Bayesian Robot Programming

OLIVIER LEBELTEL, PIERRE BESSIÈRE, JULIEN DIARD AND EMMANUEL MAZER
Laboratoire GRAVIR - CNRS - INRIA, 655 avenue de l'Europe, 38334 St Ismier, France
Pierre.Bessiere@imag.fr

Abstract: We propose a new method to program robots based on Bayesian inference and learning. It is called BRP for Bayesian Robot Programming. 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 BRP 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

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. BRP heavily relies on this Bayesian framework.

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 9 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 10 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 BRP.

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, 2003). 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). Bayesian inference has been proved to be an NP-hard problem (Cooper, 1990). However, very important technical progress has been achieved recently which permits approximated computation in reasonable time (Saul et al., 1996; Zhang & Poole, 1996; Delcher et al., 1996; Darwiche & Provan, 1997; Koller & Pfeffer, 1997; Ruiz et al., 1998; Jaakola & Jordan, 1999; Jordan et al., 1999).

Recent robot programming architectures (Aji & McEliece, 2000; 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 was 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, 1992). More recently Bayesian techniques have been largely used in POMDP[1] to plan complex paths in partially known environments (Kaelbling, Littman & Cassandra, 1996; Kaebling, Cassandra & Kurien, 1996; Koening & Simmons, 1998; Kaebling, Littman & Cassandra, 1998; Beetz & Belker, 2001; Lane & Kaebling, 2001) or for action selection (Rosenblatt, 2000). HMM[2] are also used to plan complex tasks and recognize situations in complex environments (Aycard, 1998, Thrun, 1998). Finally, a lot of works have been done about probabilistic localization and navigation (Shatkay, 1998) either with probabilistic occupancy grids (Konolidge, 1997), Markov localization (Thrun, Burgard & Fox, 1998; Gutmann, et al., 1998; Murphy, 1999; Fox et al., 2000) correlation-based Markov localization (Konolidge & Chou, 1999), Particle filters (Fox et al., 2001) or Kalman filtering (Roumeliotis & Bekey, 2000a; Roumeliotis & Bekey, 2000b).

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 the PhD of Olivier Lebeltel, summarized in the present paper (Lebetel, 1999; Diard & Lebeltel, 1999; Lebeltel et al., 2000; Diard & Lebeltel, 2000). A paper by Thrun (Thrun, 2000) explored this same direction but with less generality. BRP is a simple and generic framework for robot programming in presence of incompletness and uncertainty. It may be used as a unique formalism to restate and compare numerous classical probabilistic models such as for instance, Bayesian Network (BN), Dynamic Bayesian Network (DBN), Bayesian Filters, Hidden Markov Models (HMM), Kalman Filters, Particle Filters, Mixture Models, or Maxim Entropy Models. This is detailed in a survey by Bessiere (Bessière et al., 2003).

Finally, a presentation of the epistemological foundations of BRP may be found in two articles by Bessière (Bessière et al., 1998a; Bessière et al., 1998b).

---

1. Partially Observable Markov Decision Process.
2. Hidden Markov Models.

3. Basic concepts

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

It may be read twice, at first rapidly to acquire the main concepts, and revisited after the instances sections (5 to 11) to understand in detail the formal definitions.

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 propositions 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 ith 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[3]. 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, 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:

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

3.2   Inference postulates and rules

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

---

3. 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.

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.

$$\mathbf{P}(a \wedge b \mid \pi) = \mathbf{P}(a \mid \pi) \times \mathbf{P}(b \mid a \wedge \pi)$$
$$= \mathbf{P}(b \mid \pi) \times \mathbf{P}(a \mid b \wedge \pi)$$

[E3.2]

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

$$\mathbf{P}(a \mid \pi) + \mathbf{P}(\neg a \mid \pi) = 1$$

[E3.3]

For the purpose of this paper, we take these two rules as postulates[4].

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. Indeed, we may derive all the other necessary inference rules from those two, especially the rules concerning variables:

1 Conjunction rule for variables:

$$\mathbf{P}(X \otimes Y \mid \pi) = \mathbf{P}(X \mid \pi) \times \mathbf{P}(Y \mid X \otimes \pi)$$
$$= \mathbf{P}(Y \mid \pi) \times \mathbf{P}(X \mid Y \otimes \pi)$$

[E3.4]

2 Normalization rule for variables:

$$\sum_X \mathbf{P}(X \mid \pi) = 1$$

[E3.5]

3 Marginalization rule for variables

$$\sum_X \mathbf{P}(X \otimes Y \mid \pi) = \mathbf{P}(Y \mid \pi)$$

[E3.6]

3.3 Bayesian Programs

We define a *Bayesian program* as a mean 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:

```
                                                        ⎧Pertinent Variables
                                                        ⎪
                                                        ⎪Decomposition
                         ⎧           ⎧Preliminary        ⎨
                         ⎪           ⎪Knowledge (π)      ⎪      ⎧Parametrical Forms
            ⎧Description ⎨           ⎨                   ⎪Forms ⎨
            ⎪            ⎪           ⎪                   ⎩      ⎩Programs
Program     ⎨            ⎩           ⎩Data (δ)
            ⎪
            ⎩Question
```
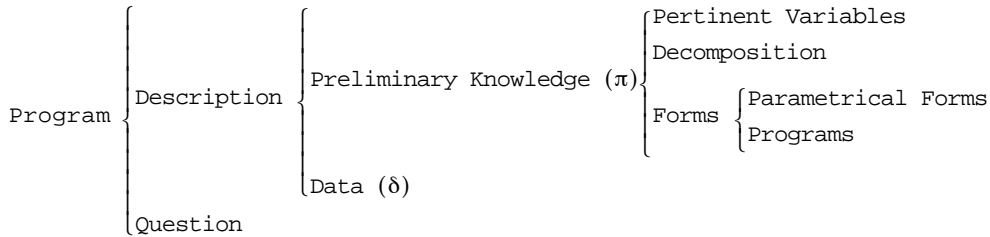
Figure 1: Structure of a Bayesian program

- 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 Bayesian programs.

Description

The purpose of a description is to specify an effective method to compute a joint distribution on

---

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

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 rule [E3.4] leads to:

$$
\begin{aligned}
&\mathbf{P}(X^1 \otimes X^2 \otimes ... \otimes X^n \mid \delta \otimes \pi) \\
&= \mathbf{P}(L^1 \mid \delta \otimes \pi) \times \mathbf{P}(L^2 \mid L^1 \otimes \delta \otimes \pi) \times ... \times \mathbf{P}(L^k \mid L^{k-1} \otimes ... \otimes L^2 \otimes L^1 \otimes \delta \otimes \pi)
\end{aligned} \qquad \text{[E3.7]}
$$

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 $L^{i-1} \otimes ... \otimes L^2 \otimes L^1$, calling R^i the conjunction of these chosen variables and setting:

$$
\mathbf{P}(L^i \mid L^{i-1} \otimes ... \otimes L^2 \otimes L^1 \otimes \delta \otimes \pi) = \mathbf{P}(L^i \mid R^i \otimes \delta \otimes \pi) \qquad \text{[E3.8]}
$$

We then obtain:

$$
\begin{aligned}
&\mathbf{P}(X^1 \otimes X^2 \otimes ... \otimes X^n \mid \delta \otimes \pi) \\
&= \mathbf{P}(L^1 \mid \delta \otimes \pi) \times \mathbf{P}(L^2 \mid R^2 \otimes \delta \otimes \pi) \times \mathbf{P}(L^3 \mid R^3 \otimes \delta \otimes \pi) \times ... \times \mathbf{P}(L^k \mid R^k \otimes \delta \otimes \pi)
\end{aligned} \qquad \text{[E3.9]}
$$

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., $\mathbf{P}(X^1 \otimes X^2 \otimes ... \otimes X^n \mid \delta \otimes \pi)$), a question is obtained by partitioning $\{X^1, X^2, ..., X^n\}$ 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:

$$
\mathbf{P}(Searched \mid Known \otimes \delta \otimes \pi) . \qquad \text{[E3.10]}
$$

3.4   Running Bayesian programs

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

Bayesian inference

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

$$\mathbf{P}(Searched \mid Known \otimes \delta \otimes \pi) = \sum_{Unknown} \mathbf{P}(Searched \otimes Unknown \mid Known \otimes \delta \otimes \pi)$$

$$= \frac{\sum_{Unknown} \mathbf{P}(Searched \otimes Unknown \otimes Known \mid \delta \otimes \pi)}{\mathbf{P}(Known \mid \delta \otimes \pi)}$$

$$= \frac{\sum_{Unknown} \mathbf{P}(Searched \otimes Unknown \otimes Known \mid \delta \otimes \pi)}{\sum_{\substack{Searched \\ Unknown}} \mathbf{P}(Searched \otimes Unknown \otimes Known \mid \delta \otimes \pi)} \qquad \text{[E3.11]}$$

$$= \frac{1}{\Sigma} \times \sum_{Unknown} \mathbf{P}(Searched \otimes Unknown \otimes Known \mid \delta \otimes \pi)$$

$$= \frac{1}{\Sigma} \times \sum_{Unknown} \left[\mathbf{P}(L^1) \times \prod_{i=2}^{k} \mathbf{P}(L^i \mid R^i) \right]$$

where the first equality results from the marginalization rule (equation [E3.6]), the second results from the product rule (equation [E3.4]) 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. Finally, the joint distribution is replaced by its decomposition.

Two main problems have to be solved: searching the modes in a high dimensional space, and marginalizing in a high dimensional space.

Since *Searched* may be a conjunction of numerous variables, each of them possibly having a lot of values or even being continuous, it is seldom possible to exhaustively compute $\mathbf{P}(Searched \mid Known)$. One may then decide either to build an approximate representation of this distribution or to directly sample from this distribution. In both cases the challenge is to find the modes where most of the probability density is concentrated. This may be very difficult, as most of the probability may be concentrated in very small sub-spaces of the whole searched space.

