
Bayesian Modeling and Reasoning for Real World Robotics: Basics and Examples

David Bellot¹, Roland Siegwart³, Pierre Bessière², Adriana Tapus³,
Christophe Coué², and Julien Diard²

¹ Department of Statistics, University of California, Berkeley, 94720-3860, USA

² Gravir/IMAG/CNRS INRIA Rhône-Alpes ZIRST - 655 avenue de l'Europe
38330 Montbonnot Saint Martin FRANCE

³ Swiss Federal Institute of Technology EPFL-STI-I2S-LSA1 CH-1015 Lausanne,
Switzerland

Summary. Cognition and Reasoning with uncertain and partial knowledge is probably the biggest challenge for autonomous mobile robotics. Previous robotics systems based on a purely logical or geometrical paradigm are limited in their ability to deal with partial or uncertain knowledge, adaptation to new environments and noisy sensors. Representing knowledge as a joint probability distribution increases the possibility for robotics systems to increase their quality of perception on their environment and helps them to take the right actions towards a more realistic and robust behavior. Dealing with uncertainty is thus a major challenge for robotics in a real and unconstrained environment. Here, we propose a new formalism and methodology called Bayesian Programming which aims at the design of efficient robotics systems evolving in a real and uncontrolled environment. This original formalism will be exemplified by two interesting experiments where robots are driven by a Bayesian Program (BP). These examples represent situations where the robot can sense only a small part of its global environment using noisy sensors. The second fact about these environments is they cannot be constrained so that to ease the control of the robot.

1 Incompleteness and Uncertainty in Robotics

One of the biggest challenge for autonomous mobile robotics is for a system to drive the robot in an unknown or partially known environment, using noisy sensors and where unexpected events happen. Even if recent research resulted in some very nice demonstrations of autonomous navigation in dynamic environments, we are still far from having concepts and algorithms that adapt to different environments and that scale well with the complexity of the environment.

This paper suggests a generic approach based on the well-known Bayes theory, in order to progress toward cognitive systems that are able to reason in

highly complex real-world environments. The proposed Bayesian framework is a generic approach for probabilistic reasoning. It combines probability distributions, established through *a priori* knowledge and learning, with Bayesian inference in order to make autonomous system capable of dealing with the uncertainty and incompleteness of the real world. *A priori* knowledge and models reduce significantly the complexity of the implementation. Thus, the probabilistic reasoning becomes more feasible for highly dynamic and complex environments.

In classical robotics [1], the programmer of the robot has himself an abstract conception of the environment, described in geometrical, analytical and/or symbolic terms because the shape of objects, the map of the world, the laws of physics and the objects are known. Programming such a robot is a difficult task because the programmer needs to completely know the environment. The main example of this kind of robotics are the robots used to manufacture cars. Their environment is highly constrained and their behavior is usually described through a finite-state automaton. This is the usual answer to the problem of uncertainty: let the environment be as predictable as possible by controlling and constraining it. But if the environment is open and if it cannot be constrained, or if the programmer aims at a more versatile robot, then the complexity of the program increases dramatically and lead to intractable models and representation of the real world. Therefore, it is necessary just to take into account a small part of the environment leading to a large number of hidden or unknown variables. The model is incomplete.

From an engineering point of view, an accurate control of both the environment and the tasks ensures that industrial robots work properly. However, this approach is no longer possible when the robot must act in an environment not specifically designed for it. The purpose of this chapter is to give an overview of a generic solution to this problem especially to present a versatile framework called Bayesian Programming (BP). Section 2 presents the Bayesian Programming paradigm. It establishes a common formalism and methodology that will be used throughout this chapter. The last section will be devoted to two complex examples in robotics and a solution based on Bayesian Programming will be presented.

2 The Bayesian Programming Framework: A Generic Formalism

2.1 Description

The Bayesian Programming (BP) formalism allows for a *unique notation and structure* to describe probabilistic knowledge and its use [2]. A Bayesian Program is a structure made of two components as shown in Figure 1.

