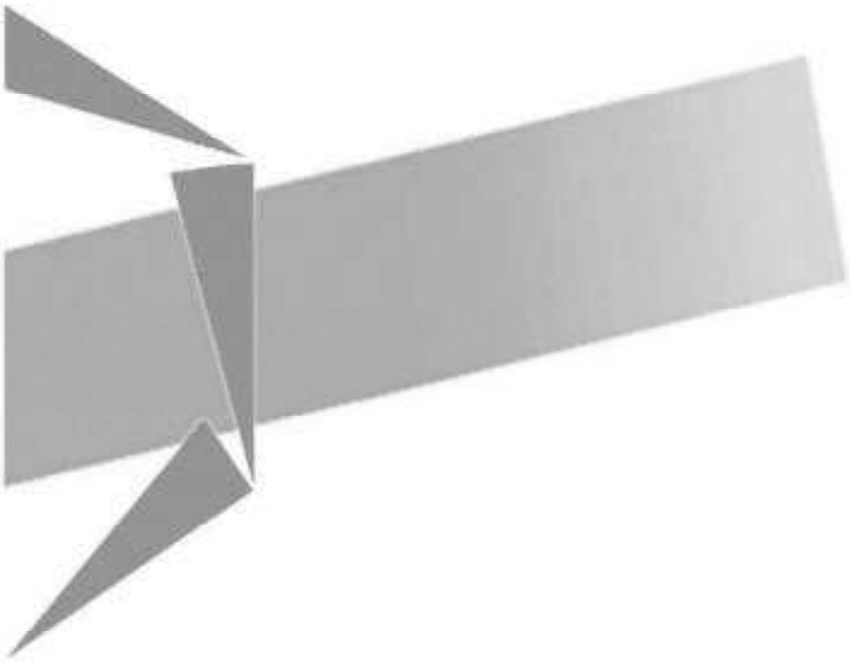
Les cahiers
du laboratoire
Leibniz



Bayesian Robots Programming

Olivier Lebeltel
Pierre Bessière
Julien Diard
Emmanuel Mazer

Laboratoire Leibniz-IMAG, 46 av. Félix Viallet, 38000 GRENOBLE, France - n° 01

ISSN : en cours

Mai 2000

Cahiers du Laboratoire LEIBNIZ

Bayesian Robots Programming

Olivier Lebeltel Olivier.Lebeltel@imag.fr
Pierre Bessière Pierre.Bessiere@imag.fr
Julien Diard Julien.Diard@imag.fr
Laboratoire LEIBNIZ - CNRS
46 avenue Félix Viallet ; 38031 Grenoble ; FRANCE

Emmanuel Mazer Emmanuel.Mazer@imag.fr
Laboratoire GRAVIR - CNRS
INRIA Rhône-Alpes, ZIRST, 38030 Montbonnot

Abstract

We propose a new method to program robots based on Bayesian inference and learning. The capacities of this programming method are demonstrated through a succession of increasingly complex experiments. Starting from the learning of simple reactive behaviors, we present instances of behavior combinations, sensor fusion, hierarchical behavior composition, situation recognition and temporal sequencing. This series of experiments comprises the steps in the incremental development of a complex robot program. The advantages and drawbacks of this approach are discussed along with these different experiments and summed up as a conclusion. These different robotics programs may be seen as an illustration of probabilistic programming applicable whenever one must deal with problems based on uncertain or incomplete knowledge. The scope of possible applications is obviously much broader than robotics.

1. Introduction

> By inference we mean simply: deductive reasoning whenever enough information is at hand to permit it; inductive or probabilistic reasoning when - as is almost invariably the case in real problems - all the necessary information is not available. Thus the topic of "Probability as Logic" is the optimal processing of uncertain and incomplete knowledge.
>
> E.T. Jaynes[1]

We assume that any model of a real phenomenon is *incomplete*. There are always some hidden variables, not taken into account in the model, that influence the phenomenon. The effect of these hidden variables is that the model and the phenomenon never have the same behavior.

Any robot system must face this central difficulty: how to use an incomplete model of its environment to perceive, infer, decide and act efficiently? We propose an original robot programming method that specifically addresses this question.

Rational reasoning with incomplete information is quite a challenge for artificial systems. The purpose of Bayesian inference and learning is precisely to tackle this problem with a well-established formal theory. Our method heavily relies on this Bayesian framework.

1. (Jaynes, 1998)

© 2000 CNRS - UJF - INPG, All rights reserved.

We present several programming examples to illustrate this approach and define *descriptions* as generic programming resources. We show that these resources can be used to incrementally build complex programs in a systematic and uniform framework. The system is based on the simple and sound basis of Bayesian inference. It obliges the programmer to explicitly state all assumptions that have been made. Finally, it permits effective treatment of incomplete and uncertain information when building robot programs.

The paper is organized as follows. Section 2 offers a short review of the main related work, Section 3 is dedicated to definitions and notations and Section 4 presents the experimental platform. Sections 5 to 10 present various instances of Bayesian programs: learning simple reactive behaviors; instances of behavior combinations; sensor fusion; hierarchical behavior composition; situation recognition; and temporal sequencing. Section 11 describes a combination of all these behaviors to program a robot to accomplish a night watchman task. Finally, we conclude with a synthesis summing up the principles, the theoretical foundations and the programming method. This concluding section stresses the main advantages and drawbacks of the approach.

2. Related work

Our work is based on an implementation of the principle of the Bayesian theory of probabilities.

In physics, since the precursory work of Laplace (1774; 1814), numerous results have been obtained using Bayesian inference techniques (to take uncertainty into account) and the maximum entropy principle (to take incompleteness into account). The late Edward T. Jaynes proposed a rigorous and synthetic formalization of probabilistic reasoning with his "Probability as Logic" theory (Jaynes, 1998). A historical review of this approach was offered by Jaynes (1979) and an epistemological analysis, by Matalon (1967). Theoretical justifications of probabilistic inference and maximum entropy are numerous. The entropy concentration theorems (Jaynes, 1982; Robert, 1990) are among the more rigorous, Cox theorem (Cox, 1961) being the most well known, although it has been partially disputed recently by Halpern (1999a; 1999b). Numerous applications and mathematical tools have been developed (Smith & Grandy, 1985; Tarentola, 1987; Bretthorst, 1988; Erickson & Smith, 1988a; Erickson & Smith, 1988b; Mohammad-Djafari & Demoment, 1992; Kapur & Kesavan, 1992).

In artificial intelligence, the importance of reasoning with uncertain knowledge has been recognized for a long time. However, the Bayesian approach clearly appeared as one of the principle trends only since the proposal of Bayesian nets (Pearl, 1988) and graphical models (Lauritzen & Spiegehalter, 1988; Lauritzen, 1996; Jordan, 1998; Frey, 1998). Very important technical progress has been achieved recently (See for instance the JAIR[2] articles on that subject: Saul et al., 1996; Zhang & Poole, 1996; Delcher et al., 1996; Darwiche & Provan, 1997; Ruiz et al., 1998; Jaakola & Jordan, 1999; Jordan et al., 1999).

Recent robot programming architectures (Alami et al., 1998; Borrelly et al., 1998; Schneider et al., 1998; Dekhil & Henderson, 1998; Mazer et al., 1998) are in general not concerned with the problem of uncertainty. In robotics, the uncertainty topic is either related to calibration (Bernhardt & Albright, 1993) or to planning problems (Brafman et al., 1997). In the latter case, some authors have considered modeling the uncertainty of the robot motions when planning assembly operations (Lozano-Perez et al., 1984; Donald, 1988) or modeling the uncertainty related to the position of the robot in a scene (Kapur & Kesavan,

2. Journal of Artificial Intelligence Research

1992). Bayesian techniques are used in POMDP[3] to plan complex paths in partially known environments (Kaelbling, Littman & Cassandra, 1996). HMM[4] are also used to plan complex tasks and recognize situations in complex environments (Aycard, 1998, Thrun, 1998). However, to the best of our knowledge, the design of a robot programming system and architecture solely based on Bayesian inference has never been investigated before.

Finally, a presentation of the epistemological foundations of the approach described in this paper may be found in two articles by Bessière et al. (1998a; 1998b) and all the technical details in the PhD dissertation of Olivier Lebeltel (1999).

3. Basic concepts

In this section, we introduce the concepts, postulates, definitions, notations and rules that are necessary to define a Bayesian robot program.

3.1 Definition and notation

Proposition

The first concept we will use is the usual notion of *logical proposition*. Propositions will be denoted by lowercase names. Propositions may be composed to obtain new proposition using the usual logical operators: $a \wedge b$ denoting the conjunction of propositions a and b, $a \vee b$ their disjunction and $\neg a$ the negation of proposition a.

Variable

The notion of *discrete variable* is the second concept we require. Variables will be denoted by names starting with one uppercase letter.

By definition, a *discrete variable* X is a set of logical propositions x_i such that these propositions are mutually exclusive (for all i, j with $i \neq j$, $x_i \wedge y_j$ is false) and exhaustive (at least one of the propositions x_i is true). x_i stands for «variable X takes its i^{th} value». $\lfloor X \rfloor$ denotes the cardinal of the set X (the number of propositions x_i).

The conjunction of two variables X and Y, denoted $X \otimes Y$, is defined as the set of $\lfloor X \rfloor \times \lfloor Y \rfloor$ propositions $x_i \wedge y_j$. $X \otimes Y$ is a set of mutually exclusive and exhaustive logical propositions. As such, it is a new variable[5]. Of course, the conjunction of n variables is also a variable and, as such, it may be renamed at any time and considered as a unique variable in the sequel.

Probability

To be able to deal with uncertainty, we will attach probabilities to propositions.

We consider that, to assign a probability to a proposition a, it is necessary to have at least some *preliminary knowledge*, summed up by a proposition π. Consequently, the probability of a proposition a is always conditioned, at least, by π. For each different π, $\mathbf{P}(. \mid \pi)$ is an application assigning to each proposition a a unique real value $\mathbf{P}(a \mid \pi)$ in the interval $[0, 1]$.

Of course, we will be interested in reasoning on the probabilities of the conjunctions,

3. Partially Observable Markoff Decision Process
4. Hidden Markov Models
5. By contrast, the disjunction of two variables, defined as the set of propositions $x_i \vee y_j$, is not a variable. These propositions are not mutually exclusive.

disjunctions and negations of propositions, denoted, respectively, by $\mathbf{P}(a \wedge b \mid \pi)$, $\mathbf{P}(a \vee b \mid \pi)$ and $\mathbf{P}(\neg a \mid \pi)$.

We will also be interested in the probability of proposition a conditioned by both the preliminary knowledge π and some other proposition b. This will be denoted $\mathbf{P}(a \mid b \wedge \pi)$.

For simplicity and clarity, we will also use probabilistic formula with variables appearing instead of propositions. By convention, each time a variable X appears in a probabilistic formula $\Phi(X)$, it should be understood as $\forall x_i \in X, \Phi(x_i)$. For instance, given three variables X, Y and Z, $\mathbf{P}(X \otimes Y \mid Z \otimes \pi) = \mathbf{P}(X \mid \pi)$ stands for:

$$\forall x_i \in X, \forall y_j \in Y, \forall z_k \in Z \qquad \mathbf{P}(x_i \wedge y_j \mid z_k \wedge \pi) = \mathbf{P}(x_i \mid \pi) \qquad \text{[E3.1]}$$

3.2 Inference postulates and rules

This section presents the inference postulates and rules necessary to carry out probabilistic reasoning.

Conjunction and normalization postulates for propositions

Probabilistic reasoning needs only two basic rules:

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

$$\begin{aligned} \mathbf{P}(a \wedge b \mid \pi) &= \mathbf{P}(a \mid \pi) \times \mathbf{P}(b \mid a \wedge \pi) \\ &= \mathbf{P}(b \mid \pi) \times \mathbf{P}(a \mid b \wedge \pi) \end{aligned} \qquad \text{[E3.2]}$$

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

$$\mathbf{P}(a \mid \pi) + \mathbf{P}(\neg a \mid \pi) = 1 \qquad \text{[E3.3]}$$

For the purpose of this paper, we take these two rules as postulates[6].

As in logic, where the resolution principle (Robinson, 1965; Robinson, 1979) is sufficient to solve any inference problem, in discrete probabilities, these two rules ([E3.2], [E3.3]) are sufficient for any computation. In particular, we may derive all the other necessary inference rules from those two, especially the rules concerning variables.

Disjunction rule for propositions

For instance, the rule concerning the disjunction of propositions:

$$\mathbf{P}(a \vee b \mid \pi) = \mathbf{P}(a \mid \pi) + \mathbf{P}(b \mid \pi) - \mathbf{P}(a \wedge b \mid \pi) \qquad \text{[E3.4]}$$

may be derived as follows:

6. See some references on justifications of these two rules in § 2.

$$\begin{aligned}
\mathbf{P}(a \vee b \mid \pi) &= 1 - \mathbf{P}(\neg a \wedge \neg b \mid \pi) \\
&= 1 - \mathbf{P}(\neg a \mid \pi) \times \mathbf{P}(\neg b \mid \neg a \wedge \pi) \\
&= 1 - \mathbf{P}(\neg a \mid \pi) \times (1 - \mathbf{P}(b \mid \neg a \wedge \pi)) \\
&= \mathbf{P}(a \mid \pi) + \mathbf{P}(\neg a \wedge b \mid \pi) \\
&= \mathbf{P}(a \mid \pi) + \mathbf{P}(b \mid \pi) \times \mathbf{P}(\neg a \mid b \wedge \pi) \\
&= \mathbf{P}(a \mid \pi) + \mathbf{P}(b \mid \pi) \times (1 - \mathbf{P}(a \mid b \wedge \pi)) \\
&= \mathbf{P}(a \mid \pi) + \mathbf{P}(b \mid \pi) - \mathbf{P}(a \wedge b \mid \pi)
\end{aligned} \qquad [E3.5]$$

Conjunction rule for variables

$$\begin{aligned}
\mathbf{P}(X \otimes Y \mid \pi) &= \mathbf{P}(X \mid \pi) \times \mathbf{P}(Y \mid X \otimes \pi) \\
&= \mathbf{P}(Y \mid \pi) \times \mathbf{P}(X \mid Y \otimes \pi)
\end{aligned} \qquad [E3.6]$$

According to our notation convention for probabilistic formula including variables, this may be restated as:

$$\begin{aligned}
&\forall x_i \in X, \forall y_j \in Y \\
\mathbf{P}(x_i \wedge y_j \mid \pi) &= \mathbf{P}(x_i \mid \pi) \times \mathbf{P}(y_j \mid x_i \wedge \pi) \\
&= \mathbf{P}(y_j \mid \pi) \times \mathbf{P}(x_i \mid y_j \wedge \pi)
\end{aligned} \qquad [E3.7]$$

which may be directly deduced from equation [E3.2].

Normalization rule for variables

$$\sum_X \mathbf{P}(X \mid \pi) = 1 \qquad [E3.8]$$

The normalization rule may obviously be derived as follows:

$$\begin{aligned}
1 &= \mathbf{P}(x_1 \mid \pi) + \mathbf{P}(\neg x_1 \mid \pi) \\
&= \mathbf{P}(x_1 \mid \pi) + \mathbf{P}(x_2 \vee \ldots\ldots \vee x_{\lfloor X \rfloor} \mid \pi) \\
&= \mathbf{P}(x_1 \mid \pi) + \mathbf{P}(x_2 \mid \pi) + \ldots\ldots + \mathbf{P}(x_{\lfloor X \rfloor} \mid \pi) \\
&= \sum_{x_i \in X} \mathbf{P}(x_i \mid \pi)
\end{aligned} \qquad [E3.9]$$

where the first equality derives from equation [E3.3], the second from the exhaustiveness of propositions x_i and the third from both the application of equation [E3.4] and the mutual exclusivity of propositions x_i.

Marginalization rule for variables

$$\sum_X \mathbf{P}(X \otimes Y \mid \pi) = \mathbf{P}(Y \mid \pi) \qquad [E3.10]$$

The marginalization rule is derived by the successive application of the product rule (equation [E3.6]) and the normalization rule (equation [E3.8]):

$$\begin{aligned}\sum_X \mathbf{P}(X \otimes Y \mid \pi) &= \sum_X \mathbf{P}(Y \mid \pi) \times \mathbf{P}(X \mid Y \otimes \pi) \\ &= \mathbf{P}(Y \mid \pi) \times \sum_X \mathbf{P}(X \mid Y \otimes \pi) \\ &= \mathbf{P}(Y \mid \pi)\end{aligned} \qquad \text{[E3.11]}$$

3.3 Bayesian Programs

We have defined a *Bayesian program* as a means of specifying a family of probability distributions. Our goal is to show that by using such a specification one can effectively control a robot to perform complex tasks.