The situation is even worse, as computing the value of $\mathbf{P}(Searched \mid Known)$ for a given value of *Searched* (a single point of the searched space of the preceeding paragraph) is by itself a difficult problem. Indeed, it supposes to marginalize the joint distribution on the space defined by *Unknown*. *Unknown* (like *Searched*) may be a conjunction of numerous variables, each of them possibly having a lot of values or even being continuous. Consequently, the sum should also be either approximated or sampled. The challenge is then to find the modes of

$$\mathbf{P}(L^1) \times \prod_{i=2}^{k} \mathbf{P}(L^i \mid R^i) \qquad \text{[E3.12]}$$

(on the search space defined by *Unknown*), where most of the probability density is concentrated and which mostly contribute to the sum. Finally, marginalizing in a high dimensional space appears to be a very similar problem to searching the modes in a high dimensional space.

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).

However, approximate inference is often tractable in practical cases for three main reasons:

1 The conditional independences, as expressed by the decomposition of the joint distribution, break the complexity of the problem by reducing drastically the size of the searched space (see section 6 for an instance of that). The importance of the decomposition has already been stressed by many authors (e.g., Zhang & Poole, 1996) and explains mainly the good performances of our engine (10 inferences per second[5]).

2 Some powerful symbolic simplifications can be made before any numerical computation (see next section on OPL).

3 Numerical optimization and marginalization have a long history and impressive

---

5. Order of magnitude on a standard desktop computer for the inferences required by the experiments described in the sequel.

numerical methods have been developed which can be reused in this context (see next section on OPL).

OPL: an API to automate Bayesian inference

An inference engine and the associated programming API[6] (named OPL for Open Probabilistic Language) has been developed and used for the experiments presented in this paper and other industrial applications.

OPL proceeds in two phases: a symbolic simplification of the required computation followed by some intensive numerical crunching.

The main goal of the simplification phase is to reduce the number of sums necessary to compute the distribution:

$$\mathbf{P}(Searched \mid Known) = \frac{1}{\Sigma} \times \sum_{Unknown} \left[\mathbf{P}(L^1) \times \prod_{i=2}^{k} \mathbf{P}(L^i \mid R^i) \right] \qquad \text{[E3.13]}$$

These kinds of simplification techniques are largely used in the litterature. For instance, the well known JLO or junction tree algorithm (Jensen, Lauritzen & Olesen, 1990) may be seen as such a simplification technique in the case of Bayesian Networks.

In OPL, a large spectrum of such simplifications is used.

First, considering the different terms of the product $\mathbf{P}(L^1) \times \prod_{i=2}^{k} \mathbf{P}(L^i \mid R^i)$, three possibilities of obvious simplifications may appear:

1 When a term is a uniform distribution it can be simplified: it vanishes from the expression and its value will implicitly be taken into account in the normalization constant Σ.

2 When a term is a distribution where all the variables have *Known* values, then it is a constant for this question and may also be simplified.

3 When a term is a distribution where all the variables are either *Searched* or *Known*, then it can be factorized out of the sum.

After these three first steps, we get a new expression of the form:

$$\mathbf{P}(Searched \mid Known) = \frac{1}{\Sigma} \times \prod_{j \in J} \mathbf{P}(L^j \mid R^j) \times \sum_{Unknown} \left[\prod_{i \in I} \mathbf{P}(L^i \mid R^i) \right] \qquad \text{[E3.14]}$$

Now, considering $\sum_{Unknown} \left[\prod_{i \in I} \mathbf{P}(L^i \mid R^i) \right]$, we can try to find an order on the sum to simplify terms that sum to 1.

Indeed, when a term $\mathbf{P}(L^i \mid R^i)$ appears in the sum, if all the variables appearing in L^i are part of *Unknown* (summed) and if all the variables appearing in R^i are either part of *Known* or *Unknown*, then $\mathbf{P}(L^i \mid R^i)$ sums to 1 and vanishes out of the global sum. This operation often leads to impressive simplifications.

Finally, the last simplification that can be made is to reorder the sums on the different unknown variables in order to minimize the number of operations to make. OPL uses the general distributive law algorithm to do this. A description of this algorithm may be found in a paper by Aji and McEliece (Aji & McEliece, 2000).

A more detailed description of this simplification phase and of related work may be found in Bessière's survey (Bessière et al., 2003).

The main goal of the numerical crunching phase is to estimate the distribution $\mathbf{P}(Searched \mid Known)$. A necessary subgoal is to estimate the corresponding sum.

Two main approaches are possible to reach these objectives, either by building approximated explicit representation of these distributions or by sampling these distributions.

OPL includes different algorithms related to both approaches. It may approximate the distribution using either particle filters (Arulampalam et al., 2001) or Multi Resolution Binary Trees (MRBT), a homemade representation described in a pending patent (Bessière, 2002). OPL also

---

6. Application Programming Interface

uses sampling techniques, mainly Monte Carlo sampling integration methods (Neal, 1993; MacKay, 1996) and an improved version of these techniques proposed by Mekhnacha (Mekhnacha, Mazer & Bessière, 2001), where they are combined with simulated annealing.

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 to this query as: **Draw**($\mathbf{P}(Searched \mid Known \otimes \delta \otimes \pi)$).

Utility functions could also be used to make the decision but they do not appear to be necessary for the work described in this paper.

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 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[7] and is commercialized by K-Team[8].

Figure 2: The Khepera mobile robot

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 $L1, ..., L8$ (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 $Px1, ..., Px8$ (see Figure 3).

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

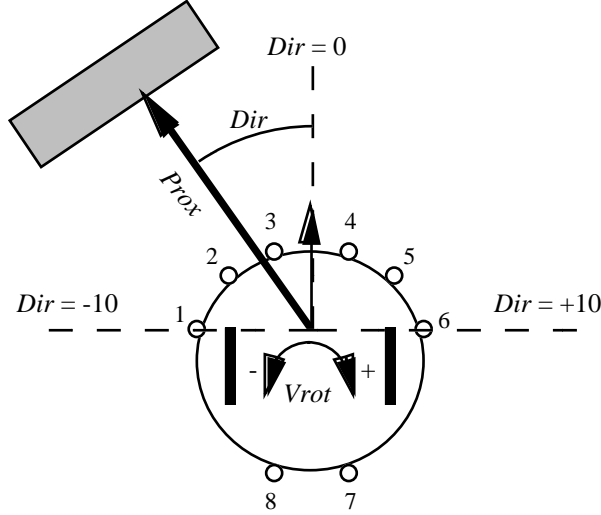
Figure 3: The sensory-motor variables of the Khepera robot.

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:

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

- $Prox$ 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:

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

- $Theta1$ 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 right) and -10 (fastest to the left).

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 con-

figuration 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 (see Figure 12).

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:

$$Dir \in \{-10, ..., 10\}, \lfloor Dir \rfloor = 21$$
$$Prox \in \{0, ..., 15\}, \lfloor Prox \rfloor = 16 \qquad \text{[S5.1]}$$
$$Vrot \in \{-10, ..., 10\}, \lfloor Vrot \rfloor = 21$$

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.

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

The first equality results from the application of the product rule (equation [E3.4]). 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:

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

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 parametric 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[9]), 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 by computing the means and standard deviations of $Vrot$ for each position of the obstacle.

Finally, it is possible to compute the joint distribution:

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

According to equation [E3.11], 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:
```

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

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

5.5   Results, lessons and comments

Results

As shown in Movie 1[9], 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[10], 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.

---

9. http://www-laplace.imag.fr/ENGLISH/PRESENTATION/Semaine-Science/Trans7/T7.mov
10. http://www-laplace.imag.fr/ENGLISH/PRESENTATION/Semaine-Science/Trans8/T8.mov

Lesson 1: A generic method for Bayesian Robot Programming

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 - Identification: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 BRP experiments.

Lesson 2: Bayesian Program = Preliminary Knowledge + Data + Question

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:

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

- 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[11]). Two new descrip-
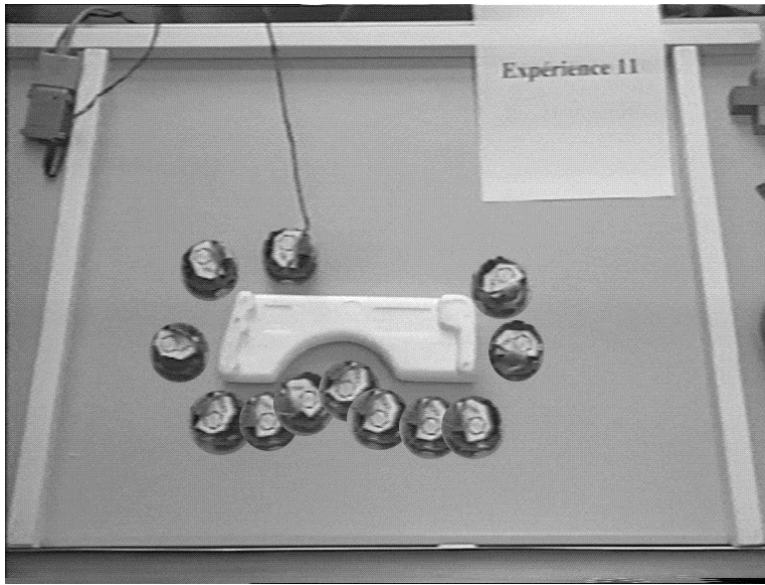

Figure 4: Contour following (superposed images)

tions[12] were obtained by changing only the driving of the robot during the 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 $\pi\text{-}phototaxy1$ uses the variables $Vrot$ and $Theta1$. $Theta1$ roughly corresponds to the direction of the light.

---

11. http://www-laplace.imag.fr/ENGLISH/PRESENTATION/Semaine-Science/Trans9/T9.mov

12. $\mathbf{P}(Dir \otimes Prox \otimes Vrot \mid \delta\text{-}avoid \otimes \pi\text{-}obstacle)$ and $\mathbf{P}(Dir \otimes Prox \otimes Vrot \mid \delta\text{-}follow \otimes \pi\text{-}obstacle)$

6. Sensor fusion

6.1   Goal and experimental protocol

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

6.2   Sensor model

Specification

Variables

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

$$Li \in \{0, ..., 511\}, \lfloor Li \rfloor = 512$$
$$Theta2 \in \{-170, ..., 180\}, \lfloor Theta2 \rfloor = 36 \qquad [\text{S6.1}]$$

Decomposition

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

$$\mathbf{P}(Theta2 \otimes Li \mid \Delta \otimes \pi\text{-}sensor)$$
$$= \mathbf{P}(Theta2 \mid \pi\text{-}sensor) \times \mathbf{P}(Li \mid Theta2 \otimes \Delta \otimes \pi\text{-}sensor) \qquad [\text{S6.2}]$$

Parametric forms

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

$$\mathbf{P}(Theta2 \mid \pi\text{-}sensor) \equiv Uniform \qquad [\text{S6.3}]$$

   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 5):
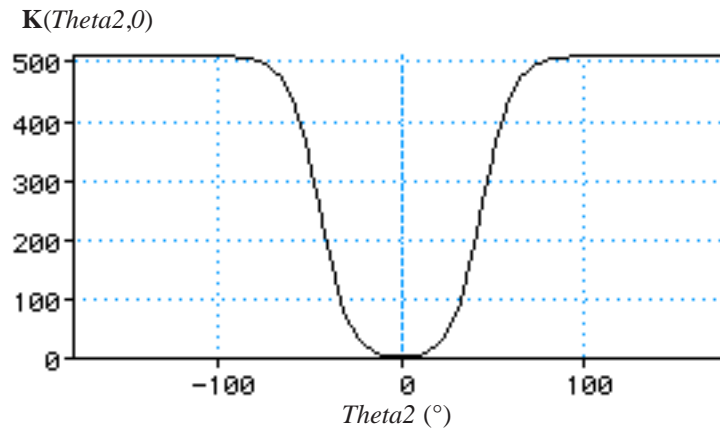

Figure 5: $\mathbf{K}(Theta2, 0)$

$$\mathbf{P}(Li \mid Theta2 \otimes \pi\text{-}sensor) \equiv \mathbf{G}_{\mathbf{K}(Theta2,\,\theta_i),\,\sigma}(Li)$$

$$\mathbf{K}(Theta2, \theta_i) \;=\; 1 - \frac{1}{1 + e^{-4\beta(|Theta2 - \theta_i| - \alpha)}} \qquad (\alpha = 45),\,(\beta = 0{,}03) \qquad \text{[S6.4]}$$

In specification [S6.4], 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 [S6.1], [S6.2], [S6.3] and [S6.4] 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.

6.3   Fusion

Specification

Variables

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

$$L1 \in \{0, \ldots, 511\}, \lfloor L1 \rfloor = 512$$
$$\ldots$$
$$L8 \in \{0, \ldots, 511\}, \lfloor L8 \rfloor = 512 \qquad \text{[S6.5]}$$
$$Theta2 \in \{-170, \ldots, 180\}, \lfloor Theta2 \rfloor = 36$$

Decomposition

The decomposition of the joint distribution is chosen to be:

$$\mathbf{P}(Theta2 \otimes L1 \otimes L2 \otimes L3 \otimes L4 \otimes L5 \otimes L6 \otimes L7 \otimes L8 \mid \Delta \otimes \pi\text{-}fusion)$$

$$= \mathbf{P}(Theta2 \mid \Delta \otimes \pi\text{-}fusion) \times \mathbf{P}(L1 \mid Theta2 \otimes \Delta \otimes \pi\text{-}fusion) \times \mathbf{P}(L2 \mid L1 \otimes Theta2 \otimes \Delta \otimes \pi\text{-}fusion)$$
$$\ldots \times \mathbf{P}(L8 \mid L7 \otimes L6 \otimes L5 \otimes L4 \otimes L3 \otimes L2 \otimes L1 \otimes Theta2 \otimes \Delta \otimes \pi\text{-}fusion) \qquad \text{[S6.6]}$$
$$= \mathbf{P}(Theta2 \mid \pi\text{-}fusion) \times \prod_{i\,=\,1}^{8} \mathbf{P}(Li \mid Theta2 \otimes \Delta \otimes \pi\text{-}fusion)$$

The first equality results from the product rule [E3.4]. The second from simplifications of the kind:

$$\mathbf{P}(Lj \mid Lj-1 \otimes \ldots \otimes L1 \otimes Theta2 \otimes \Delta \otimes \pi\text{-}fusion) \;=\; \mathbf{P}(Lj \mid Theta2 \otimes \Delta \otimes \pi\text{-}fusion) \qquad \text{[E6.1]}$$

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 sensors 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 $Theta2$:

$$\mathbf{P}(\textit{Theta2} \mid \pi\text{-}\textit{fusion}) \equiv \textit{Uniform} \tag{S6.7}$$

$\mathbf{P}(\textit{Li} \mid \textit{Theta2} \otimes \Delta \otimes \pi\text{-}\textit{fusion})$ is obtained from the model of each sensor as specified in previous section (6.2):

$$\mathbf{P}(\textit{Li} \mid \textit{Theta2} \otimes \Delta \otimes \pi\text{-}\textit{fusion}) \equiv \mathbf{P}(\textit{Li} \mid \textit{Theta2} \otimes \pi\text{-}\textit{sensor}) \tag{S6.8}$$

Identification

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

Utilization

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

$$\mathbf{Draw}(\mathbf{P}(\textit{Theta2} \mid l1_t \otimes ...... \otimes l8_t \otimes \pi\text{-}\textit{fusion})) \tag{E6.1}$$

This question may be easily answered using equation [E3.11] and specification [S6.8]:

$$
\begin{aligned}
&\mathbf{P}(\textit{Theta2} \mid l1_t \otimes ... \otimes l8_t \otimes \pi\text{-}\textit{fusion}) \\
&= \frac{1}{\Sigma} \times \prod_{i=1}^{8} \mathbf{P}(li_t \mid \textit{Theta2} \otimes \pi\text{-}\textit{sensor})
\end{aligned}
\tag{E6.2}
$$

Values drawn from this distribution may be efficiently computed given that the distribution $\mathbf{P}(\textit{Theta2} \mid l1_t \otimes ... \otimes l8_t \otimes \pi\text{-}\textit{fusion})$ 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:

$$
\begin{aligned}
&\mathbf{P}(\textit{Theta2} \mid l1_t \otimes l2_t \otimes \pi\text{-}\textit{fusion}) \\
&= \frac{1}{\Sigma} \times \mathbf{P}(l1_t \mid \textit{Theta2} \otimes \pi\text{-}\textit{sensor}) \times \mathbf{P}(l2_t \mid \textit{Theta2} \otimes \pi\text{-}\textit{sensor})
\end{aligned}
\tag{E6.3}
$$

- 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:

$$
\begin{aligned}
&\mathbf{P}(l1_t \mid l2_t \otimes ... \otimes l8_t \otimes \pi\text{-}\textit{fusion}) \\
&= \frac{1}{\Sigma} \times \sum_{\textit{Theta2}} \prod_{i=1}^{8} \mathbf{P}(li_t \mid \textit{Theta2} \otimes \pi\text{-}\textit{sensor})
\end{aligned}
\tag{E6.4}
$$

6.4   Results, lessons and comments

Results

Figure 6 presents the result obtained for a light source with a bearing of 10°:

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

Lesson 3: Breaking the complexity using conditional independences

The conditional independencies hypothesis which permits to transform:

$$\mathbf{P}(\textit{Theta2} \otimes \textit{L1} \otimes \textit{L2} \otimes \textit{L3} \otimes \textit{L4} \otimes \textit{L5} \otimes \textit{L6} \otimes \textit{L7} \otimes \textit{L8} \mid \Delta \otimes \pi\text{-}\textit{fusion}) \tag{E6.5}$$

into:
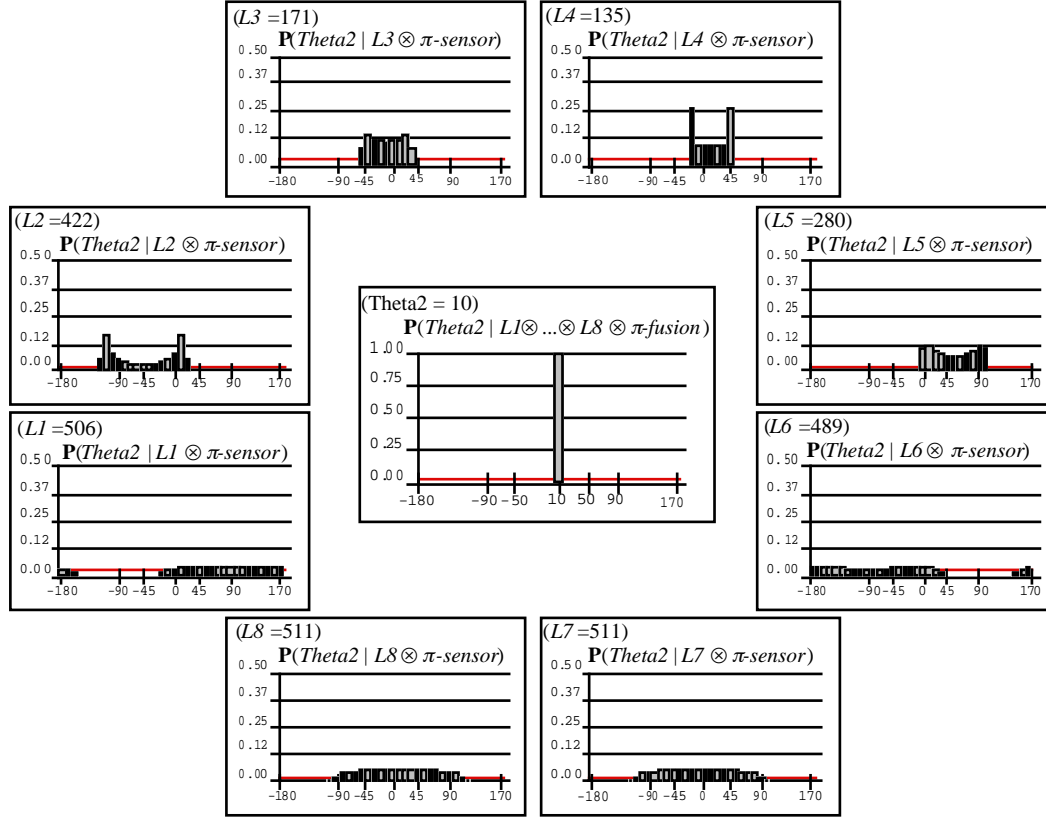
Figure 6: The result of a sensor fusion for a light source with a bearing of 10°

$$\mathbf{P}(\mathit{Theta2} \mid \pi\text{-}\mathit{fusion}) \times \prod_{i=1}^{8} \mathbf{P}(\mathit{Li} \mid \mathit{Theta2} \otimes \Delta \otimes \pi\text{-}\mathit{fusion}) \qquad [\text{E6.6}]$$

is the main tool at hand to simplify the treated problem. More than any clever inference algorithm they are the essential way to keep computation tractable. For instance, here the size of the search space for the joint distribution [E] is $36 \times 512^8 \propto 2^{79}$, when the size of the search space for the decomposition [E6.6] is $36 \times (512 \times 36) \times 8 \propto 2^{22}$.

Lesson 4: Calling Bayesian subroutines

Specification [S6.8]: $\mathbf{P}(\mathit{Li} \mid \mathit{Theta2} \otimes \Delta \otimes \pi\text{-}\mathit{fusion}) \equiv \mathbf{P}(\mathit{Li} \mid \mathit{Theta2} \otimes \pi\text{-}\mathit{sensor})$, where a distribution appearing in a decomposition is defined by a question to another Bayesian program, may be seen as the probabilistic anologous to subroutine calls in regular programming.

This Bayesian subroutine call mechanism will play the same role than the usual one: allowing to build complex Bayesian programs as hierarchies of embedded calls to simpler and simpler Bayesian programming building blocks. Section 10 will present a more complex instance of this.

Lesson 5: 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 very efficient computations.

We presented this method on a very simple case for didactic purposes. However it is a very popular technique to do sensor fusion which can be used for much more complicated cases in a large variety of applications. For instance, we used refinement of this technique for ADAS (Advanced Driver Assistance System) to merge information coming from two radars and one lidar in order to partially automate car driving (Coué et al., 2002, Coué et al., 2003).

Lesson 6: No inverse and no ill-posed problems in the probabilitic framework

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. Consequently one may define the description in one way ($\mathbf{P}(Li \mid Theta2)$) and question it in the opposite way ($\mathbf{P}(Theta2 \mid l1_t \otimes ...... \otimes l8_t)$). In theory, any inverse problem may be solved when expressed in a probabilistic framework. In practice some of these inverse problems may require a lot of computational ressources. However, this is a major difference with non probabilistic modelling where inverse problem may only be solved in rare cases.

Furthermore, there is none ill-posed problem in a probabilistic framework. If a question may have several solutions, the probabilistic answer will simply have several peaks.

7.   Behavior combination

7.1   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 a «command variable».

7.2   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.

$$
\begin{aligned}
&Dir \in \{-10, ..., 10\}, \lfloor Dir \rfloor = 21 \\
&Prox \in \{0, ..., 15\}, \lfloor Prox \rfloor = 16 \\
&Theta1 \in \{-170, ..., 180\}, \lfloor Theta1 \rfloor = 36 \\