The first is a *declarative* component, where the user defines a *description*: it is a way to specify a joint distribution over a set of variables $\{X_1 X_2 \dots X_n\}$,

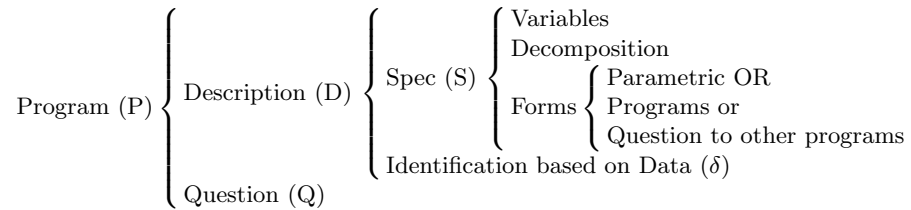


Fig. 1. Structure of a Bayesian program.

given a set of experimental data δ and preliminary knowledge π . The variables have to be relevant for the environment one would like to model. The joint distribution $P(X_1 X_2 \dots X_n | \delta \pi)$ is decomposed into a product of simpler terms based on some conditional independence assumptions. This set of assumptions belongs to the set π of *a priori* knowledge. In order to complete the description, parametric forms (also belonging to π) and *a priori* distributions are given, that is the numerical parameters of the so-called parametric forms. If there are free parameters in the parametric forms, they have to be manually defined or fitted using a learning procedure on the set of experimental data δ .

The aim of a decomposition is to introduce some conditional independence assumptions between variables so that to decrease the complexity of the inference process or more generally to introduce *a priori* knowledge about the environment or the behavior of the robot. This kind of knowledge is provided by the programmer and represents either causal interactions [3] or structural relations between variables. For example, a first-order Markov assumption claims that the belief state of a variable X_t at time t is independent of its long-term past given its short-term past. In other words X_t is independent of $X_{t-i}, \forall i > 1$ given X_{t-1} . Therefore the decomposition for such a simple system is $P(X_t|X_{t-1}).P(X_{t-1})$.

Variables represent facts about the environment or the robot. For example, a light sensor could be represented by a variable L where its probability distribution is assumed to be Gaussian, $L \sim \mathcal{N}(\mu, \sigma^2)$, and represent the intensity of light occurring at the sensor.

2.2 Question

Now, let assume that an environment can be described with the following set of variables $\mathcal{S} = \{A, B, C, D\}$. Our *a priori* (or prior) knowledge can be summarized by the statement "*C is independent of D given A and B*". But we have no other particular knowledge about A and B . Therefore, an obvious decomposition would be $P(ABCD) = P(C|AB)P(D|AB)P(AB)$. This decomposition is not easy to use since we have to compute the joint probability distribution over $\{AB\}$. $P(AB)$ can be approximated using sampling techniques or decomposed into a simpler joint probability distribution using the so-called chain's rule: $P(AB) = P(A|B)P(B) = P(B|A)P(A)$.

Next to the decomposition is the question. A Bayesian Program aims at representing both the knowledge and its use one wishes to do with it. Using knowledge is answering the question. Answering the question is solving a Bayesian inference problem on the description in order to compute the posterior probability distribution described by the question. Therefore, a question in a Bayesian Program is the posterior probability distribution one is interested given some measurements on the other variables. For example, I know some facts about B , but nothing about the other variables, say $B = b_1$. Then I would like to know the posterior distribution of D given $B = b_1$. The question is $P(D|B = b_1)$. Here we assume we have an algorithm to solve this Bayesian inference problem, where, given the description, π and δ , we can compute the probability distribution of $P(D|B = b_1)$.

The general question $P(\mathcal{S}_{\setminus B}|B = b_1)$ is also known as the belief propagation problem [4]. As this chapter is mainly concerned with modeling issues, we assume the inference problem to be solved and implemented in an efficient way by an inference engine. But the reader should be warned that Bayesian inference is not an obvious problem and inference algorithms are usually designed together with the model itself so that to obtain optimal results in terms of computational costs and accuracy. However, general algorithms are also available, based on messages and beliefs propagation [5], sampling techniques or variational approximations [6].

