Bayesian Programming and Hierarchical Learning in Robotics

Julien Diard, Olivier Lebeltel

Bayesian Programming and Hierarchical Learning in Robotics

Julien Diard

Équipe LAPLACE,
Laboratoire LEIBNIZ - CNRS,
46, avenue Félix Viallet
38031 Grenoble ; France
Julien.Diard@imag.fr

Olivier Lebeltel

Équipe LAPLACE,
Laboratoire LEIBNIZ - CNRS,
46, avenue Félix Viallet
38031 Grenoble ; France
Olivier.Lebeltel@imag.fr

Abstract

This paper presents a new robotic programming environment based on the probability calculus. We show how reactive behaviours, like obstacle avoidance, contour following, or even light following, can be programmed and learned by a Khepera robot with our system. We further demonstrate that behaviours can be combined either by programmation or learning. A homing behaviour is thus obtained by combining obstacle avoidance and light following.

1. Introduction

We propose a new robotic programming environment, which was tested on a Khepera robot. This system is based on the probability calculus. The choice of probabilities as a formal system allows an easy and rigorous translation of intuitive knowledge into a program. An example is the expression of dependence or independence between variables. In order to program a behaviour, the programmer will first have to state such a priori knowledge about the task at hand. This "seed" of program can then be tuned by confronting it to experimental data which the programmer gathers while showing to the robot the expected behaviour. Probability calculus gives the means for playing back the knowledge stored in the program. By presenting a few examples, chosen mainly for their clarity, we will show that, with this method, programming and combining reactive behaviours is easy.

The plan of this paper is as follows. After explaining this paper's contribution with respect to previous works and briefly presenting the Khepera robot, we present the mathematical background needed to understand our programming method. The main component of a program is a *description* which basically defines a joint probability distribution over relevant variables. Next we will give examples of various reactive behaviours which can easily be programmed with our system. The last section of this paper introduces a method for combining reactive behaviours; we give an example where we combine obstacle avoidance with light following to obtain a behaviour used by the Khepera to return to its base. References to much more detailed presentations, addressing all difficult technical points and debates, can be found in the text.

2. Related work

Our work is based on an implementation of the principle of the Bayesian theory of probabilities. In physics, since the precursor work of Laplace (Laplace, 1774, Laplace, 1814) numerous results have been obtained using Bayesian inference techniques (to take into account uncertainty) and maximum entropy principle (to take into account incompleteness). The late Edward T. Jaynes has proposed a rigorous and synthetic formalization of probabilistic reasoning with his "Probability as Logic" theory (Jaynes, 1998). An historic of this approach is proposed by Jaynes (Jaynes, 1979) and an epistemological analysis by Matalon (Matalon, 1967). Theoretical justifications of probabilistic inference and maximum entropy are numerous, the entropy concentration theorems (Jaynes, 1982, Robert, 1990) are the more rigorous, Cox theorem (Cox, 1961) the most well known even if this last one has been lately partially disputed by Halpern (Halpern, 1999a, Halpern, 1999b). Numerous applications and mathematical tools using bayesian inference have been developed (Smith and Grandy, 1985, Tarentola, 1987, Bretthorst, 1988, Erickson and Smith, 1988a, Erickson and Smith, 1988b, Mohammad-Djafari and Demoment, 1992, Kapur and 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 trend only since the proposal of Bayesian nets (Pearl, 1988) and graphical models (Lauritzen and Spiegelhalter, 1988,

Lauritzen, 1996, Jordan, 1998, Frey, 1998). Very important technical progress have been achieved lately (See for instance the JAIR[1] articles on that subject : (Saul et al., 1996, Zhang and Poole, 1996, Delcher et al., 1996, Darwiche and Provan, 1997, Ruiz et al., 1998, Jaakkola and Jordan, 1999, Jordan et al., 1999)).

Recent robot programming architectures (Alami et al., 1998, Borrelly et al., 1998, Schneider et al., 1998, Dekhil and Henderson, 1998, Mazer et al., 1998) are in general not concerned with the problem of uncertainty. In robotics, the uncertainty topic is either related to calibration (Bernhardt and Albright, 1993) or to planning problems (Brafman et al., 1997). In this later case some authors have considered modeling the uncertainty of the robot motions to plan assembly operations (Lozano-Perez et al., 1984, Donald, 1988) or modelling the uncertainty related to the position of the robot on the scene (Kapur and Kesavan, 1992). Bayesian techniques are used in POMDP to plan complex paths in partially known environments (Kaelbling et al., 1996). HMM are also used to plan complex tasks and recognize situations in complex environments (Aycard, 1998). Finally, a couple of recent successful robots used probabilities in modelling sensing, acting, localization and mapping (Thrun, 1998b, Thrun, 1998a). But, to the best of our knowledge, the design of a robot programming system and architecture solely based on bayesian inference has never been otherwise investigated.