&Vrot \in \{-10, ..., 10\}, \lfloor Vrot \rfloor = 21 \\
&H \in \{avoidance, phototaxy\}, \lfloor H \rfloor = 2
\end{aligned}
\qquad \text{[S7.1]}
$$

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:

$$\mathbf{P}(Dir \otimes Prox \otimes Theta1 \otimes H \otimes Vrot \mid \Delta \otimes \pi\text{-}home)$$
$$= \mathbf{P}(Dir \mid \pi\text{-}home) \times \mathbf{P}(Prox \mid \pi\text{-}home) \times \mathbf{P}(Theta1 \mid \pi\text{-}home) \times \mathbf{P}(H \mid Prox \otimes \pi\text{-}home) \quad \text{[S7.2]}$$
$$\times \mathbf{P}(Vrot \mid Dir \otimes Prox \otimes Theta1 \otimes H \otimes \pi\text{-}home)$$

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:

$$\mathbf{P}(Dir \mid \pi\text{-}home) \equiv Uniform$$
$$\mathbf{P}(Prox \mid \pi\text{-}home) \equiv Uniform \quad \text{[S7.3]}$$
$$\mathbf{P}(Theta1 \mid \pi\text{-}home) \equiv Uniform$$

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:

$$\mathbf{P}(Vrot \mid Dir \otimes Prox \otimes Theta1 \otimes avoidance \otimes \pi\text{-}home) \equiv \mathbf{P}(Vrot \mid Dir \otimes Prox \otimes \delta\text{-}avoid \otimes \pi\text{-}obstacle)$$
$$\mathbf{P}(Vrot \mid Dir \otimes Prox \otimes Theta1 \otimes phototaxy \otimes \pi\text{-}home) \equiv \mathbf{P}(Vrot \mid Theta1 \otimes \delta\text{-}phototaxy \otimes \pi\text{-}phototaxy1) \quad \text{[S7.4]}$$

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

$$\mathbf{P}(avoidance \mid Prox \otimes \pi\text{-}home) \equiv \mathbf{Sigmoid}_{\alpha, \beta}(Prox) \qquad (\alpha = 9), (\beta = 0, 25)$$
$$\mathbf{P}(phototaxy \mid Prox \otimes \pi\text{-}home) = 1 - \mathbf{P}(avoidance \mid Prox \otimes \pi\text{-}home) \qquad \text{[S7.5]}$$

The discrete approximation of the Sigmoid function we use above, which will not be defined

P(*avoidance* | *Prox* ⊗ *π-home*)
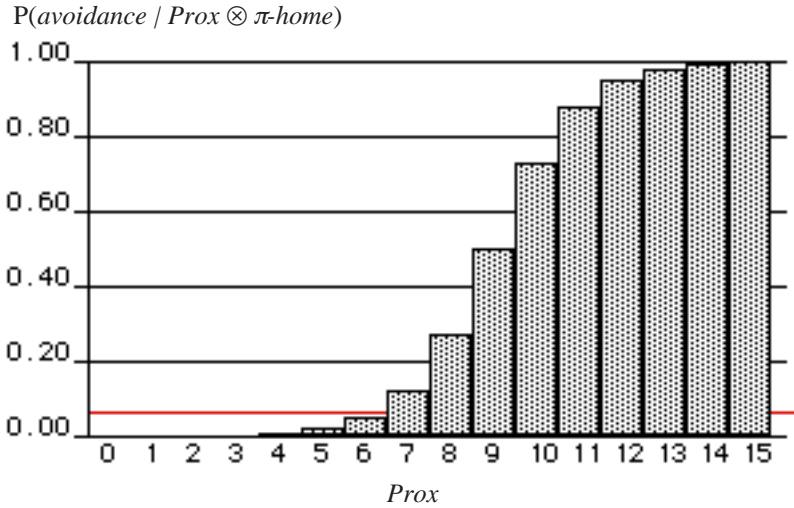

Figure 7: $\mathbf{P}(avoidance \mid Prox \otimes \pi\text{-}home)$

in the current paper, is shown in Figure 7.

The preliminary knowledge $\pi\text{-}home$ is defined by specifications [S7.1], [S7.2], [S7.3], [S7.4] and [S7.5].

7.3   Identification

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

7.4   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:

$$\mathbf{P}(Vrot \mid Dir \otimes Prox \otimes Theta1 \otimes \pi\text{-}home)$$

$$= \sum_{H} \mathbf{P}(Vrot \otimes H \mid Dir \otimes Prox \otimes Theta1 \otimes \pi\text{-}home)$$

[E7.1]

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

Equation [E7.1] 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.

7.5   Results, lessons and comments

Results

Figure 8 and Movie 4[13] show efficient homing behavior obtained this way.

Figure 8: Homing behavior (The arrow points out the light source)
(superposed images).

Figures 9 and 10 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 9). 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 10).

---

13. http://www-laplace.imag.fr/ENGLISH/PRESENTATION/Semaine-Science/Trans10/T10.mov

(*Dir* = -5, *Prox* = 10, *Lum* = -90)

P(*Vrot* | *Theta1* ⊗ δ*phototaxy* ⊗ π*-phototaxy1*)

P(*Vrot* | *Dir* ⊗ *Prox* ⊗ δ*-avoid* ⊗ π*-obstacle*)

P(*H* | *Prox* ⊗ π*-home*)

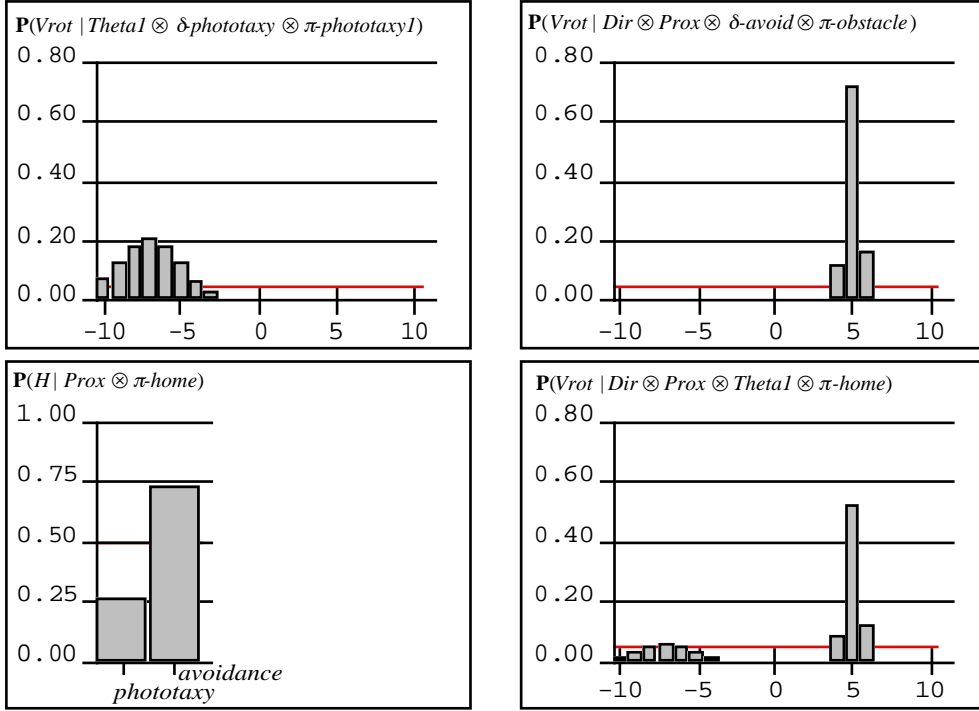**P**(*Vrot* | *Dir* ⊗ *Prox* ⊗ *Theta1* ⊗ π*-home*)

Figure 9: Homing behavior (Khepera close to an object on its left and has also the light source 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*.

(*Dir* = -5, *Prox* = 8, *Lum* = -90)

P(*Vrot* | *Theta1* ⊗ δ*-phototaxy* ⊗ π*-phototaxy1*)

P(*Vrot* | *Dir* ⊗ *Prox* ⊗ δ*-avoid* ⊗ π*-obstacle*)

P(*H* | *Prox* ⊗ π*-home*)

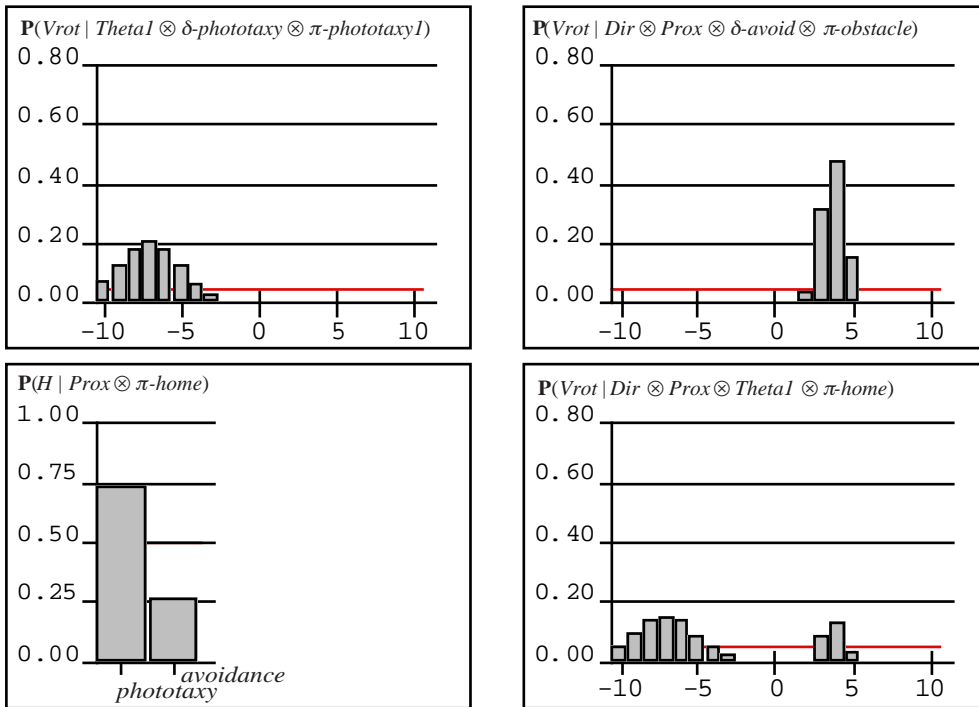**P**(*Vrot* | *Dir* ⊗ *Prox* ⊗ *Theta1* ⊗ π*-home*)

Figure 10: Homing behavior (Khepera further from the object on its left). This figure is structured as Figure 9.

Lesson 7: A probabilistic if-then-else

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[14]. 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[15] (Jordan & Jacobs, 1994) and is also closely related to mixture models (see McLachlan & Deep, 2000 for a reference document about mixture models and see Bessière et al., 2003 for detail about the relation between description combination and mixture models).

Finally, from a programming point of view, description combination can be seen as a probabilistic if-then-else. H is the condition. If H is known with certainty then we have a usual branching structure. If H is known with some uncertainty through a probability distribution then the 2 possible consequences are automatically mixed using weights proportional to this distribution.

8.   Situation recognition

8.1   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 (see below for a precise definition). 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.

8.2   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:

$$Nlt \in \{0, ..., 24\}, \lfloor Nlt \rfloor = 25$$
$$Nrt \in \{0, ..., 24\}, \lfloor Nrt \rfloor = 25$$
$$Per \in \{0, ..., 9999\}, \lfloor Per \rfloor = 10000 \qquad \text{[S8.1]}$$
$$Lrl \in \{0, ..., 999\}, \lfloor Lrl \rfloor = 1000$$
$$O \in \{0, ..., 15\}, \lfloor O \rfloor = 16$$

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 6), we consider that knowing the

---

14. see (Diard & Lebeltel, 1999)
15. Hierachical Mixture of Expert

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:

$$
\begin{aligned}
&\mathbf{P}(O \otimes Nlt \otimes Nrt \otimes Per \otimes Lrl \mid \Delta \otimes \pi\text{-}object) \\
&= \mathbf{P}(O \mid \pi\text{-}object) \times \mathbf{P}(Nlt \mid O \otimes \Delta \otimes \pi\text{-}object) \times \mathbf{P}(Nrt \mid O \otimes \Delta \otimes \pi\text{-}object) \\
&\quad \times \mathbf{P}(Per \mid O \otimes \Delta \otimes \pi\text{-}object) \times \mathbf{P}(Lrl \mid O \otimes \Delta \otimes \pi\text{-}object)
\end{aligned}
\qquad \text{[S8.2]}
$$

Parametric forms

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

$$
\mathbf{P}(O \mid \pi\text{-}object) \equiv Uniform
\qquad \text{[S8.3]}
$$

For an already observed object ($O \neq 0$), we state that the distributions on *Nlt* and *Nrt* are Laplace succession laws[16] and that the distributions on *Per* and *Lrl* are Gaussian laws:

$$
\forall o_i \in O, o_i \neq o_0
$$
$$
\mathbf{P}(Nlt \mid o_i \otimes \Delta \otimes \pi\text{-}object) \equiv \mathbf{L}_1(n_{Nlt}(o_i))
$$
$$
\mathbf{P}(Nrt \mid o_i \otimes \Delta \otimes \pi\text{-}object) \equiv \mathbf{L}_2(n_{Nrt}(o_i))
\qquad \text{[S8.4]}
$$
$$
\mathbf{P}(Per \mid o_i \otimes \Delta \otimes \pi\text{-}object) \equiv \mathbf{G}_1(\mu(o_i), \sigma(o_i))
$$
$$
\mathbf{P}(Lrl \mid o_i \otimes \Delta \otimes \pi\text{-}object) \equiv \mathbf{G}_2(\mu(o_i), \sigma(o_i))
$$

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

$$
\mathbf{P}(Nlt \mid o_0 \otimes \pi\text{-}object) \equiv Uniform
$$
$$
\mathbf{P}(Nrt \mid o_0 \otimes \pi\text{-}object) \equiv Uniform
$$
$$
\mathbf{P}(Per \mid o_0 \otimes \pi\text{-}object) \equiv Uniform
\qquad \text{[S8.5]}
$$
$$
\mathbf{P}(Lrl \mid o_0 \otimes \pi\text{-}object) \equiv Uniform
$$

The preliminary knowledge composed of specifications [S8.1], [S8.2], [S8.3], [S8.4] and [S8.5] is named $\pi\text{-}object$.

8.3   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 $\mathbf{P}(O \otimes Nlt \otimes Nrt \otimes Per \otimes Lrl \mid \delta_n \otimes \pi\text{-}object)$.

8.4   Utilization

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

$$
\mathbf{P}(O \mid nlt_n \otimes nrt_n \otimes per_n \otimes lrl_n \otimes \delta_{n-1} \otimes \pi\text{-}object)
\qquad \text{[E8.1]}
$$

This may be simply computed by:

---

16. A Laplace succession law on a variable V is defined by: $\dfrac{1 + n_v}{N + \lfloor V \rfloor}$ 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.

$$\mathbf{P}(O \mid Nlt \otimes Nrt \otimes Per \otimes Lrl \otimes \delta_{n-1} \otimes \pi\text{-}object)$$

$$= \frac{1}{\Sigma} \times \mathbf{P}(Nlt \mid O \otimes \delta_{n-1} \otimes \pi\text{-}object) \times \mathbf{P}(Nrt \mid O \otimes \delta_{n-1} \otimes \pi\text{-}object) \qquad \text{[E8.2]}$$

$$\times \mathbf{P}(Per \mid O \otimes \delta_{n-1} \otimes \pi\text{-}object) \times \mathbf{P}(Lrl \mid O \otimes \delta_{n-1} \otimes \pi\text{-}object)$$

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.

8.5   Results, lessons and comments

Results.

Figure 11: The different objects presented to Khepera.

The 15 objects shown on Figure 11 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?» mathematicaly stated as $\mathbf{P}(O \mid nlt_n \otimes nrt_n \otimes per_n \otimes lrl_n \otimes \delta_{n-1} \otimes \pi\text{-}object)$. The obtained results are presented on table 1.

It should be first noticed that two objects (top right of the picture, 3 and 7 in the order of first presentation) have the exact same square basis and thus may not be distinguished from one another given the four chosen variables. In these 2 cases, Khepera was in the position of someone asked to identify the color of an object by groping it.

The robot did not ever fail to recognize novelty (column 1) but for the first presentation of object 7 which was recognized as object 3.

At the second presentation of each object the robot recognized the object except for object 7 which was always identified as 3 and but once for object 12 (the wood pyramid) which was also recognized as 3.

	F	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14															
1		4														
2			4													
3	1			4				4					1			
4					4											
5						4										
6							4									
7								0								
8									4							
9										4						
10											4					
11												4				
12													3			
13														4		
14															4	
15																4

Table 1: Situation Recognition.
The columns correspond to the «really» presented objects.
The first column (F) corresponds to the first presentation of an object and the lines to the recognize objects.

Lesson 8: Categorization

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 6. The hypotheses are similar and the advantages are the same.

9. Temporal sequences

9.1   Goal and experimental protocol

In this section, to exemplify the Bayesian programming method, we choose a «night watchman 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 on 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.

9.2   Specification

Variables