The constituent elements of a Bayesian program are presented in Figure 1:

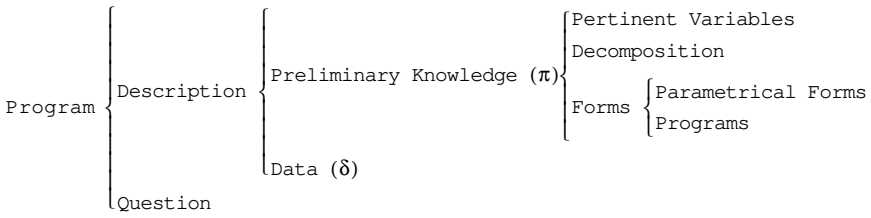


Figure 1: Structure of a bayesian program

- A program is constructed from a description and a question.
- A description is constructed from preliminary knowledge and a data set.
- Preliminary knowledge is constructed from a set of pertinent variables, a decomposition and a set of forms.
- Forms are either parametric forms or programs.

Description

The purpose of a description is to specify an effective method to compute a joint distribution on a set of variables $\{X^1, X^2, ..., X^n\}$ given a set of experimental data δ and preliminary knowledge π. This joint distribution is denoted as: $\mathbf{P}(X^1 \otimes X^2 \otimes ... \otimes X^n \mid \delta \otimes \pi)$.

Preliminary Knowledge

To specify preliminary knowledge the programmer must undertake the following:

1 Define the set of relevant variables $\{X^1, X^2, ..., X^n\}$ on which the joint distribution is defined.

2 Decompose the joint distribution:

Given a partition of $\{X^1, X^2, ..., X^n\}$ into k subsets we define k variables $L^1, ..., L^k$ each corresponding to one of these subsets.

Each variable L^i is obtained as the conjunction of the variables $\{X^{i_1}, X^{i_2}, ...\}$ belonging to the subset i. The conjunction rules [E3.6] leads to:

$$\mathbf{P}(X^1 \otimes X^2 \otimes \ldots \otimes X^n \mid \delta \otimes \pi) \quad [E3.12]$$
$$= \mathbf{P}(L^1 \mid \delta \otimes \pi) \times \mathbf{P}(L^2 \mid L^1 \otimes \delta \otimes \pi) \times \ldots \times \mathbf{P}(L^k \mid L^{k-1} \otimes \ldots \otimes L^2 \otimes L^1 \otimes \delta \otimes \pi)$$

Conditional independence hypotheses then allow further simplifications. A conditional independence hypothesis for variable L^i is defined by picking some variables X^j among the variables appearing in conjunction $L^{i-1} \otimes \ldots \otimes L^2 \otimes L^1$, calling R^i the conjunction of these chosen variables and setting:

$$\mathbf{P}(L^i \mid L^{i-1} \otimes \ldots \otimes L^2 \otimes L^1 \otimes \delta \otimes \pi) = \mathbf{P}(L^i \mid R^i \otimes \delta \otimes \pi) \quad [E3.13]$$

We then obtain:

$$\mathbf{P}(X^1 \otimes X^2 \otimes \ldots \otimes X^n \mid \delta \otimes \pi) \quad [E3.14]$$
$$= \mathbf{P}(L^1 \mid \delta \otimes \pi) \times \mathbf{P}(L^2 \mid R^2 \otimes \delta \otimes \pi) \times \mathbf{P}(L^3 \mid R^3 \otimes \delta \otimes \pi) \times \ldots \times \mathbf{P}(L^k \mid R^k \otimes \delta \otimes \pi)$$

Such a simplification of the joint distribution as a product of simpler distributions is called a decomposition.

3 Define the forms:

Each distribution $\mathbf{P}(L^i \mid R^i \otimes \delta \otimes \pi)$ appearing in the product is then associated with either a parametric form (i.e., a function $f_\mu(L^i)$) or another Bayesian program. In general, μ is a vector of parameters that may depend on R^i or δ or both. Learning takes place when some of these parameters are computed using the data set δ

Question

Given a description (i.e., $\mathbf{P}(X^1 \otimes X^2 \otimes \ldots \otimes X^n \mid \delta \otimes \pi)$), a question is obtained by partitioning $\{X^1, X^2, \ldots\ldots, X^n\}$ into three sets : the searched variables, the known variables and the unknown variables.

We define the variables $Search$, $Known$ and $Unknown$ as the conjunction of the variables belonging to these sets. We define a question as the distribution:

$$\mathbf{P}(Searched \mid Known \otimes \delta \otimes \pi). \quad [E3.15]$$

3.4 Running Bayesian programs

Running a Bayesian program supposes two basic capabilities: Bayesian inference and decision-making.

Bayesian inference

Given the joint distribution $\mathbf{P}(X^1 \otimes X^2 \otimes \ldots \otimes X^n \mid \delta \otimes \pi)$, it is always possible to compute any possible question, using the following general inference:

$$
\begin{aligned}
\mathbf{P}(Searched \mid Known \otimes \delta \otimes \pi) &= \sum_{Unknown} \mathbf{P}(Searched \otimes Unknown \mid Known \otimes \delta \otimes \pi) \\
&= \frac{\sum_{Unknown} \mathbf{P}(Searched \otimes Unknown \otimes Known \mid \delta \otimes \pi)}{\mathbf{P}(Known \mid \delta \otimes \pi)} \\
&= \frac{\sum_{Unknown} \mathbf{P}(Searched \otimes Unknown \otimes Known \mid \delta \otimes \pi)}{\sum_{\substack{Searched \\ Unknown}} \mathbf{P}(Searched \otimes Unknown \otimes Known \mid \delta \otimes \pi)} \\
&= \frac{1}{\Sigma} \times \sum_{Unknown} \mathbf{P}(Searched \otimes Unknown \otimes Known \mid \delta \otimes \pi)
\end{aligned}
\qquad [E3.16]
$$

where the first equality results from the marginalization rule (equation [E3.10]), the second results from the product rule (equation [E3.6]) and the third corresponds to a second application of the marginalization rule. The denominator appears to be a normalization term. Consequently, by convention, we will replace it by Σ.

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 NP-hard (Cooper, 1990) and the general problem of approximate inference too (Dagum & Luby, 1993). Numerous heuristics and restrictions to the generality of the possible inferences have been proposed to achieve admissible computation time (see the papers already cited in Section 2 about technical progress in this area).

An inference engine and the associated programming API[7] as been developed and used for the experiments presented in this paper. The same API has also been used for other applications such as CAD modeling (see Mekhnacha, 1999).

Our engine proceeds in two phases: a symbolic simplification phase which reduces the complexity of the considered sums, and a numeric phase that computes an approximation of the distributions.

Our symbolic simplification phase drastically reduces the number of sums necessary to compute a given distribution. However the decomposition part of the preliminary knowledge, which expresses the conditional dependencies of variables, still plays a crucial role in keeping the computation tractable. The importance of the decomposition has already been stressed by many authors (e.g., Zhang & Poole, 1996) and explains the good performances of our engine (10 inferences per second[8]).

Decision-making

For a given distribution, different decision policies are possible: for example, searching the best (highest probability) values or drawing at random according to the distribution. For our purposes, we will always use this second policy and refer this query as: $\mathbf{Draw}(\mathbf{P}(Searched \mid Known \otimes \delta \otimes \pi))$.

Control loop of the robot

To control our robot using a Bayesian program, a decision is made every tenth of a second. A typical question is to select the values of the motor variables knowing the values of

7. Application Programming Interface

8. Order of magitude on a standard desk top computer for the inferences required by the experiments described in the sequel.

the sensory variables. Consequently, the basic loop to operate the robot is to loop on the following instructions every tenth of a second:

```
1 - Read the values of the sensors
2 -  Draw(P(Motors | Sensors ⊗ δ ⊗ π))
3 - Send the returned values to the motors
```

4. Experimental platform

4.1 Khepera robot

Khepera is a two-wheeled mobile robot, 57 millimeters in diameter and 29 millimeters in height, with a total weight of 80g (See Figure 2). It was designed at EPFL[9] and is commercialized by K-Team[10].

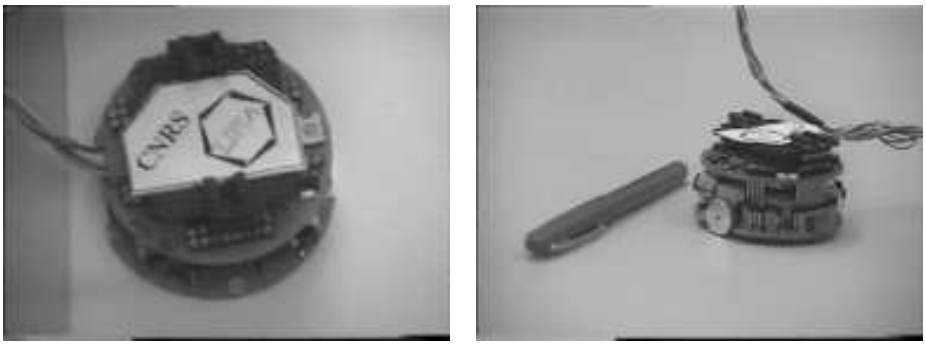


Figure 2: The Khepera mobile robot (from the top and from the left)

The robot is equipped with eight light sensors (six in front and two behind), taking values between 0 and 511 in inverse relation to light intensity, stored in variables $L1, \ldots, L8$ (see Figure 3). These eight sensors can also be used as infrared proximeters, taking values between 0 and 1023 in inverse relation to the distance from the obstacle, stored in variables $Px1, \ldots, Px8$ (see Figure 3).

The robot is controlled by the rotation speeds of its left and right wheels, stored in variables *Mg* and *Md*, respectively.

From these 18 basic sensory and motor variables, we derived three new sensory variables (Dir, $Prox$ and $Theta1$) and one new motor one ($Vrot$). They are described below.

- Dir is a variable that approximately corresponds to the bearing of the closest obstacle (See Figure 3). It takes values between -10 (obstacle to the left of the robot) and +10 (obstacle to the right of the robot), and is defined as follows:

9. Ecole Polytechnique Fédérale de Lausane (Switzerland)
10. http://www.K-team.com/

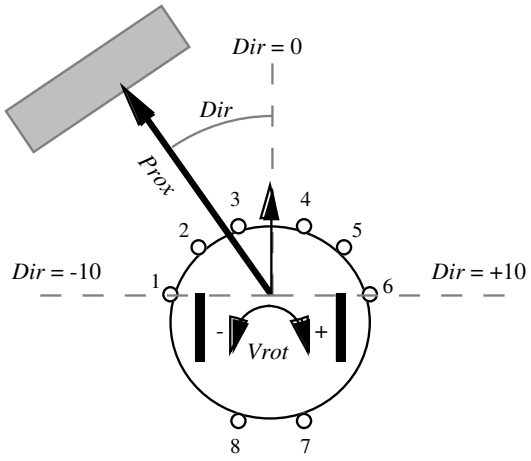


Figure 3: The sensory-motor variables of the Khepera robot.

$$Dir = \mathbf{Floor}\left(\frac{90(Px6 - Px1) + 45(Px5 - Px2) + 5(Px4 - Px3)}{9(1 + Px1 + Px2 + Px3 + Px4 + Px5 + Px6)}\right) \quad \text{[E4.1]}$$

- *Prox* is a variable that approximately corresponds to the proximity of the closest obstacle (See Figure 3). It takes values between zero (obstacle very far from the robot) and 15 (obstacle very close to the robot), and is defined as follows:

$$Prox = \mathbf{Floor}\left(\frac{\mathbf{Max}(Px1, Px2, Px3, Px4, Px5, Px6)}{64}\right) \quad \text{[E4.2]}$$

- *Theta1* is a variable that approximately corresponds to the bearing of the greatest source of illumination. It takes on 36 values from -170° to 180°.
- The robot is piloted solely by its rotation speed (the translation speed is fixed). It receives motor commands from the *Vrot* variable, calculated from the difference between the rotation speeds of the left and right wheels. *Vrot* takes on values between -10 (fastest to the left) and +10 (fastest to the right).

Khepera accepts turrets on its top to augment either its sensory or motor capacities. For the final experiment (the nightwatchman task), a linear camera of 64 pixels and a micro turbine were added on top of the robot.

4.2 Environment

For all experiments described in the current paper, the Khepera is placed in a 1 m by 1 m environment. This environment has walls around its contour, textured to be easily seen by the robot. Inside this square, we place walls made of Lego© bricks that can be moved easily to set any configuration we need quickly. We usually build a recess made of high Lego© walls in a corner, and place a small light over this recess, to create a «base» for the robot.

5. Reactive behavior

5.1 Goal and experimental protocol

The goal of the first experiment was to teach the robot how to push objects.

First, in a learning phase, we drove the robot with a joystick to push objects. During that phase, the robot collected, every tenth of a second, both the values of its sensory variables and the values of its motor variables (determined by the joystick position). This data set was then used to identify the free parameters of the parametric forms.

Then, in a restitution phase, the robot has to reproduce the behavior it had just learned. Every tenth of a second it decided the values of its motor variables, knowing the values of its sensory variables and the internal representation of the task.

5.2 Specification

Having defined our goal, we describe the three steps necessary to define the preliminary knowledge.

```
1 - Chose the pertinent variables

2 - Decompose the joint distribution

3 - Define the Parametric forms
```

Variables

First, the programmer specifies which variables are pertinent for the task.

To push objects it is necessary to have an idea of the position of the objects relative to the robot. The front proximeters provide this information. However, we chose to sum up the information of these six proximeters by the two variables Dir and $Prox$.

We also chose to set the translation speed to a constant and to operate the robot by its rotation speed $Vrot$.

These three variables are all we need to push obstacles. Their definitions are summed up as follows:

$$\begin{array}{c} Dir \in \{-10, \ldots, 10\}, \lfloor Dir \rfloor = 21 \\ Prox \in \{0, \ldots, 15\}, \lfloor Prox \rfloor = 16 \\ Vrot \in \{-10, \ldots, 10\}, \lfloor Vrot \rfloor = 21 \end{array} \quad [S5.1]$$

Decomposition

In the second specification step, we give a decomposition of the joint probability $\mathbf{P}(Dir \otimes Prox \otimes Vrot \mid \Delta \otimes \pi\text{-}obstacle)$ as a product of simpler terms. This distribution is conditioned by both $\pi\text{-}obstacle$, the preliminary knowledge we are defining, and Δ a data set that will be provided during the learning phase.

$$\begin{array}{l} \mathbf{P}(Dir \otimes Prox \otimes Vrot \mid \Delta \otimes \pi\text{-}obstacle) \\ = \mathbf{P}(Dir \mid \Delta \otimes \pi\text{-}obstacle) \times \mathbf{P}(Prox \mid Dir \otimes \Delta \otimes \pi\text{-}obstacle) \times \mathbf{P}(Vrot \mid Prox \otimes Dir \otimes \Delta \otimes \pi\text{-}obstacle) \\ = \mathbf{P}(Dir \mid \Delta \otimes \pi\text{-}obstacle) \times \mathbf{P}(Prox \mid \Delta \otimes \pi\text{-}obstacle) \times \mathbf{P}(Vrot \mid Prox \otimes Dir \otimes \Delta \otimes \pi\text{-}obstacle) \end{array} \quad [S5.2]$$

The first equality results from the application of the product rule (equation [E3.6]). The

second results from the simplification $\mathbf{P}(Prox \mid Dir \otimes \Delta \otimes \pi\text{-}obstacle) = \mathbf{P}(Prox \mid \Delta \otimes \pi\text{-}obstacle)$, which means that we consider that $Prox$ and Dir are independent. The distances to the objects and their bearings are not contingent.

Parametric forms

To be able to compute the joint distribution, we finally need to assign parametric forms to each of the terms appearing in the decomposition:

$$\begin{aligned} \mathbf{P}(Dir \mid \Delta \otimes \pi\text{-}obstacle) &\equiv Uniform \\ \mathbf{P}(Prox \mid \Delta \otimes \pi\text{-}obstacle) &\equiv Uniform \\ \mathbf{P}(Vrot \mid Prox \otimes Dir \otimes \Delta \otimes \pi\text{-}obstacle) &\equiv \mathbf{G}(\mu(Prox, Dir), \sigma(Prox, Dir)) \end{aligned} \qquad [S5.3]$$

We have no *a priori* information about the direction and the distance of the obstacles. Hence, $\mathbf{P}(Dir \mid \Delta \otimes \pi\text{-}obstacle)$ and $\mathbf{P}(Prox \mid \Delta \otimes \pi\text{-}obstacle)$ are uniform distributions; all directions and proximities have the same probability.