Finally, a presentation of the epistemological foundations of the approach described in this paper may be found in two articles by Bessière et al. (Bessière et al., 1998a, Bessière et al., 1998b), and all the technical details in the Ph.D. dissertation of Olivier Lebeltel and a related technical report (Lebeltel, 1999, Lebeltel et al., 2000).

3. The Khepera robot

The Khepera (see Figure 1) is a miniature mobile robot built by the EPFL (Ecole Polytechnique Fédérale de Lausanne) and produced by K-Team. The Khepera is a mobile robot with two wheels, is 57 mm in diameter and 29 mm tall, for a total weight of 80 g, in the basic configuration. It is equipped with eight light sensors (6 in front and 2 behind) having values ranging from 0 to 511, values decreasing with increasing light (variables $Lm0$ to $Lm7$). These eight sensors can also be used as infrared proximity sensors, with values from 0 to 1023 as a decreasing function of the obstacle distance (variables $Px0$ to $Px7$). It also has rudimentary odometry capacities that are not very reliable. The Khepera is piloted using the left and right wheel speed, using variables $Vleft$ and $Vright$. However, for convenience, we actually use the rotation and translation speed obtained by

Figure 1: A picture of the Khepera.

$$\begin{cases} Vleft &= Vtrans + Vrot \\ Vright &= Vtrans - Vrot \end{cases} \quad (1)$$

4. Bayesian robot programming system

In this section, we will describe our programming method. We introduce the few concepts, definitions, notations and rules which are necessary to understand the calculus and experiments presented in this paper.

4.1 Basic concepts

In our programming system, the manipulated objects are *logical propositions*, which can be composed using the usual operators and properties : if α and β are propositions, then $\alpha \wedge \beta$ (or $\alpha\beta$) denotes the conjunction of α and β, $\neg\alpha$ denotes the negation of α, and so forth. We can now define *discrete variables*. A discrete variable X corresponds to a set E_X of k_X logical propositions $[X = x_i]$ such that these propositions are mutually exclusive ($[X = x_i] \wedge [X = x_j]$ is false unless $i = j$) and exhaustive (at least one of the proposition $[X = x_i]$ is true). When introducing variables, we will merely give the domain E_X of possible values for that variable, along with its cardinal k_X. The conjunction $X \otimes Y$ (or simply XY) of two variables X and Y then corresponds to the set of $k_X k_Y$ propositions $[X = x_i] \wedge [Y = y_j]$. XY corresponds to a set of mutually exclusive and exhaustive logical propositions; as such, it is a new variable[2].

To be able to deal with uncertainty, we will attach probabilities to propositions. These probabilities will always be assigned with respect to some *preliminary knowledge* π. Therefore, $P(\alpha|\pi)$ denotes the probabil-

[1] Journal of Artificial Intelligence Research

[2] In contrast, the disjunction of two variables is not a variable, since the associated propositions $[X = x_i] \vee [Y = y_j]$ are not mutually exclusive.

ity that the proposition α is true, knowing π; $P(\alpha\beta|\gamma\pi)$ denotes the probability that the conjunction $\alpha\beta$ is true, knowing γ and π; finally, by convention, if X is a variable, $P(X|\pi)$ means $\forall x_i \in E_X, P([X = x_i]|\pi)$.

To manipulate probabilities, we will use classical inference rules, namely the product rule and normalization rule. The product rule for variables is written

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

and is also known as Bayes theorem. The normalization rule states that

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

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

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

Given a set of n variables $X_1, X_2, \ldots, X_n$, a *question* is defined as a partition of this set in three subsets ξ_S, ξ_K and ξ_U, for the sets of searched, known and unknown variables, respectively. Let $Searched$, $Known$ and $Unknown$ be the conjunctions of the variables in ξ_S, ξ_K and ξ_U, respectively. Given the joint distribution

$$P(X_1X_2\cdots X_n|\pi) = P(Searched \otimes Known \otimes Unknown|\pi),$$

it is possible to answer any question. We first have to compute the probability distribution $P(Searched|Known \otimes \pi)$. The derivation is as follows :

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

where Z is a normalization constant. Answering the question consists in deciding a value for the variable $Searched$ according to the distribution $P(Searched|Known \otimes \pi)$. Different decision policies are possible, in our programming system we usually choose to draw a value at random according to that distribution.

Please refer to (Jaynes, 1998) for a full-length presentation of probabilities and bayesian inference.