The first variable to consider is $Behavior$, which may take 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 :

$$Behavior \in \{idle, patrol, recognition, fire-intervention, homing, recharge\}, \lfloor Behavior \rfloor = 6$$
$$Vigil \in \{true, false\}, \lfloor Vigil \rfloor = 2$$
$$Energy \in \{very-high, high, low, very-low\}, \lfloor Energy \rfloor = 4$$
$$Base \in \{true, false\}, \lfloor Base \rfloor = 2 \qquad \text{[S9.1]}$$
$$Fire \in \{true, false\}, \lfloor Fire \rfloor = 2$$
$$Identify \in \{true, false\}, \lfloor Identify \rfloor = 2$$
$$Behavior\_t\text{-}1 \in \{idle, patrol, ..., recharge\}, \lfloor Behavior\_t\text{-}1 \rfloor = 6$$

Decomposition

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

$$\mathbf{P}(Behavior \mid Vigil \otimes Energy \otimes Base \otimes Fire \otimes Identify \otimes Behavior\_t\text{-}1 \otimes \pi\text{-}behavior) \qquad \text{[E9.1]}$$

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:

$$
\begin{aligned}
&\mathbf{P}(Behavior \otimes Vigil \otimes Energy \otimes Base \otimes Fire \otimes Identify \otimes Behavior\_t\text{-}1 \mid \pi\text{-}behavior) \\
&= \mathbf{P}(Behavior\_t\text{-}1 \mid \pi\text{-}behavior) \times \mathbf{P}(Behavior \mid Behavior\_t\text{-}1 \otimes \pi\text{-}behavior) \\
&\quad \times \mathbf{P}(Vigil \mid Behavior \otimes \pi\text{-}behavior) \times \mathbf{P}(Energy \mid Behavior \otimes \pi\text{-}behavior) \qquad \text{[S9.2]} \\
&\quad \times \mathbf{P}(Base \mid Behavior \otimes \pi\text{-}behavior) \times \mathbf{P}(Fire \mid Behavior \otimes \pi\text{-}behavior) \\
&\quad \times \mathbf{P}(Identify \mid Behavior \otimes \pi\text{-}behavior)
\end{aligned}
$$

Parametric forms

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

$$\mathbf{P}(Behavior\_t\text{-}1 \mid \pi\text{-}behavior) \equiv Uniform \qquad \text{[S9.3]}$$

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, **P**(*Behavior* | *Behavior_t-1* ⊗ *π-behavior*) is specified by table 2.

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 3 specifies the probabilities of the variable *Behavior* knowing that the behavior of the 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 set 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 stay in the current 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 2: **P**(*Behavior* | *Behavior_t-1* ⊗ *π-behavior*)

Table 3 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 3: **P**(*Vigil* | *Behavior* ⊗ *π-behavior*)

Table 4 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 4: **P**(*Energy* | *Behavior* ⊗ *π-behavior*)

Table 5 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
false	0	0.99	0.99	x	0.99	0.99
true	1	0.01	0.01	x	0.01	0.01

Table 5: **P**(*Base* | *Behavior* ⊗ *π-behavior*)

Table 6 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 6: **P**(*Fire* | *Behavior* ⊗ *π-behavior*)

Finally, Table 7 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 7: **P**(*Identify* | *Behavior* ⊗ *π-behavior*)

9.3 Identification

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

9.4 Utilization

The robot chooses its behavior with the following query:

$$\mathbf{Draw}(\mathbf{P}(Behavior \mid Vigil \otimes Energy \otimes Base \otimes Fire \otimes Identify \otimes Behavior\_t\text{-}1 \otimes \pi\text{-}behavior)) \qquad [E9.1]$$

that can be easily computed:

$$\mathbf{P}(Behavior \mid Vigil \otimes Energy \otimes Base \otimes Fire \otimes Identify \otimes Behavior\_t\text{-}1 \otimes \pi\text{-}behavior)$$

$$= \frac{1}{\Sigma} \times \mathbf{P}(Behavior\_t\text{-}1 \mid \pi\text{-}behavior) \times \mathbf{P}(Behavior \mid Behavior\_t\text{-}1 \otimes \pi\text{-}behavior)$$

$$\times \mathbf{P}(Vigil \mid Behavior \otimes \pi\text{-}behavior) \times \mathbf{P}(Energy \mid Behavior \otimes \pi\text{-}behavior)$$

$$\times \mathbf{P}(Base \mid Behavior \otimes \pi\text{-}behavior) \times \mathbf{P}(Fire \mid Behavior \otimes \pi\text{-}behavior) \times \mathbf{P}(Identify \mid Behavior \otimes \pi\text{-}behavior)$$

[E9.2]

9.5 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[17]).

For instance, these sequences are stable. Khepera does not behave like a weathercock that changes its mind every second.

---

17. http://www-laplace.imag.fr/ENGLISH/PRESENTATION/Semaine-Science/Trans12/T12.small.mov

Lesson 9: 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 it is possible to learn these distributions (see Diard & Lebeltel, 1999).

Furthermore, the stability of the behavior according to the values in the tables is a critical question. We do not have yet any quantified results to answer this question but we are trying to set up an experimental protocol to evaluate it.

10. Integration: A Night watchman Task

10.1 Goal and experimental protocol

The practical goal and experimental protocol of the night watchman task has already been presented in Section 9.1.

The scientific purpose of this last experiment is to prove that Bayesian robot 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 - $\mathbf{P}(Base \otimes Px1 \otimes \dots \otimes Px8 \otimes L1 \otimes \dots \otimes L8 | \pi\text{-}base)$ used by Khepera to decide if it is at its base

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

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

10.2 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*).

$$
\begin{aligned}
\textit{Sensory-variables} \equiv\ & \textit{Px1} \otimes \dots \otimes \textit{Px8} \otimes \textit{L1} \otimes \dots \otimes \textit{L8} \\
& \otimes \textit{Vigil} \otimes \textit{Energy} \otimes \textit{Fire} \otimes \textit{Identify} \otimes \textit{Behavior\_t-1} \\
& \otimes \textit{Move\_t-1} \otimes \textit{Tempo} \otimes \textit{Tour} \otimes \textit{Dir} \otimes \textit{Prox} \otimes \textit{DirL} \otimes \textit{ProxL} \otimes \textit{Vtrans\_c} \\
& \otimes \textit{Nlt} \otimes \textit{Nrt} \otimes \textit{Per} \otimes \textit{Lrl}
\end{aligned}
\tag{E10.1}
$$

- Five internal variables: *Base, Theta2, Behavior, Move, H*
- Three «motor» variables that Khepera must compute. These three variables are the rotation speed *Vrot* , the translation speed *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 ($\mathbf{P}$(*Sensory-variables*|π*-watchman*)) and eight questions addressed to the previously defined descriptions:

$$\mathbf{P}\left(\begin{array}{c} \textit{Sensory-variables} \\ \textit{Base} \otimes \textit{Theta2} \otimes \textit{Behavior} \otimes \textit{Move} \otimes H \\ \textit{Vrot} \otimes \textit{Vtrans} \otimes O \end{array} \middle| \pi\textit{-watchman}\right)$$

$$= \mathbf{P}(\textit{Sensory-variables}|\pi\textit{-watchman})$$
$$\times \mathbf{P}(\textit{Base}|Px1 \otimes \dots \otimes Px8 \otimes L1 \otimes \dots \otimes L8 \otimes \pi\textit{-base})$$
$$\times \mathbf{P}(\textit{Theta2} \mid L1 \otimes L2 \otimes L3 \otimes L4 \otimes L5 \otimes L6 \otimes L7 \otimes L8 \otimes \pi\textit{-fusion}) \qquad \text{[E10.2]}$$
$$\times \mathbf{P}(\textit{Behavior} \mid \textit{Vigil} \otimes \textit{Energy} \otimes \textit{Base} \otimes \textit{Fire} \otimes \textit{Identify} \otimes \textit{Behavior\_t-1} \otimes \pi\textit{-behavior})$$
$$\times \mathbf{P}(\textit{Move} \mid \textit{Behavior} \otimes \textit{Move\_t-1} \otimes \textit{Tempo} \otimes \textit{Tour} \otimes \pi\textit{-move})$$
$$\times \mathbf{P}(H \mid \textit{Prox} \otimes \pi\textit{-home})$$
$$\times \mathbf{P}(\textit{Vrot} \otimes \textit{Vtrans} \mid \textit{Move} \otimes H \otimes \textit{Dir} \otimes \textit{Prox} \otimes \textit{DirL} \otimes \textit{ProxL} \otimes \textit{Vtrans\_c} \otimes \textit{Theta2} \otimes \pi\textit{-speed})$$
$$\times \mathbf{P}(O \mid \textit{Nlt} \otimes \textit{Nrt} \otimes \textit{Per} \otimes \textit{Lrl} \otimes \pi\textit{-object})$$

10.3 Identification

No identification is required.

10.4 Utilization

The ultimate question that Khepera must answer is:

$$\mathbf{P}(\textit{Vrot} \otimes \textit{Vtrans} \otimes O \mid \textit{Sensory-variables} \otimes \pi\textit{-watchman}) \qquad \text{[E10.3]}$$

«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:

$$\mathbf{P}(\textit{Vrot} \otimes \textit{Vtrans} \otimes O \mid \textit{Sensory-variables} \otimes \pi\textit{-watchman})$$

$$= \frac{1}{\Sigma} \times \mathbf{P}(O \mid \textit{Nlt} \otimes \textit{Nrt} \otimes \textit{Per} \otimes \textit{Lrl} \otimes \pi\textit{-object})$$

$$\times \sum_{\substack{\textit{Move} \\ \textit{Theta2} \\ H}} \left(\begin{array}{l} \sum_{\textit{Behavior}} \left(\begin{array}{l} \mathbf{P}(\textit{Move} \mid \textit{Behavior} \otimes \dots \otimes \textit{Tour} \otimes \pi\textit{-move}) \times \\ \sum_{\textit{Base}} \left(\begin{array}{l} \mathbf{P}(\textit{Behavior} \mid \textit{Vigil} \otimes \dots \otimes \textit{Behavior\_t-1} \otimes \pi\textit{-behavior}) \\ \times \mathbf{P}(\textit{Base}|Px1 \otimes \dots \dots \otimes L8 \otimes \pi\textit{-base}) \end{array} \right) \end{array} \right) \\ \times \mathbf{P}(\textit{Theta2} \mid L1 \otimes \dots \otimes L8 \otimes \pi\textit{-fusion}) \\ \times \mathbf{P}(H \mid \textit{Prox} \otimes \pi\textit{-home}) \\ \times \mathbf{P}(\textit{Vrot} \otimes \textit{Vtrans} \mid \textit{Move} \otimes \dots \otimes \textit{Theta2} \otimes \pi\textit{-speed}) \end{array} \right) \qquad \text{[E10.4]}$$

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*

$$\sum_{\textit{Base}} \left(\begin{array}{l} \mathbf{P}(\textit{Behavior} \mid \textit{Vigil} \otimes \dots \otimes \textit{Behavior\_t-1} \otimes \pi\textit{-behavior}) \\ \times \mathbf{P}(\textit{Base}|Px1 \otimes \dots \otimes L8 \otimes \pi\textit{-base}) \end{array} \right) \qquad \text{[E10.5]}$$
$$= \mathbf{P}(\textit{Behavior} \mid \textit{Vigil} \otimes \dots \otimes \textit{Behavior\_t-1} \otimes Px1 \otimes \dots \otimes L8 \otimes \pi\textit{-watchman})$$

- 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(P(Base|Px1 ⊗ … ⊗ L8 ⊗ π-base))  to decide if the robot is at its base,
2 -  Draw(P(Behavior | … ⊗ Base ⊗ … ⊗ π-behavior))  to decide the Behavior knowing Base,
3 - Draw(P(Move | Behavior ⊗ … ⊗ π-move))  to chosoe a movement knowing the
Behavior,
4 - Draw(P(Theta2 | L1 ⊗ … ⊗ L8 ⊗ π-fusion))  to decide the position of the light
source,
5 - Draw(P(H | Prox ⊗ π-home))  to decide between avoidance and phototaxy,
6 - and finally,  Draw(P(Vrot ⊗ Vtrans | … ⊗ π-speed))  to control the robot.
```