For each sensory situation, we believe that there is one and only one rotation speed that should be preferred. The distribution $\mathbf{P}(Vrot \mid Prox \otimes Dir \otimes \Delta \otimes \pi\text{-}obstacle)$ is unimodal. However, depending of the situation, the decision to be made for $Vrot$ may be more or less certain. This is resumed by assigning a Gaussian parametrical form to $\mathbf{P}(Vrot \mid Prox \otimes Dir \otimes \Delta \otimes \pi\text{-}obstacle)$.

5.3 Identification

We drive the robot with a joystick (see Movie 1[11]), and collect a set of data Δ. Let us call the particular set of data corresponding to this experiment $\delta\text{-}push$. A datum collected at time t is a triplet $(vrot_t, dir_t, prox_t)$.

The free parameters of the parametric forms (means and standard deviations for all the $\lfloor Dir \rfloor \times \lfloor Prox \rfloor$ Gaussians) can then be identified.

Finally, it is possible to compute the joint distribution:

$$\begin{aligned} &\mathbf{P}(Dir \otimes Prox \otimes Vrot \mid \delta\text{-}push \otimes \pi\text{-}obstacle) \\ &= \mathbf{P}(Dir \mid \pi\text{-}obstacle) \times \mathbf{P}(Prox \mid \pi\text{-}obstacle) \times \mathbf{P}(Vrot \mid Prox \otimes Dir \otimes \delta\text{-}push \otimes \pi\text{-}obstacle) \end{aligned} \qquad [E5.1]$$

According to equation [E3.16], the robot can answer any question concerning this joint distribution.

We call the distribution $\mathbf{P}(Dir \otimes Prox \otimes Vrot \mid \delta\text{-}push \otimes \pi\text{-}obstacle)$ a *description* of the task. A description is the result of identifying the free parameters of a preliminary knowledge using some given data. Hence, a description is completely defined by a couple preliminary knowledge + data. That is why a conjunction $\delta \otimes \pi$ always appears to the right of a description.

5.4 Utilization

To render the pushing obstacle behavior just learned, the Bayesian controller is called every tenth of a second :

1 - The sensors are read and the values of dir_t and $prox_t$ are computed

2 - The Bayesian program is run with the query:

11. http://www-leibniz.imag.fr/LAPLACE/Cours/Semaine-Science/Trans7/T7.mov (QuickTime, 4.4 Mo)

$$\mathbf{Draw}(\mathbf{P}(Vrot \mid prox_t \otimes dir_t \otimes \delta\text{-}push \otimes \pi\text{-}obstacle)) \quad [E5.2]$$

3 - The drawn $vrot_t$ is sent to the motors

5.5 Results, lessons and comments

Results

As shown in Movie 1[11], the Khepera learns how to push obstacles in 20 to 30 seconds. It learns the particular dependency, corresponding to this specific behavior, between the sensory variables Dir and $Prox$ and the motor variable $Vrot$.

This dependency is largely independent of the particular characteristics of the objects (weight, color, balance, nature, etc.). Therefore, as shown in Movie 2[12], the robot is also able to push different objects. This, of course, is only true within certain limits. For instance, the robot will not be able to push the object if it is too heavy.

Method

In this experiment we apply a precise three-step method to program the robot.

1 - *Specification*: define the preliminary knowledge

  1.1 - Choose the pertinent variables

  1.2 - Decompose the joint distribution

  1.3 - Define the Parametric forms

2 - *Identificationn*:identify the free parameters of the preliminary knowledge

3 - *Utilization*: ask a question to the joint distribution

In the sequel, we will use the very same method for all the other experiments.

Variations

Numerous different behaviors may be obtained by changing some of the different components of a Bayesian program in the following ways.

- It is possible to *change the question*, keeping the description unchanged. For instance, if the $Prox$ information is no longer available because of some failure, the robot may still try to push the obstacles knowing only their direction. The query is then:

$$\mathbf{Draw}(\mathbf{P}(Vrot \mid dir_t \otimes \delta\text{-}push \otimes \pi\text{-}obstacle)) \quad [E5.3]$$

- It is possible to *change the data*, keeping the preliminary knowledge unchanged. For instance, with the same preliminary knowledge $\pi\text{-}obstacle$, we taught the robot to avoid objects or to follow their contour (see Figure 4 and Movie 3[13]). Two new descriptions[14] were obtained by changing only the driving of the robot during the

12. http://www-leibniz.imag.fr/LAPLACE/Cours/Semaine-Science/Trans8/T8.mov (QuickTime, 1Mo)
13. http://www-leibniz.imag.fr/LAPLACE/Cours/Semaine-Science/Trans9/T9.mov (QuickTime, 3.6Mo)
14. $\mathbf{P}(Dir \otimes Prox \otimes Vrot \mid \delta\text{-}avoid \otimes \pi\text{-}obstacle)$ and $\mathbf{P}(Dir \otimes Prox \otimes Vrot \mid \delta\text{-}follow \otimes \pi\text{-}obstacle)$

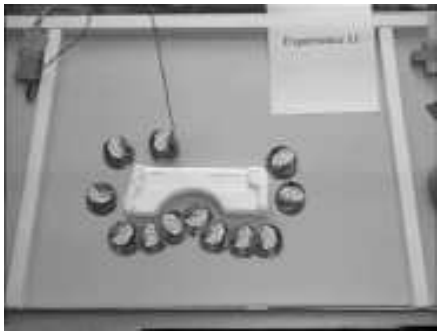


Figure 4: Contour following (superposed images)

learning phase. As a result, two new programs were obtained leading to the expected behaviors : «obstacle avoidance» and «contour following».

- Finally, it is possible to *change the preliminary knowledge,* which leads to completely different behaviors. Numerous examples will be presented in the sequel of this paper. For instance, we taught the robot another reactive behavior called phototaxy. Its goal is then to move toward a light source. This new preliminary knowledge π-*phototaxy1* uses the variables *Vrot* and *Theta1* . *Theta1* roughly corresponds to the direction of the light.

6. Behavior combination

6.1 Goal and experimental protocol

In this experiment we want the robot to go back to its base where it can recharge.

This will be obtained with no further teaching. As the robot's base is lit, the light gradient usually gives good hints on its direction. Consequently, we will obtain the homing behavior by combining together the obstacle avoidance behavior and the phototaxy behavior. By programming this behavior we will illustrate one possible way to combine Bayesian programs that make use of «command variable».

6.2 Specification

Variables

We need *Dir* , *Prox* , *Theta1* and *Vrot* , the four variables already used in the two composed behaviors. We also need a new variable *H* which acts as a command to switch from avoidance to phototaxy.

$$\begin{array}{c} Dir \in \{-10, \ldots, 10\}, \lfloor Dir \rfloor = 21 \\ Prox \in \{0, \ldots, 15\}, \lfloor Prox \rfloor = 16 \\ Theta1 \in \{-170, \ldots, 180\}, \lfloor Theta1 \rfloor = 36 \\ Vrot \in \{-10, \ldots, 10\}, \lfloor Vrot \rfloor = 21 \\ H \in \{avoidance, phototaxy\}, \lfloor H \rfloor = 2 \end{array} \quad [S6.1]$$

Decomposition

We believe that the sensory variables Dir, $Prox$ and $Theta1$ are independent from one another. Far from any objects, we want the robot to go toward the light. Very close to obstacles, we want the robot to avoid them. Hence, we consider that H should only depend on $Prox$. Finally, we believe that $Vrot$ must depend on the other four variables. These programmer choices lead to the following decomposition:

$$\begin{array}{l} \mathbf{P}(Dir \otimes Prox \otimes Theta1 \otimes H \otimes Vrot \mid \Delta \otimes \pi\text{-}home) \\ = \mathbf{P}(Dir \mid \pi\text{-}home) \times \mathbf{P}(Prox \mid \pi\text{-}home) \times \mathbf{P}(Theta1 \mid \pi\text{-}home) \times \mathbf{P}(H \mid Prox \otimes \pi\text{-}home) \\ \times \mathbf{P}(Vrot \mid Dir \otimes Prox \otimes Theta1 \otimes H \otimes \pi\text{-}home) \end{array} \quad [S6.2]$$

Parametric forms

We have no *a priori* information about either the direction and distance of objects or the direction of the light source. Consequently, we state:

$$\begin{array}{c} \mathbf{P}(Dir \mid \pi\text{-}home) \equiv Uniform \\ \mathbf{P}(Prox \mid \pi\text{-}home) \equiv Uniform \\ \mathbf{P}(Theta1 \mid \pi\text{-}home) \equiv Uniform \end{array} \quad [S6.3]$$

H is a command variable to switch from avoidance to phototaxy. This means that when $H = avoidance$ the robot should behave as it learned to do in the description $\mathbf{P}(Dir \otimes Prox \otimes Vrot \mid \delta\text{-}avoid \otimes \pi\text{-}obstacle)$ and when $H = phototaxy$ the robot should behave according to the description $\mathbf{P}(Theta1 \otimes Vrot \mid \delta\text{-}phototaxy \otimes \pi\text{-}phototaxy1)$. Therefore, we state:

$$\begin{array}{l} \mathbf{P}(Vrot \mid Dir \otimes Prox \otimes Theta1 \otimes avoidance \otimes \pi\text{-}home) \equiv \mathbf{P}(Vrot \mid Dir \otimes Prox \otimes \delta\text{-}avoid \otimes \pi\text{-}obstacle) \\ \mathbf{P}(Vrot \mid Dir \otimes Prox \otimes Theta1 \otimes phototaxy \otimes \pi\text{-}home) \equiv \mathbf{P}(Vrot \mid Theta1 \otimes \delta\text{-}phototaxy \otimes \pi\text{-}phototaxy1) \end{array} \quad [S6.4]$$

We want a smooth transition from phototaxy to avoidance as we move closer and closer to objects. Hence we finally state:

$$\begin{array}{c} \mathbf{P}(avoidance \mid Prox \otimes \pi\text{-}home) \equiv \mathbf{Sigmoid}_{\alpha,\beta}(Prox) \qquad (\alpha = 9), (\beta = 0,25) \\ \mathbf{P}(phototaxy \mid Prox \otimes \pi\text{-}home) = 1 - \mathbf{P}(avoidance \mid Prox \otimes \pi\text{-}home) \end{array} \quad [S6.5]$$

The discrete approximation of the Sigmoid function we use above, which will not be defined in the current paper, is shown in Figure 5.

The preliminary knowledge $\pi\text{-}home$ is defined by specifications [S6.1], [S6.2], [S6.3], [S6.4] and [S6.5].

6.3 Identification

There are no free parameters in preliminary knowledge $\pi\text{-}home$. No learning is required.

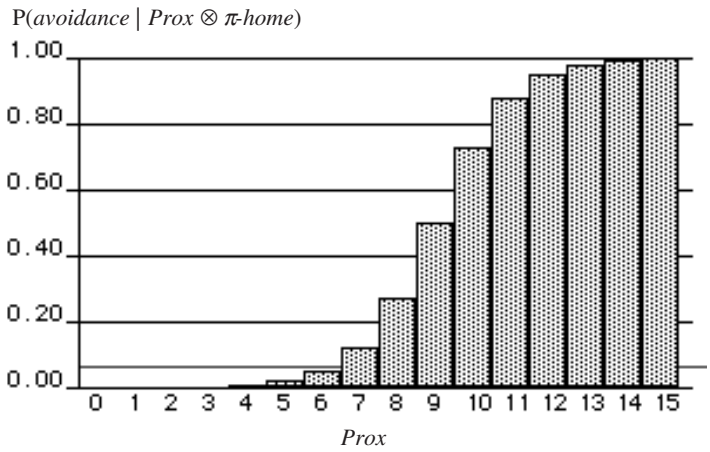


Figure 5: $\mathbf{P}(avoidance \mid Prox \otimes \pi\text{-}home)$

6.4 Utilization

While Khepera returns to its base, we do not know in advance when it should avoid obstacles or when it should go toward the light. Consequently, to render the homing behavior we will use the following question where H is unknown:

$$\begin{aligned}
&\mathbf{P}(Vrot \mid Dir \otimes Prox \otimes Theta1 \otimes \pi\text{-}home)\\
&= \sum_{H} \mathbf{P}(Vrot \otimes H \mid Dir \otimes Prox \otimes Theta1 \otimes \pi\text{-}home)\\
&= \frac{1}{\Sigma} \times \left[\begin{array}{l} [\mathbf{P}(avoidance \mid Prox \otimes \pi\text{-}home) \times \mathbf{P}(Vrot \mid Dir \otimes Prox \otimes \delta\text{-}avoid \otimes \pi\text{-}obstacle)] \\ + [\mathbf{P}(phototaxy \mid Prox \otimes \pi\text{-}home) \times \mathbf{P}(Vrot \mid Theta1 \otimes \delta\text{-}phototaxy \otimes \pi\text{-}phototaxy1)] \end{array} \right]
\end{aligned} \qquad \text{[E6.1]}$$

Equation [E6.1] shows that the robot does a weighted combination between avoidance and phototaxy. Far from any objects ($prox = 0, \mathbf{P}(phototaxy \mid prox \otimes \pi\text{-}home) = 1$) it does pure phototaxy. Very close to objects ($prox = 15, \mathbf{P}(avoidance \mid prox \otimes \pi\text{-}home) = 1$) it does pure avoidance. In between, it mixes the two.

6.5 Results, lessons and comments

Results

Figure 6 and Movie 4[15] show efficient homing behavior obtained this way.

Figures 7 and 8 present the probability distributions obtained when the robot must avoid an obstacle on the left with a light source also on the left. As the object is on the left, the robot needs to turn right to avoid it. This is what happens when the robot is close to the objects (see Figure 7). However, when the robot is further from the object, the presence of the light source on the left influences the way the robot avoids obstacles. In that case, the robot may turn left despite the presence of the obstacle (see Figure 8).

15. http://www-leibniz.imag.fr/LAPLACE/Cours/Semaine-Science/Trans10/T10.mov (QuickTime, 4.3Mo)

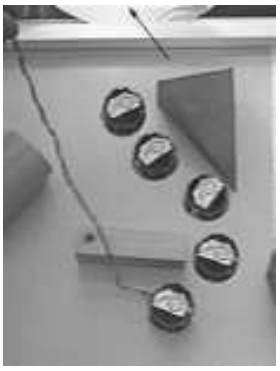


Figure 6: Homing behavior (The arrow points out the light source) (superposed images).

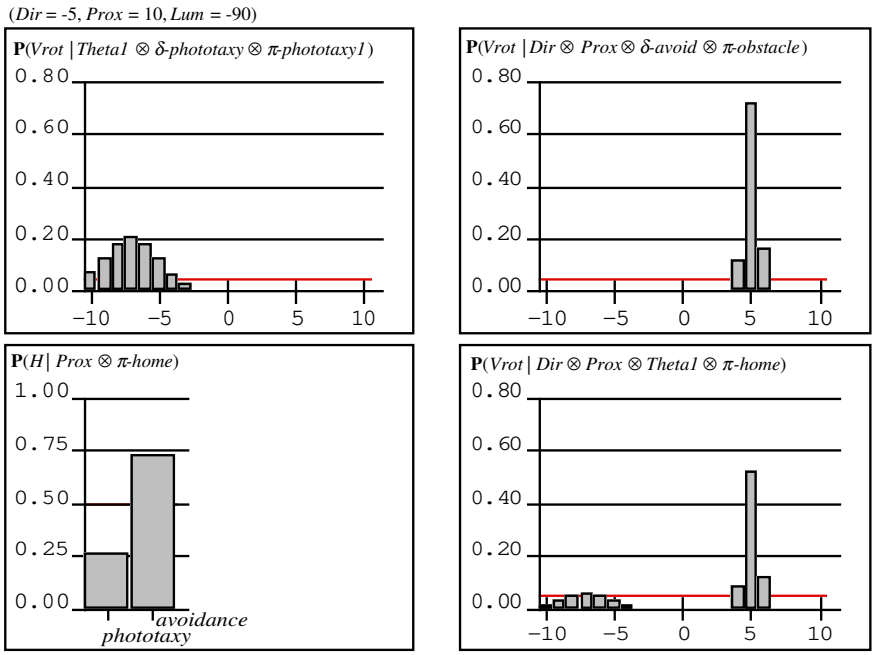


Figure 7: Homing behavior (Khepera close to an object on its left). The top left distribution shows the knowledge on $Vrot$ given by the phototaxy description; the top right is $Vrot$ given by the avoidance description; the bottom left shows the knowledge of the «command variable» H; finally the bottom right shows the resulting combination on $Vrot$.