4.2 Programs

In our robotic programming method, a program consist of two components, a *description* and a *question*. We show Figure 2 an example of a contour following program.

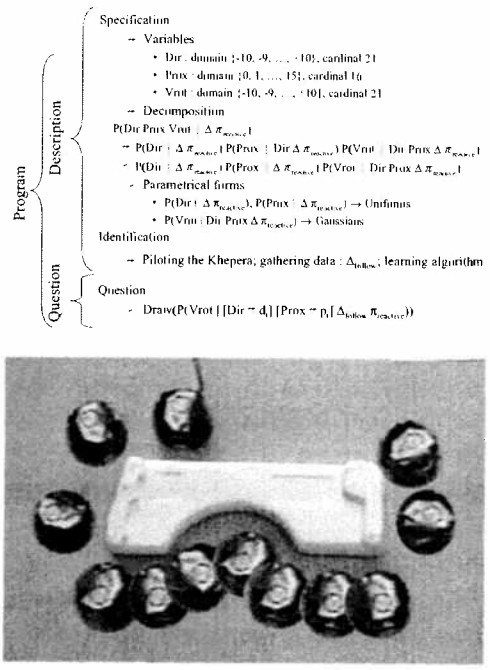

Figure 2: Above, an example of a program. It shows both the program structure our method defines, and an example where the robot follows contours. Below, a picture of the Khepera following the contour of an object.

A description can be viewed as a knowledge base, in which the programmer gives information to the robot about the task to be performed. A description consists of two parts, a *specification* phase, which aims at defining the parametrical form of a joint distribution over relevant variables (the preliminary knowledge π), and an *identification* phase, where the parameters are defined, eventually by gathering data Δ experimentally and assessing the parameter values by a learning mechanism. Therefore, a description is defined by a pair $< \pi, \Delta >$, the latter one being eventually empty if there is no identification phase.

The specification phase has three components :

- **Variables** The programmer specifies which variables are relevant for the task. In our running example, we choose to sum up the information given by the six front proximeters the Khepera has, in the two variables Dir and $Prox$. Dir roughly corresponds to the direction leading to the nearest object, with lowest (resp. highest) value -10 (resp. +10) for an obstacle on the left (resp. right). It is computed from the proximeter values by :

$$Dir = \left\lfloor \frac{90(Px5 - Px0) + 45(Px4 - Px1) + 5(Px3 - Px2)}{9(1 + Px0 + Px1 + Px2 + Px3 + Px4 + Px5)} \right\rfloor.$$

$Prox$ varies with the proximity of that object, varying from 0 (no obstacle) to 15 (contact), and is obtained by :

$$Prox = \lfloor (Max(Px0, Px1, Px2, Px3, Px4, Px5))/(64) \rfloor .$$

Concerning motor variables, we choose to set the robot's translation speed to a constant and keep only one degree of freedom, the rotation speed $Vrot$ of the robot. These three variables, Dir, $Prox$ and $Vrot$, are the only ones considered for our contour following task; their definitions are summed up as follows :

- Dir : domain $\{-10, -9, \ldots, +10\}$, cardinal $k_{Dir} = 21$,
- $Prox$: domain $\{0, +1, \ldots, +15\}$, cardinal $k_{Prox} = 16$,
- $Vrot$: domain $\{-10, -9, \ldots, +10\}$, cardinal $k_{Vrot} = 21$.

- **Decomposition of the joint distribution** The second specification step consists in giving a decomposition of the joint probability distribution $P(Dir \otimes Prox \otimes Vrot | \Delta \otimes \pi_{reactive})$, as a product of simpler terms :

$$\begin{aligned} & P(Dir \otimes Prox \otimes Vrot | \Delta \otimes \pi_{reactive}) \\ = \; & P(Dir | \Delta \otimes \pi_{reactive}) \times P(Prox | Dir \otimes \Delta \otimes \pi_{reactive}) \\ & \times P(Vrot | Dir \otimes Prox \otimes \Delta \otimes \pi_{reactive}) \\ = \; & P(Dir | \Delta \otimes \pi_{reactive}) \times P(Prox | \Delta \otimes \pi_{reactive}) \\ & \times P(Vrot | Dir \otimes Prox \otimes \Delta \otimes \pi_{reactive}). \end{aligned}$$

The first equality results from the application of the product rule, while the second one simplifies $P(Prox|Dir \otimes \Delta \otimes \pi_{reactive})$ in $P(Prox|\Delta \otimes \pi_{reactive})$. This simply means that we consider $Prox$ and Dir independent for our task; we think that the robot can ignore the relation between the distance and direction of obstacles, and yet be successful at following their contour.