10.5   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 public science fair.

The Movie 5[17] 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),
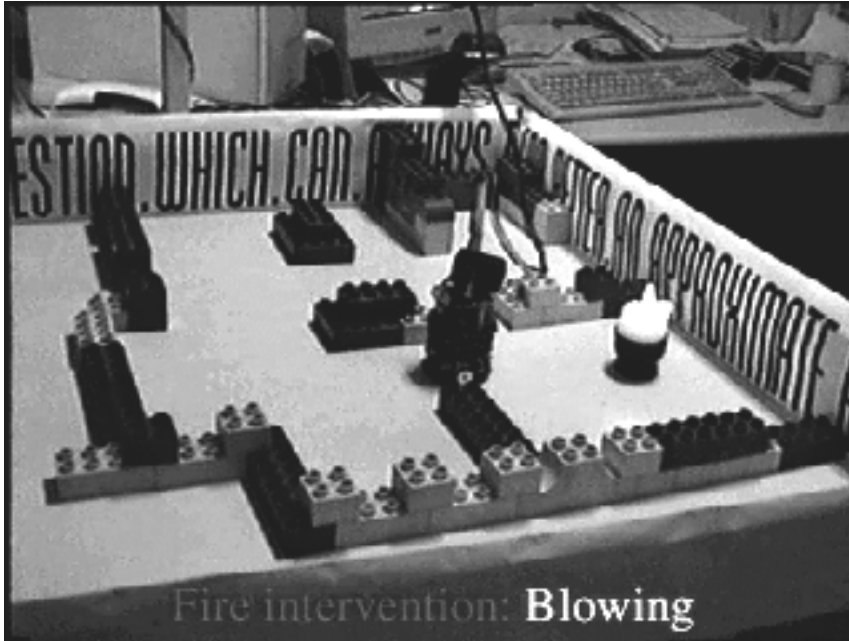- Khepera returning to its base.