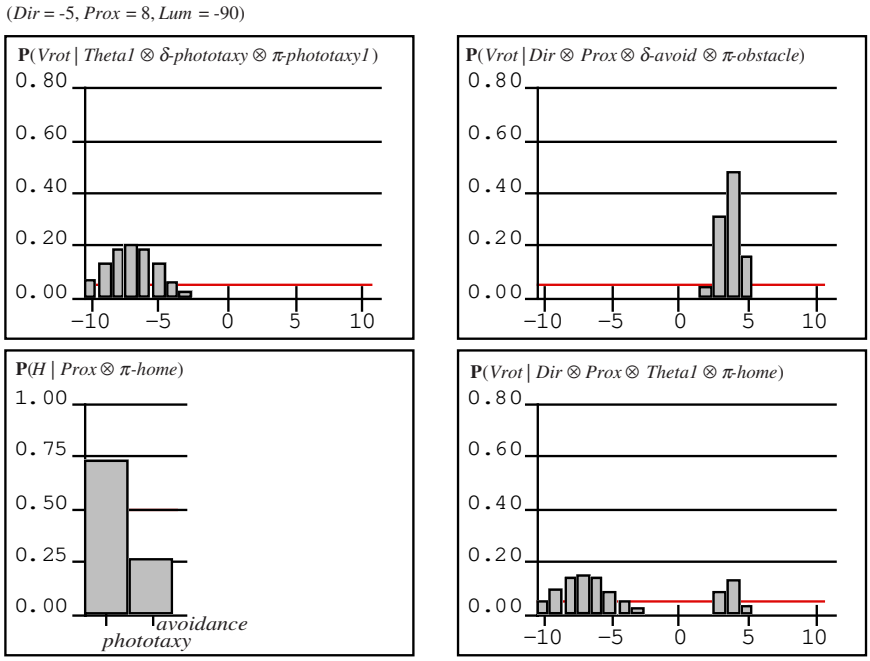


Figure 8: Homing behavior (Khepera further from the object on its left).
This figure is structured as Figure 7.

Descriptions combination method

In this experiment we present a simple instance of a general method to combine descriptions to obtain a new mixed behavior. This method uses a command variable H to switch from one of the composing behaviors to another. A probability distribution on H knowing some sensory variables should then be specified or learned[16]. The new description is finally used by asking questions where H is unknown. The resulting sum on the different cases of H does the mixing.

This shows that Bayesian robot programming allows easy, clear and rigorous specifications of such combinations. This seems to be an important benefit compared to some other methods that have great difficulties in mixing behaviors with one another, such as Brooks' subsumption architecture (Brooks, 1986; Maes, 1989) or neural networks. Description combination appears to naturally implement a mechanism similar to HEM[17] (Jordan & Jacobs, 1994).

7. Sensor fusion

7.1 Goal and experimental protocol

The goal of this experiment is to fuse the data originating from the eight light sensors to

16. see (Diard & Lebeltel, 1999)
17. Hierachical Mixture of Expert

determine the position of a light source.

This will be obtained in two steps. In the first one, we specify one description for each sensor individually. In the second one, we mix these eight descriptions to form a global one.

7.2 Sensor model

Specification

Variables

To build a model of the light sensor i, we only require two variables: Li the reading of the i^{th} sensor, and $Theta2$, the bearing of the light source.

$$\begin{array}{c} Li \in \{0, \ldots, 511\}, \lfloor Li \rfloor = 512 \\ Theta2 \in \{-170, \ldots, 180\}, \lfloor Theta2 \rfloor = 36 \end{array} \quad [S7.1]$$

Decomposition

The decomposition simply specifies that the reading of a sensor obviously depends on the position of the light source

$$\begin{array}{l} \mathbf{P}(Theta2 \otimes Li \mid \Delta \otimes \pi\text{-}sensor) \\ = \mathbf{P}(Theta2 \mid \pi\text{-}sensor) \times \mathbf{P}(Li \mid Theta2 \otimes \Delta \otimes \pi\text{-}sensor) \end{array} \quad [S7.2]$$

Parametric forms

As we have no *a priori* information on the position of the source, we state:

$$\mathbf{P}(Theta2 \mid \pi\text{-}sensor) \equiv Uniform \quad [S7.3]$$

The distribution $\mathbf{P}(Li \mid Theta2 \otimes \Delta \otimes \pi\text{-}sensor)$ is usually very easy to specify because it corresponds exactly to the kind of information that the sensor supplier provides: the expected readings of its device when exposed to a light. For the Khepera's light sensors, we obtain (see Figure 9):

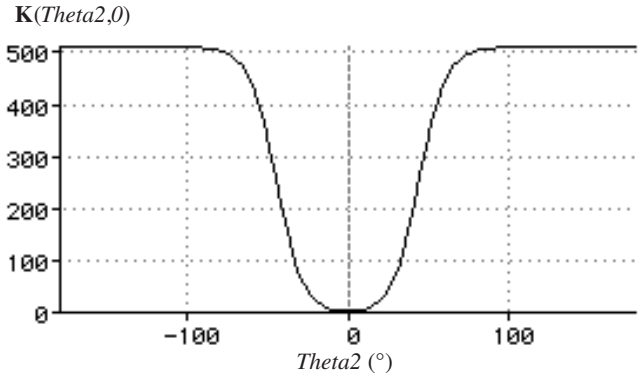


Figure 9: $\mathbf{K}(Theta2, 0)$

$$\begin{aligned} &\mathbf{P}(Li \mid Theta2 \otimes \pi\text{-}sensor) \equiv \mathbf{G}_{\mathbf{K}(Theta2,\, \theta_i),\, \sigma}(Li) \\ &\mathbf{K}(Theta2, \theta_i) \;=\; 1 - \frac{1}{1 + e^{-4\beta(|Theta2 - \theta_i| - \alpha)}} \qquad (\alpha = 45), (\beta = 0{,}03) \end{aligned} \tag{S7.4}$$

In specification [S7.4], θ_i stands for the position of the sensor with respect to the robot, and will be used later to «rotate» this model for different sensors.

Specifications [S7.1], [S7.2], [S7.3] and [S7.4] are the preliminary knowledge corresponding to this sensor model. This preliminary knowledge is named $\pi\text{-}sensor$.

Identification

No identification is required as there are no free parameters in $\pi\text{-}sensor$.

However, it may be easy and interesting to calibrate specifically each of the eight light sensors. This could be achieved, for instance, by identifying parameters α and β independently for each sensor, by observing the response of the particular sensor to a light source.

7.3 Fusion

Specification

Variables

The interesting variables are the eight variables Li and $Theta2$:

$$\begin{aligned} &L1 \in \{0, \ldots, 511\}, \lfloor L1 \rfloor = 512 \\ &\ldots \\ &L8 \in \{0, \ldots, 511\}, \lfloor L8 \rfloor = 512 \\ &Theta2 \in \{-170, \ldots, 180\}, \lfloor Theta2 \rfloor = 36 \end{aligned} \tag{S7.5}$$

Decomposition

The decomposition of the joint distribution is chosen to be:

$$\begin{aligned} &\mathbf{P}(Theta2 \otimes L1 \otimes L2 \otimes L3 \otimes L4 \otimes L5 \otimes L6 \otimes L7 \otimes L8 \mid \Delta \otimes \pi\text{-}fusion) \\ &= \mathbf{P}(Theta2 \mid \Delta \otimes \pi\text{-}fusion) \times \mathbf{P}(L1 \mid Theta2 \otimes \Delta \otimes \pi\text{-}fusion) \times \mathbf{P}(L2 \mid L1 \otimes Theta2 \otimes \Delta \otimes \pi\text{-}fusion) \\ &\ldots \times \mathbf{P}(L8 \mid L7 \otimes L6 \otimes L5 \otimes L4 \otimes L3 \otimes L2 \otimes L1 \otimes Theta2 \otimes \Delta \otimes \pi\text{-}fusion) \\ &= \mathbf{P}(Theta2 \mid \pi\text{-}fusion) \times \mathbf{P}(L1 \mid Theta2 \otimes \Delta \otimes \pi\text{-}fusion) \times \mathbf{P}(L2 \mid Theta2 \otimes \Delta \otimes \pi\text{-}fusion) \\ &\ldots \times \mathbf{P}(L8 \mid Theta2 \otimes \Delta \otimes \pi\text{-}fusion) \end{aligned} \tag{S7.6}$$

The first equality results from the product rule [E3.6]. The second from simplifications of the kind:

$$\mathbf{P}(Lj \mid Lj - 1 \otimes \ldots \otimes L1 \otimes Theta2 \otimes \Delta \otimes \pi\text{-}fusion) \;=\; \mathbf{P}(Lj \mid Theta2 \otimes \Delta \otimes \pi\text{-}fusion) \tag{E7.1}$$

These simplifications may seem peculiar as obviously the readings of the different light sensors are not independent. The exact meaning of these equations is that we consider $Theta2$ (the position of the light source) to be the main reason for the contingency of the readings. Consequently, we state that, knowing $Theta2$, the readings Lj are independent. $Theta2$ is the cause of the readings and knowing the cause, the consequences are independent. This is, indeed, a very strong hypothesis. The snesors may be correlated for numerous other reasons.

For instance, ambient temperature influences the functioning of any electronic device and consequently correlates their responses. However, we choose, as a first approximation, to disregard all these other factors.

Parametric forms

We do not have any *a priori* information on $Theta2$:

$$\mathbf{P}(Theta2 \mid \pi\text{-}fusion) \equiv Uniform \qquad [S7.7]$$

$\mathbf{P}(Li \mid Theta2 \otimes \Delta \otimes \pi\text{-}fusion)$ is obtained from the model of each sensor as specified in previous section (7.2):

$$\mathbf{P}(Li \mid Theta2 \otimes \Delta \otimes \pi\text{-}fusion) \equiv \mathbf{P}(Li \mid Theta2 \otimes \pi\text{-}sensor) \qquad [S7.8]$$

Identification

As there are no free parameters in $\pi\text{-}fusion$, no identification is required.

Utilization

To find the position of the light source the standard query is:

$$\mathbf{Draw}(\mathbf{P}(Theta2 \mid l1_t \otimes ..... \otimes l8_t \otimes \pi\text{-}fusion)) \qquad [E7.1]$$

This question may be easily answered using equation [E3.16] and specification [S7.8]:

$$\mathbf{P}(Theta2 \mid l1_t \otimes \ldots \otimes l8_t \otimes \pi\text{-}fusion) = \frac{1}{\Sigma} \times \prod_{i=1}^{8} \mathbf{P}(li_t \mid Theta2 \otimes \pi\text{-}sensor) \qquad [E7.2]$$

Values drawn from this distribution may be efficiently computed given that the distribution $\mathbf{P}(Theta2 \mid l1_t \otimes \ldots \otimes l8_t \otimes \pi\text{-}fusion)$ is simply a product of eight very simple ones, and given that the normalizing constant Σ does not need to be computed for a random draw.

Many other interesting questions may be asked of this description, as the following:

- It is possible to search for the position of the light source knowing only the readings of a few sensors:

$$\mathbf{P}(Theta2 \mid l1_t \otimes l2_t \otimes \pi\text{-}fusion) = \frac{1}{\Sigma} \times \mathbf{P}(l1_t \mid Theta2 \otimes \pi\text{-}sensor) \times \mathbf{P}(l2_t \mid Theta2 \otimes \pi\text{-}sensor) \qquad [E7.3]$$

- It is possible to check whether the sensor i is out of order. Indeed, if its reading li_t at time t, persists in being inconsistent with the readings of the others for some period, it is a good indication of a malfunction. This inconsistency may be detected by a very low probability for li_t:

$$\mathbf{P}(l1_t \mid l2_t \otimes \ldots \otimes l8_t \otimes \pi\text{-}fusion) = \frac{1}{\Sigma} \times \sum_{Theta2} \prod_{i=1}^{8} \mathbf{P}(li_t \mid Theta2 \otimes \pi\text{-}sensor) \quad [E7.4]$$

7.4 Results, lessons and comments

Results

Figure 10 presents the result obtained for a light source with a bearing of 10°:

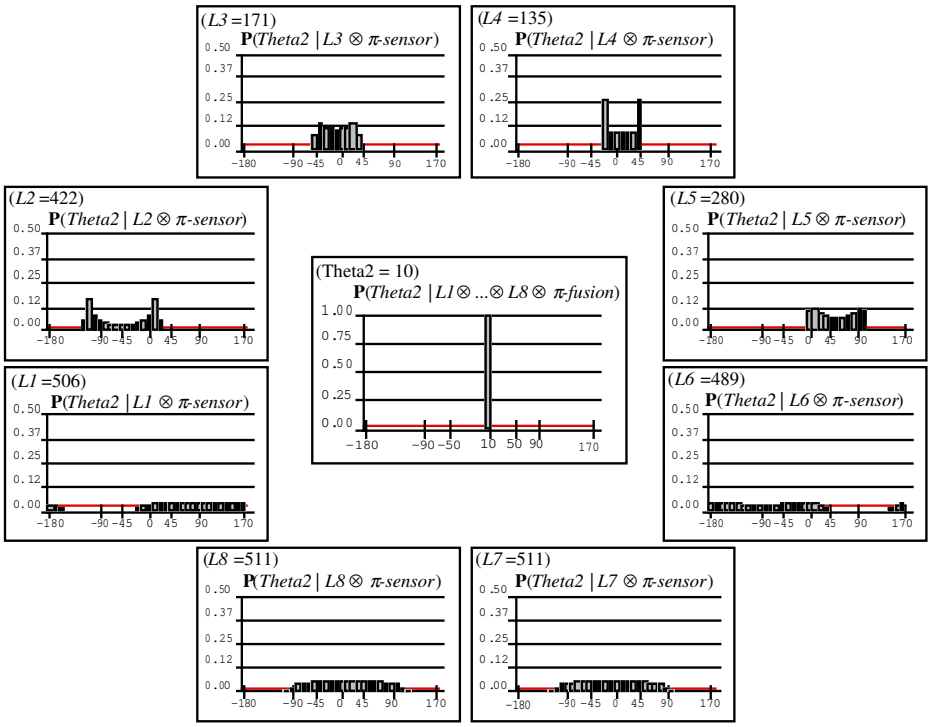


Figure 10: The result of a sensor fusion for a light source with a bearing of 10°

The eight peripheral figures present the distributions $\mathbf{P}(Theta2 \mid Li \otimes \pi\text{-}sensor)$ corresponding to the eight light sensors. The central schema presents the result of the fusion, the distribution $\mathbf{P}(Theta2 \mid l1_t \otimes \ldots \otimes l8_t \otimes \pi\text{-}fusion)$. Even poor information coming from each separate sensor may blend as a certainty.

Sensor fusion method

In the experiment just presented, we have seen a simple instance of a general method to carry out data fusion.

The key point of this method is in the decomposition of the joint distribution, which has been considerably simplified under the hypothesis that «knowing the cause, the consequences are independent». This is a very strong hypothesis, although it may be assumed in

numerous cases.

This way of doing sensor fusion is very efficient. Its advantages are manifold.
- The signal is heightened.
- It is robust to a malfunction of one of the sensors.
- It provides precise information even with poor sensors.
- It leads to simple and efficient computations.

In this experiment, another fundamental advantage of Bayesian programming is clearly evident. The description is neither a direct nor an inverse model. Mathematically, all variables appearing in a joint distribution play exactly the same role. This is why any question may be asked of a description. Furthermore, there is none ill-posed problem. If a question may have several solutions, the probabilistic answer will simply have several peaks.

8. Hierarchical behavior composition

8.1 Goal and experimental protocol

In this experiment, we want to obtain phototaxy behavior based on $Vrot$ and $Theta2$.

We have already built such behavior based on $Vrot$ and $Theta1$, named $\pi\text{-}phototaxy1$. However, as we saw in the previous section (7), $Theta2$ obtained by a sensor fusion has many advantages on $Theta1$ obtained by pretreatment.