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

$$\begin{aligned} P(Dir | \Delta \otimes \pi_{reactive}) & \equiv Uniform, \\ P(Prox | \Delta \otimes \pi_{reactive}) & \equiv Uniform, \end{aligned}$$

$$\begin{aligned} & P(Vrot | Dir \otimes Prox \otimes \Delta \otimes \pi_{reactive}) \\ \equiv \; & G(\mu(Dir, Prox), \sigma(Dir, Prox)). \end{aligned}$$

We have no a priori knowledge about the direction and distance of the obstacles, therefore we assign uniform distributions to $P(Dir|\Delta \otimes \pi_{reactive})$ and $P(Prox|\Delta \otimes \pi_{reactive})$. On the other hand, we assume that, for each sensory situation, there is one rotation speed that should be prefered. Hence, the distribution $P(Vrot|Dir \otimes Prox \otimes \Delta \otimes \pi_{reactive})$ has to be unimodal. However, the confidence in this choice may vary with the situation; this leads to assigning gaussian parametrical forms to this term.

This completes the specification phase.

In the identification phase, the programmer has to assess the values of the free parameters. In simple cases, the programmer may do it himself, by writing a function or table that stores these parameters. We obtained obstacle avoidance programs for our Khepera this way; we call this method *a priori programming*. However, it is often easier to justify parameters when they have been computed by a learning algorithm. In our example, since we only have mean values and standard deviations to set, this learning phase is simple. Using a joystick, we pilot the Khepera to follow contours. Every tenth of a second, we record experimental data $< dir, prox, vrot >$, where dir and $prox$ are computed from the proximeter values at time t, and $vrot$ is the motor command given by the user at the same time t. Given a set Δ_{follow} of such data, computing the mean values and standard deviations of the gaussian distributions associated with the $P(Vrot|Dir \otimes Prox \otimes \Delta_{follow} \otimes \pi_{reactive})$ term is straightforward.

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

$$Draw(P(Vrot|[Dir = d_t] \otimes [Prox = p_t] \otimes \Delta_{follow} \otimes \pi_{reactive})). \quad (2)$$

We observed as a result a very robust contour following behaviour, with only a few (20-30) seconds of learning.

The interested reader should refer to (Bessière et al., 1998a, Bessière et al., 1998b) for a more detailed presentation of our programming method.

5. Reactive behaviour learning

In this section, we describe two reactive programming experiments we did with the Khepera. The first one shows variations on the contour following program we presented earlier. The second one presents the facility our method provides for programming a light following behaviour for the Khepera. We end this section by analyzing the performance of a behaviour obtained by learning.

5.1 Proximity based behaviours

We presented in the previous section a contour following program which, in our system, is defined by the pair $< \pi_{reactive}, \Delta_{follow} >$ and a question (see Equation 2). Several variations are possible :

- It is possible to change the question, keeping the description unchanged. For example, $Draw(P(Vrot|[Dir = d_t] \otimes \Delta_{follow} \otimes \pi_{reactive}))$ allows the robot to operate even if some failure prevents the computation of $Prox$.

- It is possible to change the experimental data, keeping the preliminary knowledge $\pi_{reactive}$. It should be clear at this point that $\pi_{reactive}$ is rather generic, stating only a few modelization choices. What actually makes the robot follow contours is the identification phase, where the user shows what to do in some sensory situations. We can thus create completely different behaviours with the same specification $\pi_{reactive}$. For example, we programmed the robot to push or avoid obstacles, by merely changing the learning phase, and pairing $\pi_{reactive}$ with the corresponding data sequence, respectively Δ_{push} and Δ_{avoid}.

5.2 *Luminosity based behaviours*

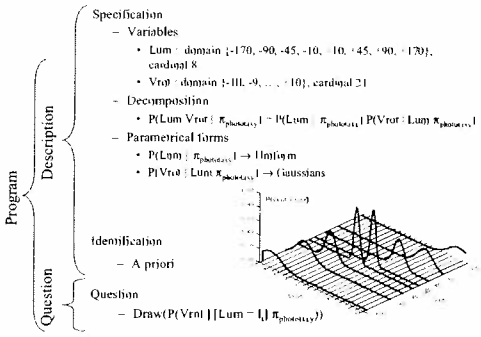

Figure 3: A straightforward program where the robot follows light. The plot shown represents the probability distributions $P(Vrot|Lum \otimes \pi_{phototaxy})$, one for each value of Lum, that were defined a priori.

It is also possible to change the specification. We present Figure 3 a program based on luminosity information, where the robot follows the (most powerful) light source. This phototaxy program necessitates few comments :

- The variables of interest are only Lum and $Vrot$, Lum being obtained by a simple computation on the luminosity variables the Khepera provides, and roughly corresponding to the direction the most powerful source of light is coming from.