3 Complex Bayesian Programming for Robotics

This section presents two applications of Bayesian Programming. The first one is an extension of occupancy grids using *a priori* knowledge to perform target position and velocity in an urban traffic situation. The grids are combined with danger estimation to perform an elementary task of obstacle avoidance with an electric car. The second application is devoted to topological global localization by using sequences of features forming a global distinctive fingerprint. The topological representation gives a compact representation since only distinctive places within the environment are encoded.

3.1 Bayesian Programming for Multi-Target Tracking: An Automotive Application

The ADAS context

Unlike regular cruise control systems, Adaptive Cruise Control (ACC) systems use a range sensor to regulate the speed of the car while ensuring collision avoidance with the vehicle in front. ACC systems were introduced on the automotive market in 1999. Since then, surveys and experimental assessments have demonstrated the interest for this kind of systems. They are the first step towards the design of future Advanced Driver Assistance Systems

(ADAS) that should help the driver in increasingly complex driving tasks. The use of today commercially available ACC systems is pretty much limited to motorways or urban expressways without crossings. The traffic situations encountered are rather simple and attention can be focused on a few, well defined detected objects (cars and trucks). Nonetheless, even in these relatively simple situations, these systems show a number of limitations: they are not very good at handling fixed obstacles and may generate false alarms; moreover, in some 'cut-in' situations, *i.e.* when the intrusion of an other vehicle or a pedestrian in the detection beam is too close to the vehicle, they may be unable to react appropriately.

A wider use of such systems requires to extend their range of operation to some more complex situations in dense traffic environments, around or inside urban areas. In such areas, traffic is characterized by lower speeds, tight curves, traffic signs, crossings and "fragile" traffic participants such as motorbikes, bicycles or pedestrians.

The related Multi-Target Tracking problem

A prerequisite to a reliable ADAS in such complex traffic situations is an estimation of dynamic characteristics of the traffic participants, such as position and velocity. This problem is basically a *Multi-target Tracking* problem. The objective is to collect *observations*, *i.e.* data from the sensor, on one or more *potential obstacles* in the environment of the vehicle, and then to estimate at each time step and as robustly as possible the obstacles position and velocity. Classical approach is to track the different objects independently, by maintaining a list of *tracks*, *i.e.* a list of currently known objects. The main difficulty of multi-target tracking is known as the *Data Association* problem. It includes observation-to-track association and track management problems. The goal of observation-to-track association is to decide whether a new sensor observation corresponds to an existing track or not. Then the goal of track maintenance is to decide the confirmation or the deletion of each existing track, and, if required, the creation of new tracks. A complete review of the tracking methods with one or more sensors can be found in [7].

Urban traffic scenarios are still a challenge in multi-target tracking area: the traditional data association problem is intractable in situations involving numerous appearances, disappearances and occlusions of a large number of rapidly maneuvering targets.

The approach presented here is a new approach for a robust perception and analysis of highly dynamic environments. This approach has been designed in order to avoid the data association problem previously mentioned. It is based on a probabilistic *grid representation of the obstacles state space*. As we consider the position and the velocity of the potential obstacles with respect to our vehicle, this grid is 4-dimensional. Then for each cell of the grid, the occupancy probability is estimated using sensor observations and some prior knowledge.

Estimation of the Occupancy Grid

The objective is to compute from the sensor observations the probability that each cell is full or empty. To avoid a combinatorial explosion of grid configuration, the cell states are estimated as *independent* random variables.

The occupancy grid framework was extensively used for mapping and localization. Of course, for an automotive application, it is impossible and useless to model the whole environment of the vehicle with a grid. Thus we will model only the near-front environment of our vehicle. As we want to estimate the relative position and the relative velocity of objects, each cell of our 4-D⁴ grid corresponds to a position and a speed relative to the vehicle.

Figure 2 presents the Bayesian Program for the estimation of the occupancy probability of a cell. To simplify notations, a particular cell of the grid is denoted by a single variable X , despite the grid is 4-D. The number of sensor observations at time k is named N^k . One sensor data at time k is denoted by the variable Z_i^k , $i = 1 \dots N_k$. The set of all sensor observations at time k is noted Z^k . The set of all sensor observations until time k is referred by the notation $Z_{1:k}$. A variable called the *matching* variable and noted M^k is added. Its goal is to specify which observation of the sensor is currently used to estimate the state of the cell.