8.2 Specification

Variables

The variables we require are the nine variables used in the sensor fusion description $L1, L2, L3, L4, L5, L6, L7, L8, Theta2$ and $Vrot$:

$$\begin{gathered} L1 \in \{0, \ldots, 511\}, \lfloor L1 \rfloor = 512 \\ \ldots \\ L8 \in \{0, \ldots, 511\}, \lfloor L8 \rfloor = 512 \\ Theta2 \in \{-170, \ldots, 180\}, \lfloor Theta2 \rfloor = 36 \\ Vrot \in \{-10, \ldots, 10\}, \lfloor Vrot \rfloor = 21 \end{gathered} \quad [S8.1]$$

Decomposition

The decomposition states that if we know $Theta2$, the position of the light source, then the exact readings of the light sensors do not matter for $Vrot$:

$$\begin{aligned} &\mathbf{P}(Theta2 \otimes L1 \otimes L2 \otimes L3 \otimes L4 \otimes L5 \otimes L6 \otimes L7 \otimes L8 \otimes Vrot \mid \Delta \otimes \pi\text{-}phototaxy2) \\ &= \mathbf{P}(Theta2 \otimes L1 \otimes L2 \otimes L3 \otimes L4 \otimes L5 \otimes L6 \otimes L7 \otimes L8 \mid \pi\text{-}phototaxy2) \times \mathbf{P}(Vrot \mid Theta2 \otimes \pi\text{-}phototaxy2) \end{aligned} \quad [S8.2]$$

Parametric forms

The first distribution is directly obtained using the sensor fusion description:

$$\begin{aligned}&\mathbf{P}(Theta2 \otimes L1 \otimes L2 \otimes L3 \otimes L4 \otimes L5 \otimes L6 \otimes L7 \otimes L8 \mid \pi\text{-}phototaxy2)\\ &\equiv \mathbf{P}(Theta2 \otimes L1 \otimes L2 \otimes L3 \otimes L4 \otimes L5 \otimes L6 \otimes L7 \otimes L8 \mid \pi\text{-}fusion)\end{aligned} \quad [S8.3]$$

The second one is obtained from π-*phototaxy1* as:

$$\mathbf{P}(Vrot \mid Theta2 \otimes \pi\text{-}phototaxy2) \equiv \mathbf{P}(Vrot \mid Theta2 \otimes \pi\text{-}phototaxy1) \quad [S8.4]$$

Specifications [S8.1], [S8.2], [S8.3] and [S8.4] are the components of preliminary knowledge π-*phototaxy2*.

8.3 Identification

No identification is required as there are no free parameters in π-*phototaxy2*.

8.4 Utilization

In order to drive Khepera toward the light with description π-*phototaxy2*, one should answer the following question:

$$\mathbf{P}(Vrot \mid l1_t \otimes \ldots \otimes l8_t \otimes \pi\text{-}phototaxy2) \quad [E8.1]$$

Applying successively the marginalization rule ([E3.10]), the product rule ([E3.6]) and the specifications [S8.3] and [S8.4], we obtain:

$$\begin{aligned}&\mathbf{P}(Vrot \mid L1 \otimes \ldots \otimes L8 \otimes \pi\text{-}phototaxy2)\\ &= \sum_{Theta2} \mathbf{P}(Vrot \otimes Theta2 \mid L1 \otimes \ldots \otimes L8 \otimes \pi\text{-}phototaxy2)\\ &= \sum_{Theta2} \mathbf{P}(Theta2 \mid L1 \otimes \ldots \otimes L8 \otimes \pi\text{-}phototaxy2) \times \mathbf{P}(Vrot \mid Theta2 \otimes \pi\text{-}phototaxy2)\\ &= \sum_{Theta2} \mathbf{P}(Theta2 \mid L1 \otimes \ldots \otimes L8 \otimes \pi\text{-}fusion) \times \mathbf{P}(Vrot \mid Theta2 \otimes \pi\text{-}phototaxy1)\end{aligned} \quad [E8.2]$$

8.5 Results, lessons and comments

Results

The results obtained this way are presented in the three following figures.

Figure 11 presents the results for a light source in front of the robot. The left part shows $\mathbf{P}(Theta2 \mid L1 \otimes \ldots \otimes L8 \otimes \pi\text{-}fusion)$ and the right part $\mathbf{P}(Vrot \mid L1 \otimes \ldots \otimes L8 \otimes \pi\text{-}phototaxy2)$.

Figure 12 presents the results for two light sources, one 90° left of the robot, and the second 90° right. The left part exhibits two symmetrical peaks for $\mathbf{P}(Theta2 \mid L1 \otimes \ldots \otimes L8 \otimes \pi\text{-}fusion)$. Consequently, the right part also shows two symmetrical peaks for $\mathbf{P}(Vrot \mid L1 \otimes \ldots \otimes L8 \otimes \pi\text{-}phototaxy2)$. The robot may decide to turn left or right with equal probabilities.

If we suppose it decided to turn left, at next the time step Khepera will have the left light source 80° to its left and the right one 100° to its right. Figure 13 shows that it has then a high probability of continuing toward the left light source.

Hierarchical composition method

In Section 6 we showed a method of combining different behaviors (descriptions) in order to obtain more complex ones. By contrast, in this experiment we present a method to hierarchically compose descriptions, in order to incrementally obtain more abstract ones.

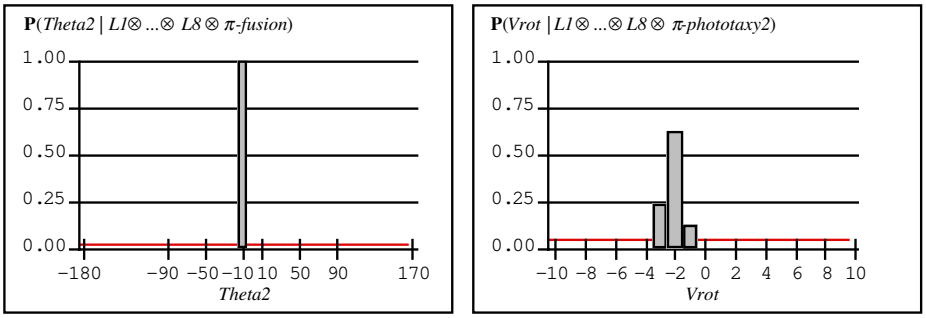


Figure 11: Light source in front of the robot.

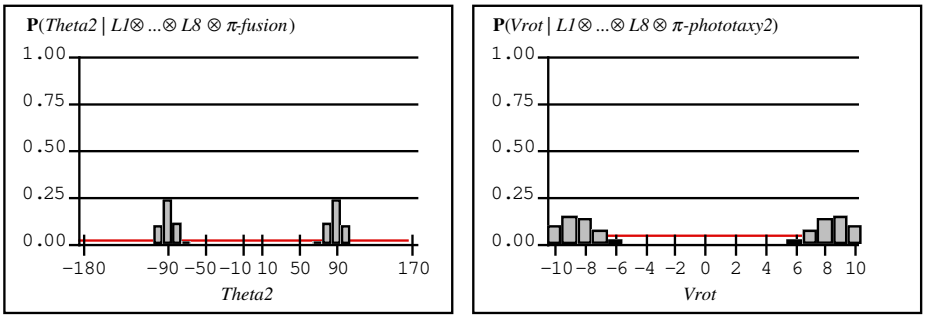


Figure 12: Two light sources: 90° left and 90° right.

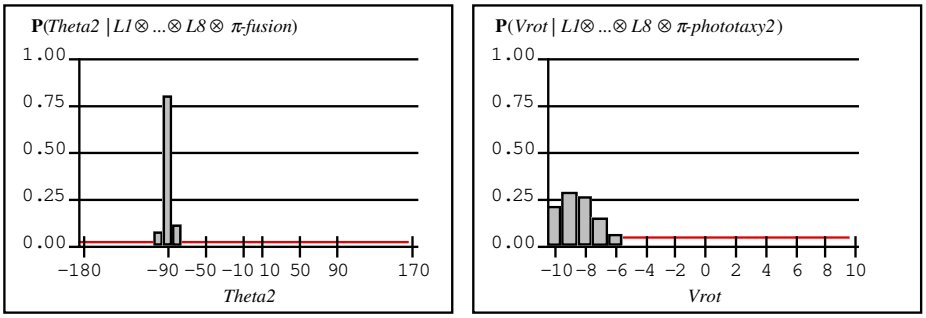


Figure 13: Two light sources: 80° left and 100° right.

A description of level n may, thus, be used to infer a variable for a description at level $n+1$. In the experiment just described, for instance, the description π-*fusion* is used to infer *Theta2* for description π-*phototaxy1*, and the result of this hierarchical composition is the description π-*phototaxy2*. In this experiment, all the information about *Theta2* is preserved. This information is passed as the distribution $\mathbf{P}(Theta2 \mid L1 \otimes \ldots \otimes L8 \otimes \pi\text{-}fusion)$ and all the possible values of *Theta2* are taken into account by way of the sum over this variable. However, as we

will show in 11, there is a possible alternative where a value for the variable at level n is first decided, then passed to the description at level $n+1$. In our example, we could have drawn a value for *Theta2* according to $\mathbf{P}(Theta2 \mid L1 \otimes \ldots \otimes L8 \otimes \pi\text{-}fusion)$ and then passed this value to $\mathbf{P}(Vrot \mid Theta2 \otimes \pi\text{-}phototaxy1)$. This second method is obviously much more computationally efficient than the first one (the sum over *Theta2* is no longer necessary), although the price to pay is that some information is lost in this second process.

9. Situation recognition

9.1 Goal and experimental protocol

The goal of this experiment is to distinguish different objects from one another.

At the beginning of the experiment the robot does not know any object. It must incrementally build categories for the objects it encounters. When it knows n of them, the robot must decide if a presented object enters in one of the n categories or if it is something new. If it is a new object, the robot must create a new category and should start to learn it.

9.2 Specification

Variables

The Khepera does not use its camera for this task. It must «grope» for the object. It uses the «contour following» behavior to do so (see Figure 4). It does a tour of the presented object and computes at the end of this tour four new variables: *Nlt* the number of left turns, *Nrt* the number of right turns, *Per* the perimeter and *Lrl* the longest straight line. The values of these variables are not completely determined by the shape of the object, given that the contour following behavior is quite choppy.

We also require a variable O to identify the different classes of object. The value $O = 0$ is reserved for the class of unknown (not yet presented) objects.

Finally, we obtain:

$$\begin{array}{c} Nlt \in \{0, \ldots, 24\}, \lfloor Nlt \rfloor = 25 \\ Nrt \in \{0, \ldots, 24\}, \lfloor Nrt \rfloor = 25 \\ Per \in \{0, \ldots, 9999\}, \lfloor Per \rfloor = 10000 \\ Lrl \in \{0, \ldots, 999\}, \lfloor Lrl \rfloor = 1000 \\ O \in \{0, \ldots, 15\}, \lfloor O \rfloor = 16 \end{array} \qquad [S9.1]$$

Decomposition

Obviously, the four variables *Nlt*, *Nrt*, *Per* and *Lrl* are not independent of one another. However, by reasoning similar to the sensor fusion case (see Section 7), we consider that knowing the object O, they are independent. Indeed, if the object is known, its perimeter or the number of turns necessary to complete a tour are also known. This leads to the following decomposition:

$$\begin{aligned}&\mathbf{P}(O \otimes Nlt \otimes Nrt \otimes Per \otimes Lrl \mid \Delta \otimes \pi\text{-}object)\\ &= \mathbf{P}(O \mid \pi\text{-}object) \times \mathbf{P}(Nlt \mid O \otimes \Delta \otimes \pi\text{-}object) \times \mathbf{P}(Nrt \mid O \otimes \Delta \otimes \pi\text{-}object)\\ &\times \mathbf{P}(Per \mid O \otimes \Delta \otimes \pi\text{-}object) \times \mathbf{P}(Lrl \mid O \otimes \Delta \otimes \pi\text{-}object)\end{aligned} \quad [S9.2]$$

Parametric forms

We have no *a priori* information on the presented object:

$$\mathbf{P}(O \mid \pi\text{-}object) \equiv Uniform \quad [S9.3]$$

For an observed object ($O \neq 0$), we state that the distributions on Nlt and Nrt are Laplace succession laws[18] and that the distributions on Per and Lrl are Gaussian laws:

$$\begin{aligned}&\forall o_i \in O, o_i \neq o_0\\ &\mathbf{P}(Nlt \mid o_i \otimes \Delta \otimes \pi\text{-}object) \equiv \mathbf{L}_1(n_{Nlt}(o_i))\\ &\mathbf{P}(Nrt \mid o_i \otimes \Delta \otimes \pi\text{-}object) \equiv \mathbf{L}_2(n_{Nrt}(o_i))\\ &\mathbf{P}(Per \mid o_i \otimes \Delta \otimes \pi\text{-}object) \equiv \mathbf{G}_1(\mu(o_i), \sigma(o_i))\\ &\mathbf{P}(Lrl \mid o_i \otimes \Delta \otimes \pi\text{-}object) \equiv \mathbf{G}_2(\mu(o_i), \sigma(o_i))\end{aligned} \quad [S9.4]$$

Finally, we state that for a new object ($O = 0$) we have no *a priori* information about Nlt, Nrt, Per and Lrl:

$$\begin{aligned}&\mathbf{P}(Nlt \mid o_0 \otimes \pi\text{-}object) \equiv Uniform\\ &\mathbf{P}(Nrt \mid o_0 \otimes \pi\text{-}object) \equiv Uniform\\ &\mathbf{P}(Per \mid o_0 \otimes \pi\text{-}object) \equiv Uniform\\ &\mathbf{P}(Lrl \mid o_0 \otimes \pi\text{-}object) \equiv Uniform\end{aligned} \quad [S9.5]$$

The preliminary knowledge composed of specifications [S9.1], [S9.2], [S9.3], [S9.4] and [S9.5] is named $\pi\text{-}object$.

9.3 Identification

When an object is presented to the robot, if it is recognized as a member of a class o_i, the parameters of the two Laplace succession laws and the two Gaussian laws corresponding to this class are updated.

If the object is considered by Khepera to be a new one, then a new class is created and the parameters of the distributions are initialized with the values of Nlt, Nrt, Per and Lrl just read.

The learning process is incremental. Contrary to what we have seen up to this point, the identification and utilization phases are not separated. Each new experience changes the set of data Δ, and leads to a new description $\mathbf{P}(O \otimes Nlt \otimes Nrt \otimes Per \otimes Lrl \mid \delta_n \otimes \pi\text{-}object)$.

---

18. A Laplace succession law on a variable V is defined by: $\frac{1 + n_v}{N + \lfloor V \rfloor}$ with N the total number of observations, $\lfloor V \rfloor$ the number of possible values for V and n_v the number of observations of the specific value v.

9.4 Utilization

After $n-1$ experiences, to recognize a presented object, the question to answer is:

$$\mathbf{P}(O \mid nlt_n \otimes nrt_n \otimes per_n \otimes lrl_n \otimes \delta_{n-1} \otimes \pi\text{-}object) \quad [E9.1]$$

This may be simply computed by:

$$\begin{aligned}&\mathbf{P}(O \mid Nlt \otimes Nrt \otimes Per \otimes Lrl \otimes \delta_{n-1} \otimes \pi\text{-}object)\\ &= \frac{1}{\Sigma} \times \mathbf{P}(Nlt \mid O \otimes \delta_{n-1} \otimes \pi\text{-}object) \times \mathbf{P}(Nrt \mid O \otimes \delta_{n-1} \otimes \pi\text{-}object)\\ &\times \mathbf{P}(Per \mid O \otimes \delta_{n-1} \otimes \pi\text{-}object) \times \mathbf{P}(Lrl \mid O \otimes \delta_{n-1} \otimes \pi\text{-}object)\end{aligned} \quad [E9.2]$$

If the most probable value for O is zero, then Khepera assumes that it is facing a new object. Otherwise, this most probable value is considered to correspond to the recognized object.

9.5 Results, lessons and comments

Results

The objects shown on Figure 14 have been presented to the robot, five times each, in random order. Each time the question was as follows: «Do you know this object, or is it a new one?» The robot did not ever fail to recognize novelty. At the end of the experiment, it was able to classify all the objects except for the two in the upper right corners. These two objects have the exact same square basis and thus may not be distinguished from one another given the four chosen variables. In these cases, Khepera was in the position of someone asked to identify the color of an object by groping it.

Figure 14: The different objects presented to Khepera.

Lessons

The main lesson to retain from this experiment is that categorization of objects or situations may be considered as developing some specific sensor. Indeed, the method used in this section for object recognition is very similar to what was achieved for sensor fusion in Section 7. The hypotheses are similar and the advantages are the same.

10. Temporal sequences

10.1 Goal and experimental protocol

In this paper, to exemplify the Bayesian programming method, we choose a «nightwatchman task». This may be obtained as temporal sequences of six simpler behaviors:

1 *idle*: The robot is at its base, recharging its batteries. It waits for both an order and enough energy to leave.
2 *patrol*: It wanders around its environment and sounds an alarm if it detects any movement.
3 *recognition*: The robot tours object to identify them.
4 *fire-intervention*: Khepera tries to extinguish fires by blowing them using its micro-turbine.
5 *homing*: It goes to its base when ordered to do so.
6 *recharge*: When low on energy, it goes to its base to recharge.

