard, and E. Mazer, “Bayesian robots programming,” *Les cahiers du laboratoire Leibniz ; 01*, 2000.
12. R. Cox, *The algebra of probable inference*, The John Hopkins Press, Baltimore, USA, 1961.
13. G. Cooper, “The computational complexity of probabilistic inference using bayesian belief networks,” *Artificial Intelligence*, **42**, 1990.