- Here again, the term $P(Vrot|Lum \otimes \pi_{phototaxy})$ is associated with a set of gaussian distributions. However, for this program, we decided it would be easier to set the few parameter values a priori.

- The result is also satisfactory : the robot quickly orientates itself toward the light source, and follows it if it is moving in the environment.

More experiments are to be found in (Bessière et al., 1994, Lebeltel, 1999, Lebeltel et al., 2000) : the first presents a very simple experiment, with one degree of freedom and one sensory variable, while the second and third explore the power of our programming method, with scores of experiments on the Khepera.

5.3 *Quantitative results on learning*

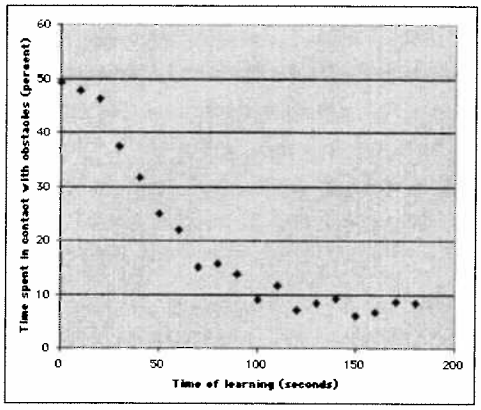

Figure 4: Performance of the obstacle avoidance behaviour as it is learned.

The goal of this section is to show some numerical results on our learning method. We chose to conduct experiments concerning the learning of the obstacle avoidance behaviour. The experimental setup was as follows.

We put the Khepera in a very crowded environment : roughly 10% of the environment area was occupied by lego bricks, that were evenly scattered. The result is an environment with very few and narrow (with respect to the Khepera diameter) paths to move around. The task was made even more challenging to the robot by adding legos of dark colours[3].

The Khepera was then taught how to avoid obstacles for three minutes. We then reset the knowledge of the robot to its initial ignorance and play back the learning sequence, evaluating the performance of the behaviour learned so far every ten seconds. The performance of the Khepera was taken by a human watcher, who would measure the time spent touching obstacles, on runs of 1 minute. The results, averaged over five experiments, are shown Figure 4.

In Figure 4, we plotted the average time spent in collision with obstacles, versus the time of learning. We

[3]Because such colours absorb the infrared lights, the proximeters do not sense them easily.

clearly see how the performance increases as the number of data given to the robot increases. In the first 30 seconds or so, the time spent in collision decreases slowly, because we usually teach the robot at the beginning to go straight when it does not see any obstacle. This obviously is not very useful for avoiding obstacles, since the behaviour stays random when anything is sensed, but it gives a more stable behaviour, which is also more pleasant to the observer. After about a hundred seconds of learning, we see that the performance has reached a very good mark, resulting in a behaviour that is very satisfying to the observer (the robot mainly brushes dark obstacles when trying to avoid them late). Finally, we see that the performance increases more slowly after 100 seconds, corresponding to limits inherent of the environment. This hypothesis is supported by reporting much better performances when decreasing the number of obstacles, organizing them in straight walls (this results in more open spaces), or even when the translation speed is reduced.

6. Learning how to combine behaviours

With the programming method sketched in the previous sections, we have implemented a "nightwatchman" task, where the Khepera has to :

- patrol its environment and alert whenever movement is detected,
- alert in case of fire and extinguish it (the fire is simulated by a lit candle, and we added a mini-fan on top of the Khepera for this task),
- recognize objects, if requested,
- go back to its base when appropriate,
- manage its energy level, by going back to its base to recharge batteries.

The program written to achieve this goal was obtained very easily by putting together 10 simpler programs, organized in 4 hierarchical layers, and which concern 42 variables. More details on how well our approach scales to complex behaviours are contained in the technical description of the "nightwatchman" program, which can be found in (Lebeltel, 1999, Lebeltel et al., 2000).

In the environment designed for the "nightwatchman" task, we usually put in a corner a base for the Khepera. It consists of a recess in the environment, with a light over it. Thus, going back to the base is simply made by combining a light following and an obstacle avoidance behaviour. In this section, we first present a program that implements such a combination; we then show how we recently managed to *learn* by experimentation how these behaviours were to be combined.