The purpose of this section is to show how such temporal sequences may be specified in the Bayesian framework.

10.2 Specification

Variables

The first variable to consider is $Behavior$,which may takes the six preceding values *idle*, *patrol*, *recognition*, *fire-intervention*, *homing* and *recharge*. This variable will be used to select a given behavior.

This selection will be made according to the values of the six following variables:

- $Vigil$: a binary variable, used to order the khepera to work.
- $Energy$: a variable that measures the level of available energy. $Energy$ may take four different values : *very-high*, *high*, *low* and *very-low*.
- $Base$: a binary variable, true if the robot is at its base.
- $Fire$: a binary variable, true if the robot detects any fire.
- $Identify$: a binary variable, used to order the Khepera to recognize an object.
- Finally, $Behavior_t\text{-}1$ a variable taking the same six values as $Behavior$, used to memorize which behavior was selected at time $t-1$.

This may be summed up as usual :

$$\begin{gathered} Behavior \in \{idle, patrol, recognition, fire-intervention, homing, recharge\}, \lfloor Behavior \rfloor = 6 \\ Vigil \in \{true, false\}, \lfloor Vigil \rfloor = 2 \\ Energy \in \{very-high, high, low, very-low\}, \lfloor Energy \rfloor = 4 \\ Base \in \{true, false\}, \lfloor Base \rfloor = 2 \\ Fire \in \{true, false\}, \lfloor Fire \rfloor = 2 \\ Identify \in \{true, false\}, \lfloor Identify \rfloor = 2 \\ Behavior_t\text{-}1 \in \{idle, patrol, \ldots, recharge\}, \lfloor Behavior_t\text{-}1 \rfloor = 6 \end{gathered} \quad [S10.1]$$

Decomposition

At each time step the robot will select a behavior knowing the values of these six variables by answering the question:

$$\mathbf{P}(Behavior \mid Vigil \otimes Energy \otimes Base \otimes Fire \otimes Identify \otimes Behavior_t\text{-}1 \otimes \pi\text{-}behavior) \quad [E10.1]$$

It is tempting to specify this distribution directly. It would correspond to the usual programming method where the conditions at time $t-1$ establish what should be done at time t.

We propose to do the exact opposite. Indeed, it is quite easy, knowing the behavior, to have some notion of the possible values of the variables *Vigil*, *Energy*, *Base*, *Fire*, and *Identify*. For instance, if the Khepera is patrolling, it means that it has been necessarily ordered to do so and that *Vigil* is *true*. Furthermore, we consider that knowing the behavior, these five variables are independent. These assumptions lead to the following decomposition:

$$\begin{aligned} &\mathbf{P}(Behavior \otimes Vigil \otimes Energy \otimes Base \otimes Fire \otimes Identify \otimes Behavior_t\text{-}1 \mid \pi\text{-}behavior) \\ &= \mathbf{P}(Behavior_t\text{-}1 \mid \pi\text{-}behavior) \times \mathbf{P}(Behavior \mid Behavior_t\text{-}1 \otimes \pi\text{-}behavior) \\ &\times \mathbf{P}(Vigil \mid Behavior \otimes \pi\text{-}behavior) \times \mathbf{P}(Energy \mid Behavior \otimes \pi\text{-}behavior) \\ &\times \mathbf{P}(Base \mid Behavior \otimes \pi\text{-}behavior) \times \mathbf{P}(Fire \mid Behavior \otimes \pi\text{-}behavior) \\ &\times \mathbf{P}(Identify \mid Behavior \otimes \pi\text{-}behavior) \end{aligned} \quad [S10.2]$$

Parametric forms

First we chose a uniform *a priori* value for $\mathbf{P}(Behavior_t\text{-}1 \mid \pi\text{-}behavior)$:

$$\mathbf{P}(Behavior_t\text{-}1 \mid \pi\text{-}behavior) \equiv Uniform \quad [S10.3]$$

We chose to specify all the other terms of this decomposition as discrete distributions. Their different values will be given *a priori*, one by one, using tables.

For instance, $\mathbf{P}(Behavior \mid Behavior_t\text{-}1 \otimes \pi\text{-}behavior)$ is specified by table 1.

This table should be read by column. Each column corresponds to the probability of *Behavior* knowing a certain behavior of the robot at time $t-1$. Consequently, each column should sum to 1 to respect the normalization constraint.

For instance, the first column of table 2 specifies the probabilities of variable *Behavior* knowing that the behavior of robot at time $t-1$ was *idle*. If Khepera was *idle,* then it may stay *idle* with a high probability (90%), it may not directly change its behavior to either *recognition*, *homing* or *recharge* (probability 0), it may switch to *patrol* or *fire-intervention* with a low probability (0.05 for both case obtained by normalization as specified by the «x»).

If the Khepera was in mode *patrol* (second column), the most probable behavior is that it stays in this mode, although it can switch to any other one. If the Khepera was in mode *recognition* (third column) we fix a very high probability for it to stay in this mode because we

do not want it to be easily distracted from this task and we preclude any possibility of switching to *idle*. In mode *fire-intervention* (column 4) we exclude any switch to *idle*, *recognition* or *homing*. Finally, when in mode *homing* or *recharge*, the most probable behavior is to not change mode, although nothing is definitely excluded.

Behavior / Behavior_t-1	*idle*	*patrol*	*recognition*	*fire-interv.*	*homing*	*recharge*
idle	*0.9*	*x*	*0*	*0*	*x*	*x*
patrol	*x*	*0.9*	*x*	*x*	*x*	*x*
recognition	*0*	*x*	*0.99*	*0*	*x*	*x*
fire-interv.	*x*	*x*	*x*	*x*	*x*	*x*
homing	*0*	*x*	*x*	*0*	*0.9*	*x*
recharge	*0*	*x*	*x*	*x*	*x*	*0.9*

Table 1: $\mathbf{P}(Behavior \mid Behavior_t\text{-}1 \otimes \pi\text{-}behavior)$

Table 2 mainly says that *patrol* and *recognition* suppose that *Vigil* is true and that *homing* supposes that *Vigil* is false. When *idle* the probability that *Vigil* is true is not 0, because the Khepera may be *idle* to recharge its batteries even when ordered to work.

Vigil / Behavior	*idle*	*patrol*	*recognition*	*fire-interv.*	*homing*	*recharge*
false	0.9	0	0	x	1	x
true	0.1	1	1	x	0	x

Table 2: $\mathbf{P}(Vigil \mid Behavior \otimes \pi\text{-}behavior)$

Table 3 specifies that when *idle* it is more probable that *Energy* is low than high. It also says that *patrol* and *recognition* suppose a high *Energy* and *recharge* the opposite.

Energy / Behavior	*idle*	*patrol*	*recognition*	*fire-interv.*	*homing*	*recharge*
very-low	0.325	0	0	x	x	0.8
low	0.325	0.1	0.1	x	x	0.2
high	0.25	x	x	x	x	0
very-high	0.1	x	x	x	x	0

Table 3: $\mathbf{P}(Energy \mid Behavior \otimes \pi\text{-}behavior)$

Table 4 says that *idle* imposes that *Base* is true, when *patrol*, *recognition*, *homing* and *recharge* suppose with a high probability that Khepera is not at its base.

Base/ Behavior	*idle*	*patrol*	*recognition*	*fire-interv.*	*homing*	*recharge*

Table 4: $\mathbf{P}(Base \mid Behavior \otimes \pi\text{-}behavior)$

false	0	0.99	0.99	x	0.99	0.99
true	1	0.01	0.01	x	0.01	0.01

Table 4: $\mathbf{P}(Base \mid Behavior \otimes \pi\text{-}behavior)$

Table 5 means that when Khepera is facing a fire, it is necessarily in mode *fire-intervention*.

Fire / Behavior	*idle*	*patrol*	*recognition*	*fire-interv.*	*homing*	*recharge*
false	1	1	1	0	1	1
true	0	0	0	1	0	0

Table 5: $\mathbf{P}(Fire \mid Behavior \otimes \pi\text{-}behavior)$

Finally, Table 6 says *recognition* imposes that Khepera has been ordered to do so (*Identify* is true).

Identify / Behavior	*idle*	*patrol*	*recognition*	*fire-interv.*	*homing*	*recharge*
false	x	x	0	x	x	x
true	x	x	1	x	x	x

Table 6: $\mathbf{P}(Identify \mid Behavior \otimes \pi\text{-}behavior)$

10.3 Identification

No identification is required, as there are no free parameters in π-*behavior*

10.4 Utilization

The robot chooses its behavior with the following query:

$$\mathbf{Draw}(\mathbf{P}(Behavior \mid Vigil \otimes Energy \otimes Base \otimes Fire \otimes Identify \otimes Behavior_t\text{-}1 \otimes \pi\text{-}behavior)) \quad \text{[E10.1]}$$

that can be easily computed:

$$\begin{aligned}
&\mathbf{P}(Behavior \mid Vigil \otimes Energy \otimes Base \otimes Fire \otimes Identify \otimes Behavior_t\text{-}1 \otimes \pi\text{-}behavior)\\
&= \frac{1}{\Sigma} \times \mathbf{P}(Behavior_t\text{-}1 \mid \pi\text{-}behavior) \times \mathbf{P}(Behavior \mid Behavior_t\text{-}1 \otimes \pi\text{-}behavior)\\
&\times \mathbf{P}(Vigil \mid Behavior \otimes \pi\text{-}behavior) \times \mathbf{P}(Energy \mid Behavior \otimes \pi\text{-}behavior)\\
&\times \mathbf{P}(Base \mid Behavior \otimes \pi\text{-}behavior) \times \mathbf{P}(Fire \mid Behavior \otimes \pi\text{-}behavior) \times \mathbf{P}(Identify \mid Behavior \otimes \pi\text{-}behavior)
\end{aligned} \quad \text{[E10.2]}$$

10.5 Results, lessons and comments

Results

Using these techniques, Khepera obtains temporal sequences of behaviors that appear convincing to a human observer (an instance of such a sequence will be given in the next section, see Movie 5[19]).

For instance, these sequences are stable. Khepera does not behave like a weathercock

that changes its mind every second.

Inverse programming

This experiment demonstrates a completely new method of specifying temporal sequences of tasks that could be called «inverse temporal programming». Indeed, the programmer does not specify, as usual, the necessary conditions for an action. On the contrary, he or she specifies for each action the expected observations and assumes that knowing the action these observations are independent.

Inverse programming presents two main advantages.

- It is robust to unforeseen situations. A sequence of actions is always produced, even in cases that the programmer did not explicitly take into account.
- Due to the conditional independence assumption, the number of cases to take into account grows only linearly with the number of conditioning variables.

The a priori specification of the probability distributions of the observed variables knowing the behavior may be a difficulty. However its is possible to learn these distributions (see Diard & Lebeltel, 1999).

11. Integration: A Nightwatchman Task

11.1 Goal and experimental protocol

The practical goal and experimental protocol of the night watchman task has already been presented in Section 10.1.

The scientific purpose of this last experiment is to prove that Bayesian robots programming is an efficient constructive methodology and that all the previous descriptions may be integrated into a single synthetic one.

Three descriptions and a few corresponding variables necessary for the night watchman task have not yet been presented to keep the paper short:

1 - $\mathbf{P}(Base \otimes Px1 \otimes \ldots \otimes Px8 \otimes L1 \otimes \ldots \otimes L8 | \pi\text{-}base)$ used by Khepera to decide if it is at its base

2 - $\mathbf{P}(Move \otimes Behavior \otimes Move_t\text{-}1 \otimes Tempo \otimes Tour \mid \pi\text{-}move)$ another temporal sequencing description required because some of the behaviors are successions of reactive movements.

3 - $\mathbf{P}(Vrot \otimes Vtrans \otimes Move \otimes H \otimes Dir \otimes Prox \otimes DirL \otimes ProxL \otimes Vtrans_c \otimes Theta2 \mid \pi\text{-}speed)$ built on the reactive behaviors to finally decide the rotation and translation speeds.

11.2 Specification

Variables

The nightwatchman task requires 41 variables:

- Thirty-three «sensory» variables that Khepera may read every tenth of a second. When convenient, we will summarize these 33 variables by their conjunction (a variable named *Sensory-variables*).

19. http://www-leibniz.imag.fr/LAPLACE/Cours/Semaine-Science/Trans12/T12.small.mov (QuickTime,5.5Mo)

$$\begin{aligned} \textit{Sensory-variables} \equiv\ & \textit{Px1} \otimes \ldots \otimes \textit{Px8} \otimes \textit{L1} \otimes \ldots \otimes \textit{L8} \\ & \otimes \textit{Vigil} \otimes \textit{Energy} \otimes \textit{Fire} \otimes \textit{Identify} \otimes \textit{Behavior_t-1} \\ & \otimes \textit{Move_t-1} \otimes \textit{Tempo} \otimes \textit{Tour} \otimes \textit{Dir} \otimes \textit{Prox} \otimes \textit{DirL} \otimes \textit{ProxL} \otimes \textit{Vtrans_c} \\ & \otimes \textit{Nlt} \otimes \textit{Nrt} \otimes \textit{Per} \otimes \textit{Lrl} \end{aligned} \qquad \text{[E11.1]}$$

- Five internal variables: $\textit{Base}$, $\textit{Theta2}$, $\textit{Behavior}$, $\textit{Move}$, H
- Three «motor» variables that Khepera must compute. These three variables are the rotation speed $\textit{Vrot}$, the translation speed $\textit{Vtrans}$ and the identity of the object O.

Decomposition and parametric forms

The decomposition of the joint distribution on these 41 variables is a product of a uniform distribution on the sensory variables ($\mathbf{P}(\textit{Sensory-variables}|\pi\textit{-watchman})$) and eight questions addressed to the previously defined descriptions:

$$\begin{aligned} & \mathbf{P}\left(\begin{array}{c} \textit{Sensory-variables} \\ \textit{Base} \otimes \textit{Theta2} \otimes \textit{Behavior} \otimes \textit{Move} \otimes H \\ \textit{Vrot} \otimes \textit{Vtrans} \otimes O \end{array} \middle| \pi\textit{-watchman}\right) \\ & = \mathbf{P}(\textit{Sensory-variables}|\pi\textit{-watchman}) \\ & \times \mathbf{P}(\textit{Base}|\textit{Px1} \otimes \ldots \otimes \textit{Px8} \otimes \textit{L1} \otimes \ldots \otimes \textit{L8} \otimes \pi\textit{-base}) \\ & \times \mathbf{P}(\textit{Theta2} \mid \textit{L1} \otimes \textit{L2} \otimes \textit{L3} \otimes \textit{L4} \otimes \textit{L5} \otimes \textit{L6} \otimes \textit{L7} \otimes \textit{L8} \otimes \pi\textit{-fusion}) \\ & \times \mathbf{P}(\textit{Behavior} \mid \textit{Vigil} \otimes \textit{Energy} \otimes \textit{Base} \otimes \textit{Fire} \otimes \textit{Identify} \otimes \textit{Behavior_t-1} \otimes \pi\textit{-behavior}) \\ & \times \mathbf{P}(\textit{Move} \mid \textit{Behavior} \otimes \textit{Move_t-1} \otimes \textit{Tempo} \otimes \textit{Tour} \otimes \pi\textit{-move}) \\ & \times \mathbf{P}(H \mid \textit{Prox} \otimes \pi\textit{-home}) \\ & \times \mathbf{P}(\textit{Vrot} \otimes \textit{Vtrans} \mid \textit{Move} \otimes H \otimes \textit{Dir} \otimes \textit{Prox} \otimes \textit{DirL} \otimes \textit{ProxL} \otimes \textit{Vtrans_c} \otimes \textit{Theta2} \otimes \pi\textit{-speed}) \\ & \times \mathbf{P}(O \mid \textit{Nlt} \otimes \textit{Nrt} \otimes \textit{Per} \otimes \textit{Lrl} \otimes \pi\textit{-object}) \end{aligned} \qquad \text{[E11.2]}$$

11.3 Identification

No identification is required.

11.4 Utilization

The ultimate question that Khepera must answer is:

$$\mathbf{P}(\textit{Vrot} \otimes \textit{Vtrans} \otimes O \mid \textit{Sensory-variables} \otimes \pi\textit{-watchman}) \qquad \text{[E11.3]}$$

«What order should be sent to the motors, knowing the sensory state, and ignoring the values of the internal variables?»