Figure 12: The night watchman task.

11. Synthesis

11.1   Principles, theoretical foundation and methodology

Principles

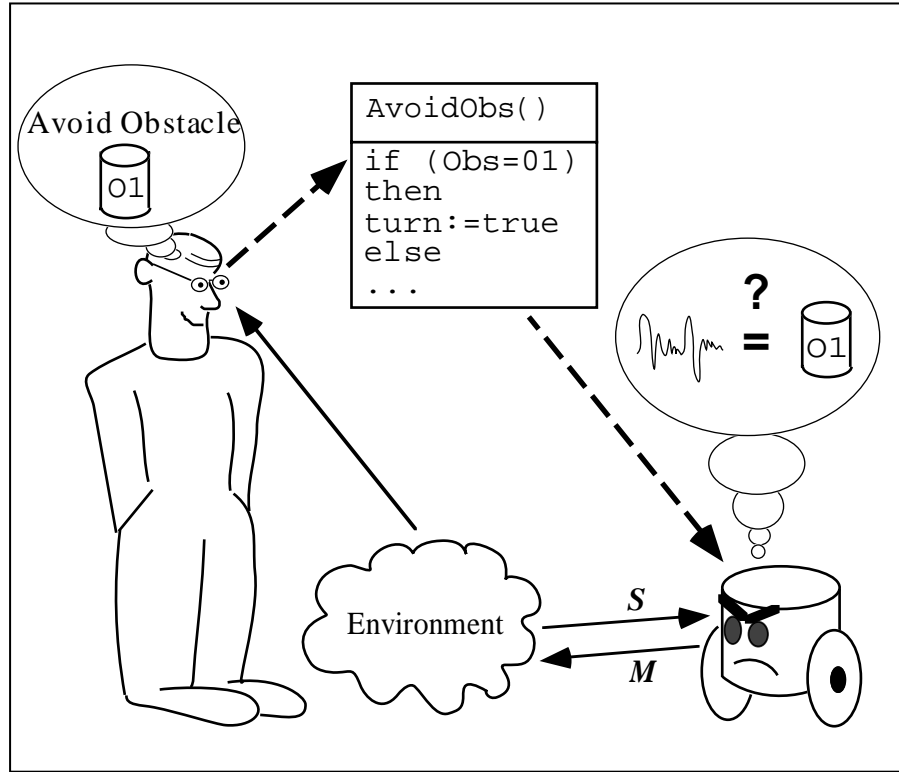The dominant paradigm in robotics may be caricatured by Figure 13.

Figure 13: The symbolic approach in robotics.

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 of the world can be specified. He or she may 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 (BRP) as a possible solution.

Figure 14 presents the principles of this approach.

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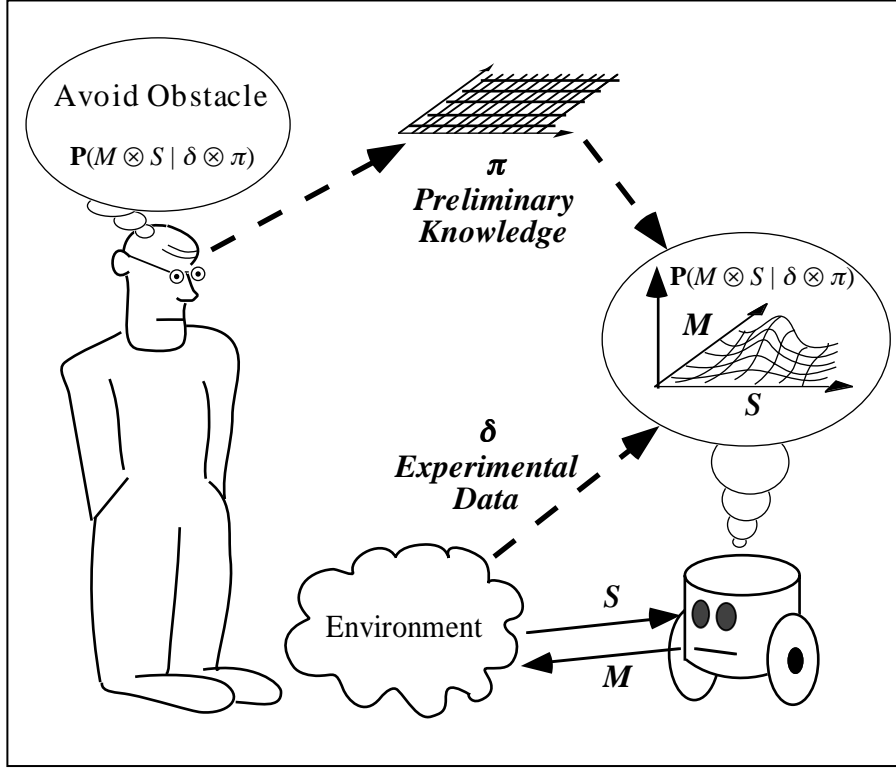

Figure 14: The BRP approach in robotics.

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 15.

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

Incompletness

Preliminary Knowledge
+
Experimental Data
=
Probability Distributions

Maximum Entropy
Principle

$-\Sigma\, p_i \ln(p_i)$

Uncertainty

Bayesian Inference

$\mathbf{P}(a \wedge b\, |\pi) = \mathbf{P}(a\, |\pi)\mathbf{P}(b\, |a \wedge \pi) = \mathbf{P}(b\, |\pi)\mathbf{P}(a\, |b \wedge \pi)$

$\mathbf{P}(\neg a\, |\pi) + \mathbf{P}(\neg a\, |\pi) = 1$
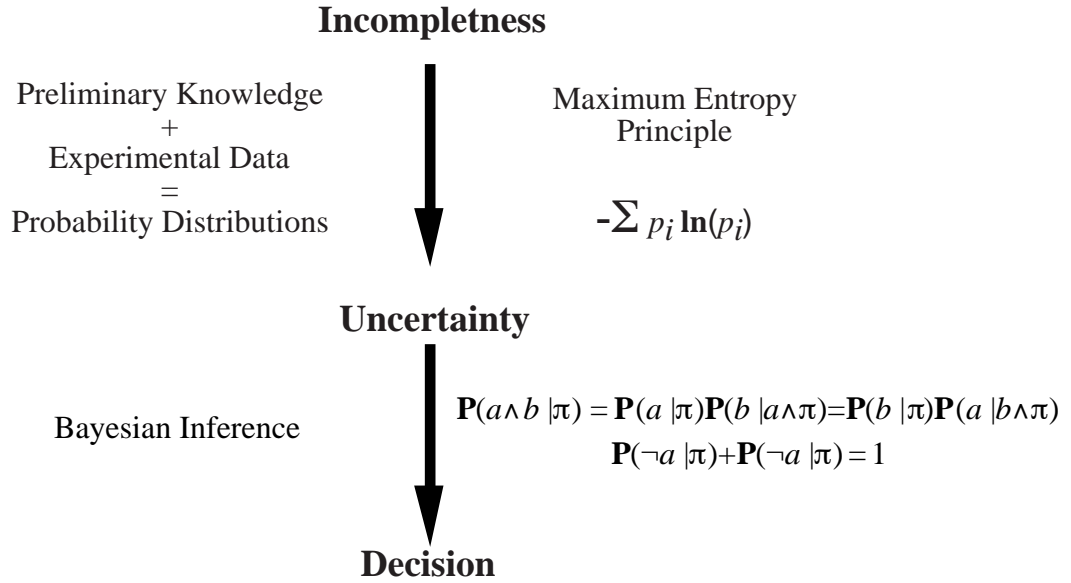
Decision

Figure 15: Theoretical foundation.