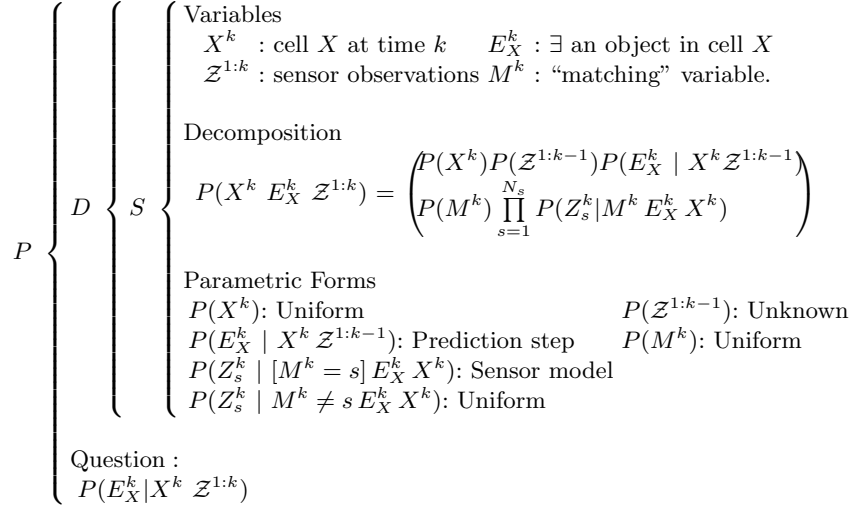


Fig. 2. Estimation Step at time k .

⁴ 2 dimensions for the x, y position and 2 dimensions for the \dot{x}, \dot{y} velocities

Bayesian Occupancy Filter

To take into account the dynamic environment, and to be as robust as possible relatively to objects occlusions, it is necessary to take into account the *sensor observations history and the temporal consistency of the scene*. This is done by introducing a two-step mechanism in the occupancy grid estimation. This mechanism includes a prediction (history) and an estimation (new measurements) steps. This mechanism is derived from the *Bayes filters* approach [8] and it is called the *Bayesian Occupancy Filter* (BOF).

The corresponding Bayesian Program is presented by Figure 3.

$$P \left\{ \begin{array}{l} D \left\{ \begin{array}{l} S \left\{ \begin{array}{l} \text{Variables} \\ X^k, E_X^k, X^{k-1}, E_X^{k-1}, \mathcal{Z}^{1:k-1} : \text{same semantic as previously} \\ U^{k-1} : \text{control of the Cycab at time } k-1 \\ \\ \text{Decomposition} \\ P(X^k E_X^k X^{k-1} E_X^{k-1} \mathcal{Z}^{1:k-1} U^{k-1}) = \\ \left(\begin{array}{l} P(\mathcal{Z}^{1:k-1})P(U^{k-1})P(X^{k-1}) P(E_X^{k-1} | X^{k-1} \mathcal{Z}^{1:k-1}) \\ P(X^k | X^{k-1} U^{k-1}) P(E_X^k | E_X^{k-1} X^k X^{k-1}) \end{array} \right) \\ \\ \text{Parametric Forms} \\ P(\mathcal{Z}^{1:k-1}) : \text{Unknown} \\ P(U^{k-1}) : \text{Uniform} \\ P(X^{k-1}) : \text{Uniform} \\ P(X^k | X^{k-1}) : \text{dyn. model} \\ P(E_X^k | E_X^{k-1} X^k X^{k-1}) : \text{Dirac.} \\ P(E_X^{k-1} | X^{k-1} \mathcal{Z}^{1:k-1}) : \text{estim. at } k-1 \\ \\ \text{Question :} \\ P(E_X^k | X^k \mathcal{Z}^{1:k-1} U^{k-1}) \end{array} \right. \end{array} \right. \end{array} \right.$$

Fig. 3. Prediction Step at time k .