The answer to that question is obtained, as usual, by summing over the five ignored variables. This leads to the following result:

$$\mathbf{P}(Vrot \otimes Vtrans \otimes O \mid Sensory\text{-}variables \otimes \pi\text{-}watchman)$$
$$= \frac{1}{\Sigma} \times \mathbf{P}(O \mid Nlt \otimes Nrt \otimes Per \otimes Lrl \otimes \pi\text{-}object)$$
$$\times \sum_{\substack{Move \\ Theta2 \\ H}} \left(\begin{array}{l} \sum_{Behavior} \left(\begin{array}{l} \mathbf{P}(Move \mid Behavior \otimes \ldots \otimes Tour \otimes \pi\text{-}move) \times \\ \sum_{Base} \left(\begin{array}{l} \mathbf{P}(Behavior \mid Vigil \otimes \ldots \otimes Behavior_t\text{-}1 \otimes \pi\text{-}behavior) \\ \times \mathbf{P}(Base \mid Px1 \otimes \ldots\ldots \otimes L8 \otimes \pi\text{-}base) \end{array} \right) \end{array} \right) \\ \times \mathbf{P}(Theta2 \mid L1 \otimes \ldots \otimes L8 \otimes \pi\text{-}fusion) \\ \times \mathbf{P}(H \mid Prox \otimes \pi\text{-}home) \\ \times \mathbf{P}(Vrot \otimes Vtrans \mid Move \otimes \ldots \otimes Theta2 \otimes \pi\text{-}speed) \end{array} \right) \qquad \text{[E11.4]}$$

This expression may seem complex. In fact, it exactly reflects the structure of the reasoning required to solve the problem.

- Recognizing the object is independent of the Khepera control.
- The innermost sum searches the *Behavior* ignoring *Base*

$$\sum_{Base} \left(\begin{array}{l} \mathbf{P}(Behavior \mid Vigil \otimes \ldots \otimes Behavior_t\text{-}1 \otimes \pi\text{-}behavior) \\ \times \mathbf{P}(Base \mid Px1 \otimes \ldots \otimes L8 \otimes \pi\text{-}base) \end{array} \right) \qquad \text{[E11.5]}$$
$$= \mathbf{P}(Behavior \mid Vigil \otimes \ldots \otimes Behavior_t\text{-}1 \otimes Px1 \otimes \ldots \otimes L8 \otimes \pi\text{-}watchman)$$

- The intermediary sum searches the movement ignoring the *Behavior* and *Base*.
- The position of the light source (*Theta2*) is estimated by the fusion of the light sensors information.
- The command variable *H* is estimated according to the value of *Prox*.
- The outermost sum searches for *Vrot* and *Vtrans* ignoring the precise values of the five internal variables.

No decision is made except the ultimate one about *Vrot* and *Vtrans*. Uncertainty is propagated from the innermost level to the outermost. All the available information is taken into account. The resulting observed robot behavior is, indeed, a probabilistic mixture of the different component descriptions.

Discarding no information has an obvious computational cost. The evaluation of the three levels of cascading sums may be very time consuming. Thus, the programmer may choose to make decisions on any intermediary variables. This choice will always trade a gain of efficiency for a loss of information. For instance, the most efficient possible program would make a decision for all the internal variables:

1 - **Draw**($\mathbf{P}(Base \mid Px1 \otimes \ldots \otimes L8 \otimes \pi\text{-}base)$) to decide if the robot is at its base,

2 - **Draw**($\mathbf{P}(Behavior \mid \ldots \otimes Base \otimes \ldots \otimes \pi\text{-}behavior)$) to decide the *Behavior* knowing *Base*,

3 - **Draw**($\mathbf{P}(Move \mid Behavior \otimes \ldots \otimes \pi\text{-}move)$) to chose a movement knowing the *Behavior*,

4 - **Draw**($\mathbf{P}(Theta2 \mid L1 \otimes \ldots \otimes L8 \otimes \pi\text{-}fusion)$) to decide the position of the light source,

5 - **Draw**($\mathbf{P}(H \mid Prox \otimes \pi\text{-}home)$) to decide between avoidance and phototaxy,

6 - and finally, **Draw**($\mathbf{P}(Vrot \otimes Vtrans \mid \ldots \otimes \pi\text{-}speed)$) to control the robot.

11.5 Results, lessons and comments

The results obtained are satisfactory to a human observer. The Khepera performed this task hundreds of time in various environments and conditions. The behavior was very robust; for instance, this experiment ran without interruption, 10 hours a day for three days as a demonstration during a conference.

The Movie 5[19] shows the Khepera during one of these experiments. It successively shows:

- Khepera identifying an object,
- Khepera aborting its object recognition due to a possible fire detection,
- Khepera verifying that it is really facing a fire by trying to blow it,
- Khepera extinguishing the fire,
- Khepera patrolling the environment (it stops occasionally to detect movement and sounds an alarm if it succeeds),
- Khepera returning to its base.

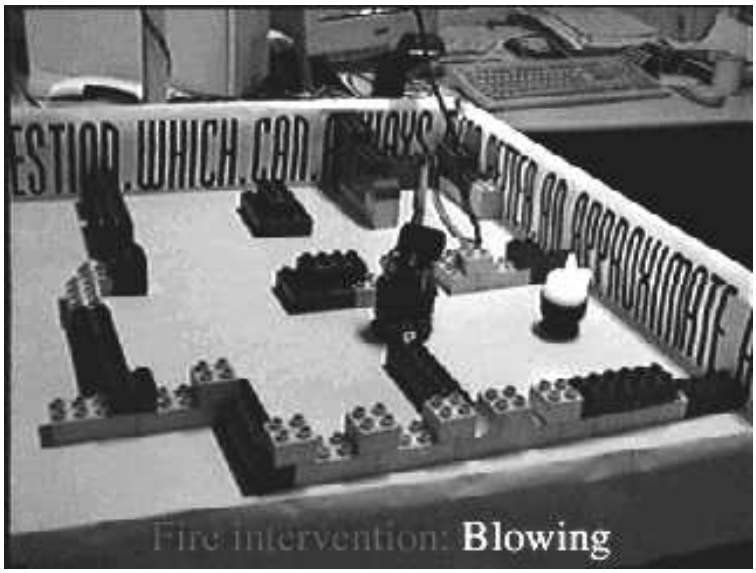


Figure 15: The night watchman task.

12. Synthesis

12.1 Principles, theoretical foundation and methodology

Principles

The dominant paradigm in robotics may be caricatured by Figure 16.

The programmer of the robot has an abstract conception of its environment. He or she may describe the environment in geometrical terms because the shape of objects and the map

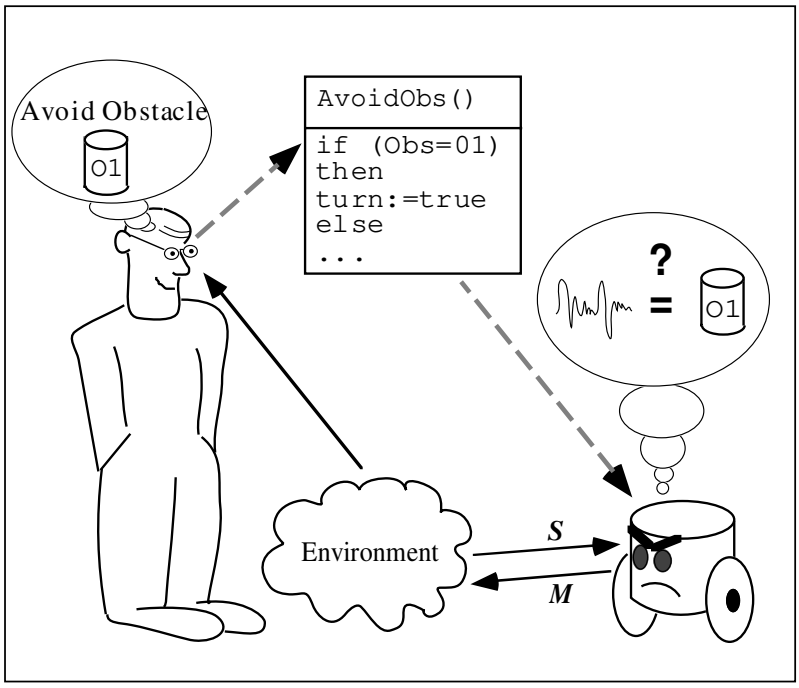


Figure 16: The symbolic approach in robotics.

of the world can be specified. He or she may be described the environment in analytical terms because laws of physics that govern this world are known. The environment may also be described in symbolic terms because both the objects and their characteristics can be named.

The programmer uses this abstract representation to program the robot. The programs use these geometric, analytic and symbolic notions. In a way, the programmer imposes on the robot his or her own conception of the environment.

The difficulties of this approach appear when the robot needs to link these abstract concepts with the raw signals it obtains from its sensors and sends to its actuators.

The central origin of these difficulties is the irreducible incompleteness of the models. Indeed, there are always some hidden variables, not taken into account in the model, that influence the phenomenon. The effect of these hidden variables is that the model and the phenomenon never behave exactly the same. The hidden variables prevent the robot from relating the abstract concepts and the raw sensory-motor data reliably. The sensory-motor data are then said to be «noisy» or even «aberrant». A queer reversal of causality occurs that seem to consider that the mathematical model is exact and that the physical world has some unknown flaws.

Compelling the environment is the usual answer to these difficulties. The programmer of the robot looks for the causes of «noises» and modifies either the robot or the environment to suppress these «flaws». The environment is modified until it corresponds to its mathematical model. This approach is both legitimate and efficient from an engineering point of view. A

precise control of both the environment and the tasks ensures that industrial robots work properly.

However, compelling the environment may not be possible when the robot must act in an environment not specifically designed for it. In that case, completely different solutions must be devised.

The purpose of this paper is to propose Bayesian robot programming as a possible solution.

Figure 17 presents the principles of this approach.

The fundamental notion is to place side by side the programmer's conception of the task

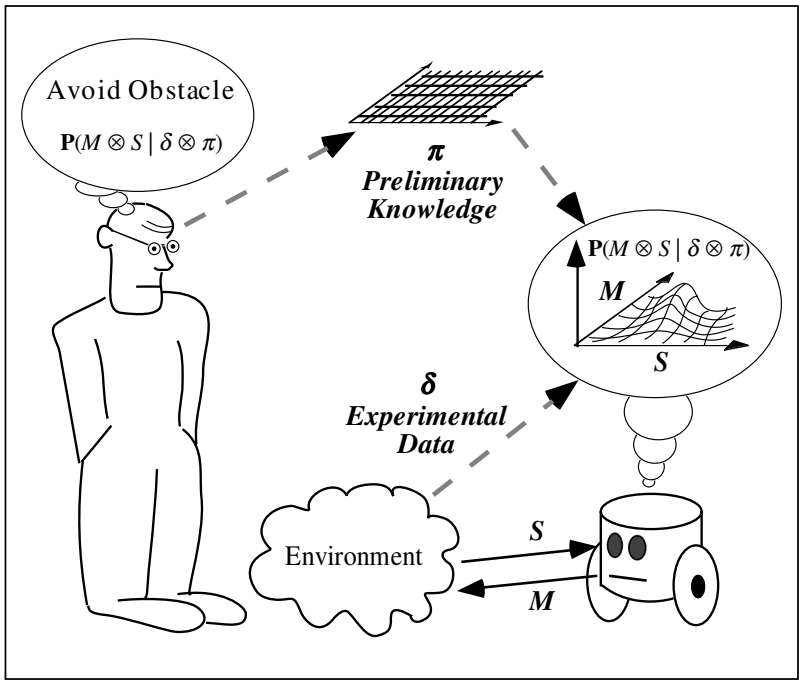


Figure 17: The Bayesian programming approach in robotics.

(the preliminary knowledge) and the experimental data to obtain the programming resources called «descriptions». As seen in the different examples described in this paper, both the preliminary knowledge and the descriptions may be expressed easily and efficiently in probabilistic terms.

The preliminary knowledge gives some hints to the robot about what it may expect to observe. The preliminary knowledge is not a fixed and rigid model purporting completeness. Rather, it is a gauge, with open parameters, waiting to be molded by the experimental data. Learning is the means of setting these parameters. The resulting descriptions result from both the views of the programmer and the physical specificities of each robot and environment. Even the influence of the hidden variables is taken into account and quantified; the

more important their effects, the more noisy the data, the more uncertain the resulting descriptions.

However, Bayesian robot programming preserves two very important merits of the symbolic approach. Thanks to the preliminary knowledge, the descriptions are comprehensible to the programmer. Thanks to Bayesian inference, complex reasoning is possible.

Theoretical foundations

The theoretical foundations of Bayesian robot programming may be summed up by Figure 18.

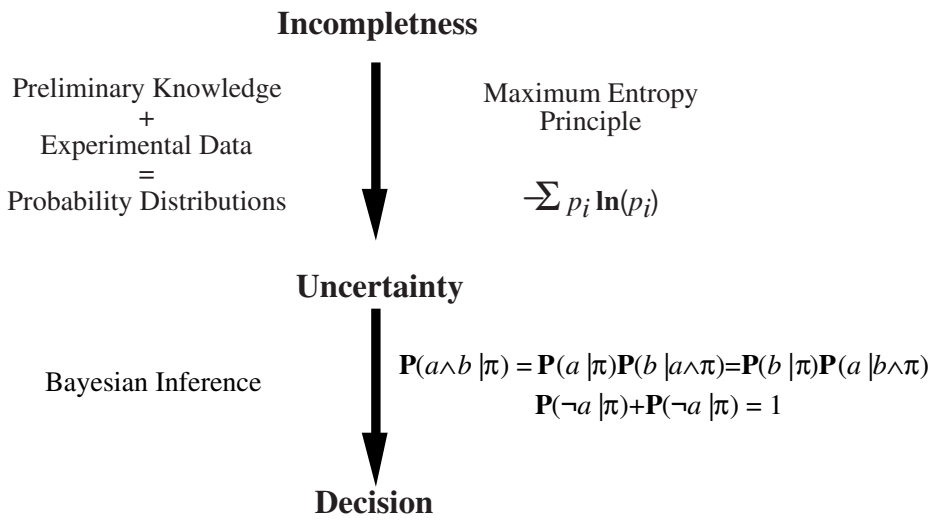


Figure 18: Theoretical foundation.

The first step transforms the irreducible incompleteness to uncertainty. Starting from the preliminary knowledge and the experimental data, learning builds probability distributions.

The maximum entropy principle is the theoretical foundation of this first step. Given some preliminary knowledge and some data, the probability distribution that maximizes the entropy is *the* distribution that *best* represents this couple. Entropy gives a precise, mathematical and quantifiable meaning to the «quality» of a distribution (for justifications of the maximum entropy principle see, for instance, Jaynes, 1982; Robert, 1990; Bessière et al., 1998b).

Two extreme examples may help to understand what occurs:

- Suppose that we are studying a formal phenomenon. There are no hidden variables. A complete model may be proposed. If we select this model as the preliminary knowledge, any data set will lead to a description made of Diracs. There is no uncertainty, any question may be answered either by true or false. Logic appears as a special case of the Bayesian approach in that particular context (see Cox, 1979).
- On the opposite extreme, suppose that preliminary knowledge consists of very poor hypotheses about the modeled phenomenon. Learning will lead to «flat» distribu-

tions, containing no information. No relevant decisions can be made, only completely random ones.

Hopefully, most common cases, are somewhere in between these two extremes. Preliminary knowledge, even imperfect and incomplete is relevant and provides interesting hints about the observed phenomenon. The resulting descriptions are neither Diracs nor uniform distributions. They give no certitudes, although they provide a means of taking the best possible decision according to the available information.

The second step consists of reasoning with the probability distributions obtained by the first step.

To do so, we only require the two basic rules of Bayesian inference (see Section 3). These two rules are to Bayesian inference what the resolution principle is to logical reasoning (see Robinson, 1965; Robinson, 1979; Robinson & Sibert, 1983a; Robinson & Sibert, 1983b). These inferences may be as complex and subtle as those usually achieved with logical inference tools, as demonstrated in the different examples in this paper.

Methodology

The proposed robot programming method results directly from this theoretical foundations. Let us recall it for the last time:

1 – *Specification*: define the preliminary knowledge

  1.1 – Choose the pertinent variables

  1.2 – Decompose the joint distribution

  1.3 – Define the Parametric forms

2 – *Identification* : identify the free parameters of the preliminary knowledge

3 – *Utilization*: ask a question of the joint distribution

12.2 Advantages

In this section we survey, comment and briefly discuss the advantages of the Bayesian robot programming method proposed in this paper.