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» distributions, 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 illustrated in the different examples in this paper.

11.2 Advantages

In this section we survey, comment and briefly discuss the advantages of the BRP 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 BRP 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 Section 8 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 6 about sensor fusion or Section 8 about object recognition). The «nightwatchman» task (see Section 10) 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*: BRP 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.

• *Homogeneity of representations and resolution processes*: BRP 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 robotic 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 6), to recognize situations (Section 8), to carry out temporal sequencing (Section 9) and finally, to specify a task integrating all the previously defined descriptions (Section 10).

° *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 6), situation recognition (Section 8) or inverse programming (Section 9) 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 6.3).

- *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 can really be used in practice. The final experiment (Section 10) 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 8). 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 able for 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 PhD. thesis (Dedieu, 1995).

11.3 BRP in practice

BRP is a very simple, practical and efficient way to program robots for the 3 main following reasons:

- BRP proposes a generic methodology for robot programming:
```
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.
```
- The conditional independences are used to break the complexity and keep the computation tractable.
- BRP may really be considered as a programming language because it proposes three main tools to build complex programs from simpler ones:
  - ° Calling Bayesian subroutines.
  - ° Probabilistic if-then-else.
  - ° Inverse programming to sequence behaviors.

11.4 Scalability

This paper presented only very simple instances of robot programming for didactic purposes. The goal was to illustrate the fundamental principles rather than to present applications.

Consequently two fundamental questions are still open: is BRP scalable? may it be used for practical robotics applications?

Elements of answer to these two questions may be found in may be found in other works that used the BRP approach:

- Kamel Mekhnacha build a Bayesian robotic CAD system which offers the usual capa-

bilities of such a system but also handles the uncertainties. This system works with hundreds of variables and tens of probability distributions (Mekhnacha, 1999; Mekhnacha, Mazer & Bessière, 2000; Mekhnacha, Mazer & Bessière, 2001)

- Julien Diard proposed in his PhD thesis the concept of Bayesian maps, which is a specialization of the BRP formalism, for representing the environment in navigation tasks. In particular, the descriptions and the combination operators considered in this work are special cases of the one presented here (Diard, 2003).
- Christophe Coué is using BRP for an Advanced Driving Assistance System (ADAS) (Coué et al., 2002; Coué et al., 2003).

Ongoing works include manipulation with 6 DOF arms, pick and place problems, videogame Bot programming, outdoor SLAM with automotive vehicles and indoor service robotics.

References

Aji S. M. and McEliece R. J. ; (2000) ; The Generalized Distributive Law ; *IEEE Trans. Information Theory*, Vol. 46, No. 2Alami, 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

Arulampalam, S., Maskell, S., Gordon, N. & Clapp, T. ; (2001) ; A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking ; *IEEE Transactions on Signal Processing* citeseer.nj.nec.com/maskell01tutorial.html

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

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

Beetz, M. & Belker, T.; (2001); Learning strucured reactive navigation plans from executing MDP navigation policies; in Proceedings of Agents 2001

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

Bessière, Pierre; (2002) ; Procédé de détermination de la valeur à donner à différents paramètres d'un système ; Demande de brevet d'invention n°0235541154.4 ; Institut Européen des Brevets

Bessière P. & the BIBA-INRIA Research Group ; (2003) ; Survey: Probabilistic Methodology and Techniques for Artefact Conception and Devlopment ; INRIA Technical Report RR-4730, (http://www.inria.fr/rrrt/rr-4730.html)

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

Boutilier, C., Reiter, R., Soutchanski, M. & Thrun, S.; (2000); Decision-Theoretic, High-level Agent Programming in the Situation Calculus; in Proceedings of AAAI 2000

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

Coué, C. & Bessière, P. ; (2001) ; Chasing an elusive target with a mobile robot ; IEEE/IROS 2001 ; Maui, Hawaii, USA

Coué, C., Fraichard, Th., Bessière, P. & Mazer, E.; (2002); Multi-Sensor Data Fusion Using Bayesian Programming: an Automotive Application; Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems (IROS)

Coué, C., Fraichard, Th., Bessière, P. & Mazer, E.; (2003); Using Bayesian Programming for Multi-Sensor Multi-Target Tracking in Automotive Applications; Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA); in Press

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 ;

Diard J. & Lebeltel O. (2000); Bayesian programming and hierarchical learning in robotics; Meyer, Berthoz, Floreano, Roitblat and Wilson (Eds) ; SAB2000 Proceedings Supplement Book ; Publication of the International Society for Adaptive Behavior, Honolulu.

Diard, J.; (2003); *La carte bayésienne : Un modèle probabiliste hiérarchique pour la navigation en robotique mobile*; PhD thesis, Institut National Polytechnique de Grenoble (INPG), 27 janvier 2003

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

Fox, D., Burgard, W., Kruppa, H. & Thrun, S. ; (2000) ; A Probabilistic Approach to Collaborative Multi-Robot Localization ; *Autonomous Robots*, Vol. 8, pp. 325-344

Fox, D., Thrun, S., Dellaert, F. & Burgard, W. ; (2001) ; Particle filters for mobile robot localization; in Doucet A., de Freitas N. & Gordon N. (eds); *Sequential Monte Carlo Methods in Practice*; Spinger-Verlag, New York, USA

Gutmann, J-S., Burgard, W., Fox, D. & Konolidge, K. ; (1998) ; Experimental comparison of localization methods ; Intereanational Conference on Intelligent Robots and Systems

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. ; (2003) ; *Probability theory - The logic of science* ; Cambridge University Press (in press)

Jensen, F., Lauritzen, S. & Olesen, K. ; (1990) ; Bayesian updating in recursive graphical models by local computations ; *Computational Statistical Quaterly*, 4:269-282

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

Kaebling, L.P., Cassandra, A.R. & Kurien, J.A. ; (1996) ; Acting Under Uncertainty: Discrete Bayesian Models for Mobile-Robot Navigation, in Proceedings of IEEE/RSJ International Conference onIntelligent Robots and Systems

Kaebling, L.P., Littman, M.L. & Cassandra, A.R. ; (1998) ; Planning and Acting in Partially Observable Stochastic Domains ; *Artificial Intelligence*, Vol. 101

Kapur, J.N., & Kesavan, H.K. ; (1992) ; *Entropy optimization principles with applications* ; Academic PressKoening, S. & Simmons, R.; (1998); A robot navigation architecture based on partially observable Markov decision process models; in Kortenkamp D., Bonasso, R.P. & Murphy, R. (eds); *Mobile Robots and Artificial Intelligence*; AAAI Press

Koller, D., & Pfeffer, A. ; (1997) ; Object-Oriented Bayesian networks ; Proceedings of the 13th Annual Conference on Uncertainty in AI (UAI) ; Providence, Rhode Island, USA

Konolidge, K. ; (1997) ; Improved Occupancy Grids for Map Building ; *Autonomous Robots*, Vol. 4, pp. 351-367

Konolidge, K. & Chou, K. ; (1999) ; Markov Localization using Correlation ; International Joint Conference on Artificial Intelligence, Stockolm, Sweden

Lane, T. & Kaebling, L.P. ; (2001) ; Toward hierachical decomposition for Planning in Uncertain Environments ; Workshop on planning under Uncertainty and Incomplete Information at the 2001 International Joint Conference on Artificial Intelligence (IJCAI-2001)

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* ; PhD. Thesis, Institut National Polytechnique de Grenoble (INPG); Grenoble, France

Lebeltel, O., Diard, J., Bessière, P. & Mazer, E. (2000); A bayesian framework for robotic programming; Twentieth International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering (MaxEnt 2000) ; Paris, 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

MacKay, D.G. ; (1996) ; Introduction to Monte Carlo Methods ; Proc. of an Erice summer school, ed. M. Jordan

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)

McLachlan G.J. & Deep D. ; (2000) ; *Finite Mixture Models* ; Wiley, New York, USA

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

Mekhnacha, K., Mazer, E. & Bessière, P. ; (2000) ; A Robotic CAD system using a Bayesian framework ; In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2000, *Best Paper Award*), Vol. 3, pp. 1597-1604, Takamatsu, Japan

Mekhnacha, K., Mazer, E. & Bessière, P. ; (2001) ; The design and implementation of a Bayesian CAD modeler for robotic applications ; *Advanced Robotics*, Vol. 15, N. 1

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

Murphy, K. ; (1999) ; Bayesian Map Learning in Dynamic Environments ; in Proceedings of NIPS 99

Neal Radford M. ; (1993) ; *Probabilistic inference using Markov chain Monte-Carlo Methods* ; Technical Report, CRG-TR-93-1, university of Toronto

Parr, R. & Russell, S.; (1998); Reinforcement Learning with Hierarchies of Machines; in Proceedings of NIPS 1998

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.

Rosenblatt, J.K. ; (2000) ; Optimal Slection of Uncertain Actions by Maximizing Expected Utility ; *Autonomous Robots*, Vol. 9, pp. 17-25

Roumeliotis S.I. and Bekey G. ; (2000) ; Collective localization: a distributed kalman filter approach to localization of groups of mobile robots . In IEEE International Conference on Robotics and Automation

Roumeliotis S.I. and Bekey G.A. ; (2000) ; Bayesian estimation and Kalman Filtering: A unified framework for Mobile Robot Localization . In Proc. IEEE Int. Conf. on Robotics and Automation, pages 2985-2992, San Fransisco, CA.

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

Hagit Shatkay ; (1998) ; *Learning Models for Robot Navigation* ;PhD. dissertation and Technical Report cs-98-11, Brown University, Department of Computer Science, Providence, RI

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

Thrun, S., Burgard, W., Fox, D. ; (1998) ; A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots ; *Autonomous Robots*, Vol. 5, pp 253-271

Thrun, S. ; (2000) ; Towards programming tools for robots that integrate probabilistic computation and learning ; in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)

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