6.1 Combining obstacle avoidance with light following

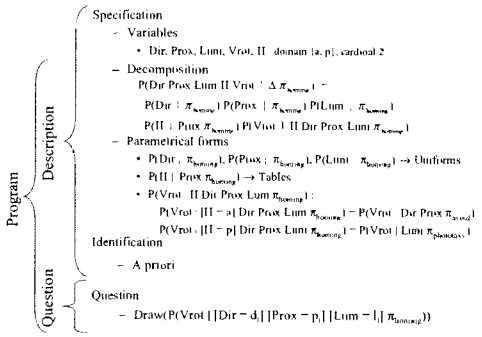

Figure 5: A homing program where the robot follows light while avoiding obstacles on the way.

Figure 5 shows a program implementing a combination of programs obtained from specifications π_{avoid} (the a priori version of our obstacle avoidance behaviour) and $\pi_{phototaxy}$ (the light following behaviour). Some comments are necessary about it.

- Variables : we keep all variables appearing in specifications π_{avoid} and $\pi_{phototaxy}$, and add a new variable H. It can take two values, a for *avoid* and p for *phototaxy*.
- Decomposition of the joint distribution : the term $P(H|Prox \otimes \pi_{homing})$ results from the independence hypothesis between H on the one hand, and Dir and Lum on the other hand. It means that we think the proximity of obstacles is a good enough criterion when deciding whether to avoid an obstacle or to follow the light.
- Parametrical forms : since we do not have any knowledge about the different sensory situations, terms $P(Dir|\pi_{homing})$, $P(Prox|\pi_{homing})$ and $P(Lum|\pi_{homing})$ are set to uniforms. The term $P(H|Prox \otimes \pi_{homing})$ is defined a priori, by giving intuitive values : we define the probability of doing obstacle avoidance, $P([H = a]|Prox \otimes \pi_{homing})$, as varying like $Prox$. Therefore, when $Prox$ is near 0, no obstacle is seen, and the probability of doing obstacle avoidance is near 0 too. Conversely, high values for $Prox$ mean an imminent collision, thus a high probability of avoiding the obstacle. The last term of the decomposition, $P(Vrot|H \otimes Dir \otimes Prox \otimes Lum \otimes \pi_{homing})$, makes the link between specifications π_{homing}, π_{avoid} and $\pi_{phototaxy}$. We state here that, when $[H = a]$, the motor command $Vrot$ is chosen accordingly to the obstacle avoidance knowledge

π_{avoid}; otherwise, $[H = p]$, and $Vrot$ follows the light following knowledge $\pi_{phototaxy}$.

- Question : notice that in the probabilistic question $P(Vrot|[Dir = d_t][Prox = p_t][Lum = l_t]\pi_{homing})$, the set of unknown variables is $\{H\}$. The inference thus gives :

$$P(Vrot|[Dir = d_t][Prox = p_t][Lum = l_t]\pi_{homing}) = \left(\begin{array}{ll} & P([H = a]|[Prox = p_t]\pi_{homing}) \\ \times & P(Vrot|[Dir = d_t][Prox = p_t]\pi_{avoid}) \end{array} \right) + \left(\begin{array}{ll} & P([H = p]|[Prox = p_t]\pi_{homing}) \\ \times & P(Vrot|[Lum = l_t]\pi_{phototaxy} \end{array} \right)$$

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

- Result : with this program, we successfully combine obstacle avoidance with light following. However, as expected, the robot can be stuck in local maxima of the light gradient, should a U-shaped corridor be taken.

6.2 *Learning the combination term*

What we show in this section is how it is possible to include learning in the previous program. Given the two base behaviours, we want to learn how to combine them, which in our system corresponds to identify the $P(H|Prox \otimes \Delta \otimes \pi_{homing})$ term.

One straightforward solution is to give control to the user via the H variable. When piloting the Khepera, the user thus commands whether it should follow the light or avoid obstacles, resulting in gathering experimental data of the form $< dir, prox, lum, h, vrot >$. Identification of the $P(H|Prox \otimes \Delta \otimes \pi_{homing})$ term is then very easy.

For various reasons, we prefered to keep the same piloting method as in the previous experiments, that is to say, the user pilots the robot with the joystick. Therefore, the data gathered is of the form $< dir, prox, lum, vrot >$. Notice that the data contains no information about the variable H. However, it can be inferred using probability rules, by asking the question

$$P(H|d \otimes p \otimes l \otimes v \otimes \Delta \otimes \pi_{homing}),$$

where d, p, l, and v stand for propositions $[Dir = dir]$, $[Prox = prox]$, $[Lum = lum]$ and $[Vrot = vrot]$, respectively. For clarity purposes, we omit the $\otimes$ and the π_{homing} symbols in the following expressions :

$$P(H|DirProxLumVrot\Delta) = \frac{P(HDirProxLumVrot|\Delta)}{\sum_H P(HDirProxLumVrot|\Delta)}$$