- *Ability to treat incomplete and uncertain information*: The basis of this work is related to the fundamental difficulty of robot programming in real environment. For us this difficulty is the direct consequence of the irreducible incompleteness of models. Consequently, the first advantage of the proposed approach is its ability to take into account this incompleteness and the resulting uncertainty. This is obtained in three steps, thanks to the following three abilities of the method:
  - *Ability to convert incompleteness to uncertainty by learning*, as demonstrated in the numerous instances where the free parameters of preliminary knowledge are identified from experimental data (see, for instance, Section 5 concerning reactive behaviors or Setcion 9 concerning object recognition). Object recognition, for instance, shows that with simple preliminary knowledge, we are able to learn descriptions sufficient for classification. However, in this task there are numerous ignored variables such as, for instance, the color and material of the objects, the

global lighting of the room, the approximate quality of the contour following behavior or the position from where the robot has started.

- ° *Ability to reason despite uncertainty*, as demonstrated by all the experiments requiring inference (see, for instance, Section 7 about sensor fusion or Section 9 about object recognition). The «nightwatchman» task (see Section 11) shows the complexity of the possible reasoning (41 variables, 12 descriptions, four hierarchical levels).
- ° *Ability to decide, taking uncertainty into account*: The decision strategy selected in this work has been to draw the values of the searched variables from the distributions obtained by the preceding inference step. This strategy «renders» uncertainty, the decision are nearly deterministic when the distributions are sharp, and conversely, nearly random when they are flat.

- *Simple and sound theoretical bases*: The proposed approach is founded on simple theoretical bases. Essential questions may be asked clearly and formally and eventually answered by mathematical reasoning. For instance, one may consider to fairly compare Bayesian inference and logic as two possible models of reasoning. Thanks to these theoretical bases, the experimental results (successes or even more enlightening failures) may be analyzed and understood in detail.
- *Generic, systematic and simple programming method*: The proposed programming method is simple, systematic and generic. Simple, as this method may be learned and mastered easily. Systematic, as it may be applied with rigor and efficiency. Generic, as this method may be also used in numerous other domains than robot programming, for instance CAD (see Mekhnacha, 1999).
- *Homogeneity of representations and resolution processes*: This method is based on a unique data structure, called a description, associated with two inference rules. This homogeneity leads to simple and generic program development.
- *Obligation to state all hypothesis*: Choosing a description as the only data structure to specify robotics programs and following a systematic method to do so compel the programmer to exhaustively express his knowledge about the task. Everything that should be known about a given robotics problem is in its 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.
- *Large capacity of expression*: Descriptions offer a large capacity of expression to specify models and to question them as well.
  - ° *Specification capacity*: The different experiments described in this paper prove that descriptions may be used to specify numerous different models. Let us recall that we used descriptions to learn simple reactive behaviors (Section 5), to combine them (Section 6), to hierarchically compose them (Section 8), to merge sensor information (Section 7), to recognize situations (Section 9), to carry out temporal sequencing (Section 10) and finally, to specify a task integrating all the previously defined descriptions (Section 11).
  - ° *Question capacity*: Let us also recall that any question may be asked to a joint distribution. Mathematically, all variables appearing in a joint distribution play the exact same role. They may all, indifferently, be known, unknown or searched. The description is neither a direct nor an inverse model. Sensor fusion (Section 7), situation recognition (Section 9) or inverse programming (Section 10) offer

instances where the questions asked do not correspond to the specification order. Furthermore, there is no ill-posed problem. If a question may have several solutions, the probabilistic answer will simply have several peaks. Some instances of sensor fusion exemplified this point (see Section 7.3).

- *Ability for real incremental development of robots*: Bayesian robot programming, thanks to its clear theoretical foundations and to its rigorous programming methodology, appears to be an incremental method of developing robot programs that could really be used in practice. The final experiment (Section 11) demonstrates that point.
  - *Ability to combine descriptions*: The first incremental development tool is description combination (Section 6). With this tool it is possible to define new behaviors as weighted mixtures of different simpler ones.
  - *Ability to compose descriptions*: The second incremental development tool is hierarchical description composition (Section 8). It is in some senses 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.
  - *Description = Resource*: More generally, a description, as an internal representation of a physical phenomenon, may be considered as a programming ressource. For instance, a description may offer new variables to be used in other descriptions. This is the case with the variable O that identifies the object, provided by the object recognition description (Section 9). Object recognition also proposes another example of the use of a description as a programming resource. Indeed, the countour following behavior is a necessary tool to be ablefor computing the four variables Nlt, Nrt, Per and Lrl used by the object recognition description. Numerous other possibilities for enhancing the capacity of a robot using descriptions as resources may be found in Dedieu's Ph.D. thesis (Dedieu, 1995).

12.3 Conclusion

We have introduced a new formalism to program robots. Our approach closely implements the Bayesian inference paradigm and, as a result, follows a clear mathematical framework. It permits programming of robots while explicitly taking into account the incompleteness of the models chosen by the programmer. The proposed system has been used to program several tasks. We have demonstrated that complex programs may be obtained by combining simpler components. Experimental tests have shown the effectiveness and the robustness of the programs built. Many developments are considered. At the theoretical level, we are testing new methods to automatically infer the decomposition from a set of examples. We are also improving the resolution method used in our inference engine. At the application level the system will be used to fuse sensors to control an automated car (European Project Car Sense). It will also be used to program a new version of the Khepera robot : the Koala. This robot is equipped with several new sensors (color ccd, compass, directional microphones) and can run in a larger environment. To extend the range of application we plan to use the Bayesian programming scheme to control artificial agents in virtual worlds. We believe this approach may ultimately lead to a new generation of robot programming languages.

References

Alami, R., Chatila, R., Fleury, S., Ghallab, M. & Ingrand, F. ; (1998) ; An Architecture for Autonomy ; *International Journal for Robotics Research* (IJRR) ; Vol. 17(4), pp. 315-337

Aycard, O.; (1998) ; *Architecture de contrôle pour robot mobile en environnement intérieur structuré* ; Ph.D. thesis, Univeristé Henri Poincaré, Nancy, France

Bernhardt, R. & Albright, S.L. (editors) ; (1993) ; *Robot Calibration* ; Chapman & Hall

Bessière, P., Dedieu, E., Lebeltel, O., Mazer, E. & Mekhnacha, K. ; (1998a) ; Interprétation ou Description (I) : Proposition pour une théorie probabiliste des systèmes cognitifs sensori-moteurs ; *Intellectica* ; Vol. 26-27, pp. 257-311; Paris, France

Bessière, P., Dedieu, E., Lebeltel, O., Mazer, E. & Mekhnacha, K. ; (1998a) ; Interprétation ou Description (I) : Fondements mathématiques de l'approche F+D ; *Intellectica* ; Vol. 26-27, pp. 313-336 ; Paris, France

Borrelly, J-J., Coste, E., Espiau, B., Kapellos, K., Pissard-Gibollet, R., Simon, D. & Turro, N.; (1998) ; The ORCCAD Architecture ; *International Journal for Robotics Research* (IJRR) ; Vol. 17(4), pp. 338-359

Brafman, R.I., Latombe, J-C., Moses, Y. & Shoham, Y.; (1997) ; Applications of a logic of knowledge to motion planning under uncertainty ; *Journal of the ACM*, vol.44(5), pp. 633-68

Bretthorst, G.L. ; (1988) ; *Bayesian spectrum analysis and parameter estimation* ; Spinger Verlag

Brooks, R.A. ; (1986) ; A robust layered control systems for a mobile robot ; IEEE Journal of Robotics and Automation ; Vol. 2(1), pp. 14-23

Cooper, G. ; (1990) ; The computational complexity of probabilistic inference using Bayesian belief networks ; *Artificial Intelligence*, Vol. 42, pp. 393-405

Cox, R.T. ; (1961) ; *The algebra of probable inference* ; The John Hopkins Press, Baltimore, USA

Cox, R.T. ; (1979) ; Of inference and inquiry, an essay in inductive logic ; in *The maximum entropy formalism*, edited by Raphael D. Levine & Myron Tribus ; M.I.T. Press, U.S.A.

Dagum, P. & Luby, M. ; (1993) ; Approximate probabilistic reasoning in Bayesian belief network is NP-Hard ; *Artificial Intelligence*, Vol. 60, pp. 141-153

Darwiche, A. and Provan, G. ; (1997) ; Query DAGs: A Practical Paradigm for Implementing Belief-Network Inference ; *Journal of Artificial Intelligence Research* (JAIR), Vol. 6, pp. 147-176

Dedieu, E. ; (1995) ; *La représentation contingente : Vers une reconciliation des approches fonctionnelles et structurelles de la robotique autonome*. Thèse de troisième cycle INPG (Institut National Polytechnique de Grenoble) ; Grenoble, France

Dekhil, M. & Henderson, T.C. ; (1998) ; Instrumented Sensor System Architecture ; *International Journal for Robotics Research* (IJRR) ; Vol. 17(4), pp. 402-417

Delcher, A.L., Grove, A.J., Kasif, S. and Pearl, J. ; (1996) ; Logarithmic-Time Updates and Queries in Probabilistic Networks ; *Journal of Artificial Intelligence Research* (JAIR) ; Vol. 4, pp. 37-59

Diard, J. & Lebeltel, O. ; (1999) ; Bayesian Learning Experiments with a Khepera Robot in Experiments with the Mini-Robot Khepera : Proceedings of the 1st International Khepera Workshop, December 1999, Löffler Mondada Rückert (Editors), Paderborn, HNI-Verlagsschriftenreihe ; Band 64 ; Germany ; pp. 129-138 ;

Donald, B.R.; (1988) ; A geometric approach to error detection and recovery for robot motion planning with uncertainty ; *Artificial Intelligence*, vol.37, pp. 223-271

Erickson, G.J. & Smith, C.R. ; (1988a) ; *Maximum-Entropy and Bayesian methods in science and engineering ; Volume 1 : Foundations* ; Kluwer Academic Publishers

Erickson, G.J. & Smith, C.R. ; (1988b) ; *Maximum-Entropy and Bayesian methods in science and engineering ; Volume 2 : Application*s ; Kluwer Academic Publishers

Frey, B.J. ; (1998) ; *Graphical Models for Machine Learning and Digital Communication* ; MIT Press

Halpern, J.Y. ; (1999a) ; A Counterexample to Theorems of Cox and Fine ; *Journal of Artificial Intelligence Research* (JAIR), Vol. 10, pp. 67-85.

Halpern, J.Y. ; (1999b) ; Cox's Theorem Revisited ; *Journal of Artificial Intelligence Research* (JAIR), Vol. 11, pp. 429-435.

Jaakkola, T.S. and Jordan, M.I. ; (1999) ; Variational Probabilistic Inference and the QMR-DT Network ; *Journal of Artificial Intelligence Research* (JAIR), Vol. 10, pp. 291-322

Jaynes, E.T. ; (1979) ; Where do we Stand on Maximum Entropy? ; in *The maximum entropy formalism* ; edited by Raphael D. Levine & Myron Tribus ; M.I.T. Press

Jaynes, E.T. ; (1982) ; On the rationale of maximum-entropy methods ; Proceedings of the IEEE

Jaynes, E.T. ; (1998) ; *Probability theory - The logic of science* ; unfinished book available at http:// bayes.wustl.edu/etj/prob.html

Jordan MI and Jacobs RA (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation* ; Vol. 6, pp. 181-214.

Jordan, M. ; (1998) ; *Learning in Graphical Models ;* MIT Press

Jordan, M., Ghahramani, Z., Jaakkola, T.S. & Saul, L.K. ; (1999) ; An introduction to variational methods for graphical models ; In press, *Machine Learning*

Kaelbling, L.P., Littman, M.L. & Cassandra, A.R.; (1996) ; Partially observable Markov decision processes for artificial intelligence ; Reasoning with Uncertainty in Robotics. International Workshop, RUR'95, Proceedings pp.146-62 ; Springer-Verlag

Kapur, J.N., & Kesavan, H.K. ; (1992) ; *Entropy optimization principles with applications* ; Academic Press

Laplace, Pierre Simon de (1774); *Mémoire sur la probabilités des causes par les évènements*; Mémoire de l'académie royale des sciences; Reprinted in *Oeuvres complètes de Laplace*, (vol. 8), Gauthier Villars, Paris, France

Laplace, Pierre Simon de (1814); *Essai philosphique sur les probabilités*; Courcier Imprimeur, Paris; Reprinted in *Oeuvres complètes de Laplace*, (vol. 7), Gauthier Villars, Paris, France

Lauritzen, S. & Spiegelhalter, D. ; (1988) ; Local computations with probabilities on graphical structures and their application to expert systems ; *Journal of the Royal Stastical Society B* ; Vol. 50, pp. 157-224

Lauritzen, S. L. ; (1996) ; *Graphical Models* ; Oxford University Press

Lebeltel, O. ; (1999) ; *Programmation Bayésienne des Robots* ; Ph.D. Thesis, Institut National Polytechnique de Grenoble (INPG); Grenoble, France

Lozano-Perez, T., Mason, M.T., Taylor, R.H.; (1984) ; Automatic synthesis of fine-motion strategies for robots ; *International Journal of Robotics Research*, vol.3(1), pp. 3-24

Maes, P. ; (1989) ; How to Do the Right Thing ; *Connection Science Journal* ; Vol. 1, N°3, pp. 291-323

Matalon, B. ; (1967) ; Epistémologie des probabilités ; in *Logique et connaissance scientifique* edited by Jean Piaget ; Encyclopédie de la Pléiade ; Editions Gallimard ; Paris, France

Mazer, E., Boismain, G., Bonnet des Tuves, J., Douillard, Y., Geoffroy, S., Dubourdieu, J., Tounsi, M. & Verdot, F.; (1998) ; START: an Industrial System for Teleoperation, Proc. of the IEEE Int. Conf. on Robotics and Automation, Vol. 2, pp. 1154-1159, Leuven (BE)

Mekhnacha, K. ; (1999) ; *Méthodes probabilistes baysiennes pour la prise en compte des incertitudes géométriques : Application à la CAO-robotique* ; Ph.D. thesis INPG (Institut National Polytechnique de Grenoble), Grenoble, France

Mohammad-Djafari, A.& Demoment, G. ; (1992) ; *Maximum entropy and bayesian methods* ; Kluwer Academic Publishers

Pearl, J. ; (1988) ; *Probabilistic reasoning in intelligent systems : Networks of plausible inference* ; Morgan Kaufmann Publishers ; San Mateo, California, USA

Robert, C. ; (1990) ; An entropy concentration theorem: applications ; in artificial intelligence and descriptive statistics ; *Journal of Applied Probabilities*

Robinson, J.A. ; (1965) ; A Machine Oriented Logic Based on the Resolution Principle ; *Jour. Assoc. Comput. Mach.*; vol. 12

Robinson, J.A. ; (1979) ; *Logic : Form and Function* ; North-Holland, New York, USA

Robinson, J.A. & Sibert, E.E. ; (1983a) ; LOGLISP : an alternative to PROLOG ; *Machine Intelligence*, Vol. 10.

Robinson, J.A. & Sibert, E.E. ; (1983b) ; LOGLISP : Motivation, design and implementation ; *Machine Intelligence*, Vol. 10.

Ruiz, A., Lopez-de-Teruel, P.E. and Garrido, M.C. ; (1998) ; Probabilistic Inference from Arbitrary Uncertainty using Mixtures of Factorized Generalized Gaussians ; *Journal of Artificial Intelligence Research* (JAIR) ; Vol. 9, pp. 167-217

Saul, L.K., Jaakkola, T. and Jordan, M.I. ; (1996) ; Mean Field Theory for Sigmoid Belief Networks ; *Journal of Artificial Intelligence Research* (JAIR), Vol. 4, pp. 61-76

Schneider, S.A., Chen, V.W., Pardo-Castellote, G., Wang, H.H.; (1998) ; ControlShell: A Software Architecture for Complex Electromechanical Systems ; *International Journal for Robotics Research* (IJRR) ; Vol. 17(4), pp. 360-380

Smith, C.R. & Grandy, W.T. Jr. ; (1985) ; *Maximum-Entropy and bayesian methods in inverse problems* ; D. Reidel Publishing Company

Tarentola, A. ; (1987) ; *Inverse Problem Theory: Methods for data fitting and model parameters estimation* ; Elsevier ; New York, USA

Thrun, S.; (1998) ; Bayesian landmark learning for mobile robot localization ; *Machine Learning*, vol. 33(1), pp.41-76

Zhang, N.L. and Poole, D. ; (1996) ; Exploiting Causal Independence in Bayesian Network Inference ; *Journal of Artificial Intelligence Research* (JAIR), Vol. 5, pp. 301-328