$$= \frac{\left(\begin{array}{l} \frac{1}{k_{Dir}} \frac{1}{k_{Prox}} \frac{1}{k_{Lum}} P(H|Prox\Delta) \\ P(Vrot|HDirProxLum) \end{array} \right)}{\sum_H \left(\begin{array}{l} \frac{1}{k_{Dir}} \frac{1}{k_{Prox}} \frac{1}{k_{Lum}} P(H|Prox\Delta) \\ P(Vrot|HDirProxLum) \end{array} \right)}$$

$$= \frac{P(H|Prox\Delta)P(Vrot|HDirProxLum)}{\sum_H P(H|Prox\Delta)P(Vrot|HDirProxLum)}. \quad (3)$$

We see that Equation 3 needs $P(H|Prox\Delta)$, which is the term we want to identify. In order to simplify further this equation, we choose to consider each experimental datum as if it were the first seen. We thus set $P(H|Prox\Delta)$ to $P(H|Prox\Delta_0)$, which is the initial value of the term, as defined by the parametrical form and learning mechanism. In our case, the $P(H|Prox\Delta_0)$ term is identical to a uniform distribution; Equation 3 thus becomes

$$P(H|DirProxLumVrot\Delta) = \frac{\frac{1}{k_H} P(Vrot|HDirProxLum)}{\sum_H \frac{1}{k_H} P(Vrot|HDirProxLum)} = \frac{P(Vrot|HDirProxLum)}{\sum_H P(Vrot|HDirProxLum)}. \quad (4)$$

Equation 4 can be given an interesting intuitive interpretation. It means the probability for H, given the experimental data $< dir, prox, lum, vrot >$ is

$$P(H|dplv\Delta) = \frac{P(v|Hdpl)}{\sum_H P(v|Hdpl)}.$$

Replacing terms in Equation 6.2 by their definition given Figure 5, we obtain :

$$P([H = a]|dplv\Delta) = \frac{P([Vrot = vrot]|[Dir = dir][Prox = prox]\pi_{avoid})}{P([Vrot = vrot]|[Dir = dir][Prox = prox]\pi_{avoid}) + P([Vrot = vrot]|[Lum = lum]\pi_{phototaxy})},$$

$$P([H = p]|dplv\Delta) = \frac{P([Vrot = vrot]|[Lum = lum]\pi_{phototaxy})}{P([Vrot = vrot]|[Dir = dir][Prox = prox]\pi_{avoid}) + P([Vrot = vrot]|[Lum = lum]\pi_{phototaxy})}.$$

The probability distribution for H is thus a comparison of probabilities of the motor command given by the user, considering the sensory situation, in the context of programs obtained from specifications π_{avoid} and $\pi_{phototaxy}$. In a sense, it corresponds to context recognition : "in the current sensory situation, does the motor command $vrot$ looks more like obstacle avoidance or light following?".

We therefore obtain a probability distribution over H. We propagate the knowledge it contains by generating new experimental data. For example, if the inference gave a probability for $[H = a]$ of 0.82, given some experimental datum $< dir, prox, lum, vrot >$, we would generate 82 $< dir, prox, lum, a, vrot >$ data, and 18 $< dir, prox, lum, p, vrot >$ data. We can now easily learn the $P(H|Prox \otimes \Delta \otimes \pi_{homing})$ term.

We report very satisfactory results from this experiment. We have successfully learned how to combine

obstacle avoidance with light following to obtain a homing behaviour. Notice that our method is homogeneous with our system used for reactive behaviour programming. Moreover, we believe it is quite general, since we stated no hypothesis concerning the behaviours to be combined. One could actually imagine using our system to combine more than two programs. If we had a *library* of base components, we could program a new behaviour by combining them all using our method, then dropping the ones with lowest (overall) probabilities, until only a few pertinent behaviours are combined. This tool could provide great help for the designer of the robot.

A longer description of this experiment is in (Diard, 1999), which also addresses all technical difficulties not mentioned in this paper.

7. Conclusion

We have presented a robotic programming environment that allows easy programming and combining of behaviours. We have shown that it also nicely included experimental learning. In further experiments described in (Lebeltel, 1999, Lebeltel et al., 2000), we also have managed, with the same formalism, to do sensor fusion, inverse programming, temporal sequences, and a "night-watchman" task, which integrates all these various kinds of descriptions. Future work will aim at including experimental learning in all these different types of program, so that a whole hierarchy of behaviours could be learned, from reactive to high level ones. We have in thought to make a Koala robot, our next experimental platform, become a robotic pet, detecting and chasing intruders, showing visitors around, carrying or bringing back items from room to room, etc...

Acknowledgements. It is a great pleasure to thank Pierre Bessière and Emmanuel Mazer for their careful readings of this manuscript, this would not have been achieved without their help.

References

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

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

Bernhardt, R. and Albright, S. e. (1993). Robot calibration. *Chapman & Hall.*

Bessière, P., Dedieu, E., Lebeltel, O., Mazer, E., and Mekhnacha, K. (1998a). Interprétation ou description i : Proposition pour une théorie probabiliste des systèmes cognitifs sensori-moteurs. *Intellectica 1998/1-2, 26-27, pp. 257-311.*

Bessière, P., Dedieu, E., Lebeltel, O., Mazer, E., and Mekhnacha, K. (1998b). Interprétation ou description ii : Fondements mathématiques. *Intellectica 1998/1-2, 26-27, pp. 313-336.*

Bessière, P., Dedieu, E., and Mazer, E. (1994). Representing robot/environment interactions using probabilities: the "beam in the bin" experiment. *PerAc'94 (From Perception to Action); Lausanne, Switzerland.*

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

Brafman, R., Latombe, J.-C., Moses, Y., and 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. (1988). Bayesian spectrum analysis and parameter estimation. *Springer Verlag.*

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

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.*

Dekhil, M. and Henderson, T. (1998). Instrumented sensor system architecture. *International Journal for Robotics Research (IJRR) ; Vol. 17(4), pp. 402-417.*

Delcher, A., Grove, A., 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. (1999). Apprentissage hiérarchique bayesien. *Masters Degree Thesis, UJF Grenoble.*

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

Erickson, G. and Smith, C. (1988a). Maximum-entropy and bayesian methods in science and engineering ; volume 1 : Foundations. *Kluwer Academic Publishers.*

Erickson, G. and Smith, C. (1988b). Maximum-entropy and bayesian methods in science and engineering ; volume 2 : Applications. *Kluwer Academic Publishers.*

Frey, B. (1998). Graphical models for machine learning and digital communication. *MIT Press.*

Halpern, J. (1999a). A counterexample to theorems of cox and fine. *Journal of Artificial Intelligence Research (JAIR), Vol. 10, pp. 67-85.*

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

Jaakkola, T. and Jordan, M. (1999). Variational probabilistic inference and the qmr-dt network. *Journal of Artificial Intelligence Research (JAIR), Vol. 10, pp. 291-322.*

Jaynes, E. (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. (1998). Probability theory - the logic of science. *Unfinished book. Note : Publicly available at http://bayes.wustl.edu.*

Jaynes, E. (September 1982). On the rationale of maximum-entropy methods. *Proceedings of the IEEE, Vol. 70, No. 9, pp. 939-952.*

Jordan, M. (1998). Learning in graphical models. *MIT Press.*

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

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

Kapur, J. and Kesavan, H. (1992). Entropy optimization principles with applications. *Academic Press.*

Laplace, P. S. d. (1774). Mémoire sur la probabilités des causes par les évènements. *Mémoire de lacadémie royale des sciences; Reprinted in Oeuvres complètes de Laplace, (vol. 8), Gauthier Villars, Paris, France.*

Laplace, P. S. d. (1814). Essai philosophique sur les probabilités. *Courcier Imprimeur, Paris; Reprinted in Oeuvres complètes de Laplace, (vol. 7), Gauthier Villars, Paris, France.*

Lauritzen, S. and Spiegelhalter, D. (1988). Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *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, INP Grenoble.*

Lebeltel, O., Bessière, P., Diard, J., and Mazer, E. (2000). Bayesian robots programming. *Les cahiers du laboratoire Leibniz ; 01.*

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

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., and 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).*

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

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

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

Ruiz, A., Lopez-de Teruel, P., and Garrido, M. (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., Jaakkola, T., and Jordan, M. (1996). Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research (JAIR), Vol. 4, pp. 61-76.*

Schneider, S., Chen, V., Pardo-Castellote, G., and Wang, H. (1998). Controlshell: A software architecture for complex electromechanical systems. *International Journal for Robotics Research (IJRR) ; Vol. 17(4), pp. 360-380.*

Smith, C. and Grandy, W. J. (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. (1998a). Bayesian landmark learning for mobile robot localization. *Machine Learning, vol. 33(1), pp.41-76.*

Thrun, S. (1998b). Learning metric-topological maps maps for indoor mobile robot navigation. *Artificial Intelligence 99(1), 21-71.*

Zhang, N. and Poole, D. (1996). Exploiting causal independence in bayesian network inference. *Journal of Artificial Intelligence Research (JAIR), Vol. 5, pp. 301-328.*