Experimental Results

To test the estimation of occupancy grids both a simulator and the real Cycab vehicle were used. Figure 4 shows the first results of estimation and prediction steps, for a static scene. The upper left scheme depicts the situation: two static objects are present in front of the Cycab. These two objects are fixed. The Cycab is static too. Thus only 2-dimensional grids are depicted, corresponding to the object's position at a null speed. Figure 4b represents the occupancy grid, knowing only the first sensor observations. The gray level corresponds to the probability that a cell is occupied. In this case, the two objects are detected

by the sensor. Consequently, two areas with high occupancy probabilities are visible (dark gray areas). These probability values depend on the probability of detection, the probability of false alarm, and on the sensor precision. All these characteristics of the sensor are taken into account in the sensor model. The cells hidden by a sensor observation have not been observed. Thus we can not conclude about their occupancy. That explains the two areas of probability values close to 0.5. Thanks to this property of occupancy grids and to the prediction phase, the estimation of the grid is robust to temporary occlusion between moving objects. Finally, for cells located far from any sensor observation, the occupancy probability is low (plain grey areas).

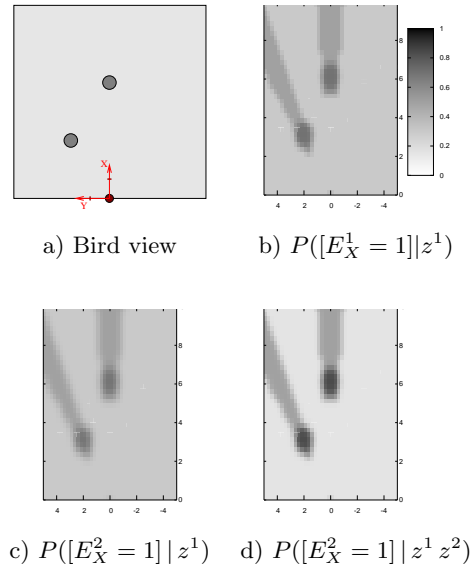


Fig. 4. First example of grid estimation, for a static scene.

To validate the approach in dynamic situations, an application involving an electric car has been implemented [9]. The car is longitudinally controlled in order to avoid obstacles. This basic behavior is obtained by combining the occupancy probability and the danger probability of each cell of the grid. Results of the experiments clearly show that this approach is able to prevent collisions even when moving obstacles (pedestrians for example) are temporally hidden (by a parked car for example).

3.2 Bayesian Programming for Topological Navigation with the Fingerprint Concept

Introduction

The topological approach yields a compact representation and allows high-level symbolic reasoning for map building and navigation. With this method we try to eliminate the perceptual aliasing (*i.e.* distinct locations within the environment appearing identical to the robot's sensors) and to improve the distinctiveness of the places in the environment. To maximize the reliability in navigation, the information from all the sensors that are available to the robot must be used. The notion of fingerprint is used [10, 11] to characterize the environment is especially interesting when used within a topological localization and multiple modality framework. A fingerprint is a circular list of features, where the ordering matches the relative ordering of the features around the robot. We denote the fingerprint sequence using a list of characters, where each character represents the instance of a specific feature type. In our case we choose to extract color patches and vertical edges from visual information and corners and beacons from laser scanner. The letter 'v' is used to characterize an edge, the letters A, B, C, . . . , P to represent hue bins, the letter 'c' to characterize a corner feature and the letter 'b' to characterize a beacon feature [11, 12, 13].

Fingerprint Generation

The fingerprint generation is done in three steps (see Figure 5). The extraction of the different features (*e.g.* vertical edges, corners, color patches, beacons) from the sensors is the first phase of the fingerprint generation. The order of the features, given by their angular positions ($0^\circ..359^\circ$) is kept in an array. At this stage a new type of virtual feature 'f' is introduced, that reflects the correspondence between a corner and an edge. The ordering of the features in a fingerprint sequence is highly informative and for that reason the notion of angular distance between two consecutive features is added. This geometric information increases, once again, the distinctiveness between the fingerprints. Therefore, we introduced an additional type of feature, the empty space feature 'n', to reflect angular distance. Each 'n' covers the same angle of the scene (20 degrees). This insertion is the last step of the fingerprint generation [11].

Fingerprint Matching for Localization

The string-matching problem is not an easy one. Usually strings do not match exactly because the robot may not be exactly located on a map point and/or some changes in the environment or perception errors occurred. The standard algorithms are quite sensitive to insertion and deletion errors, which cause

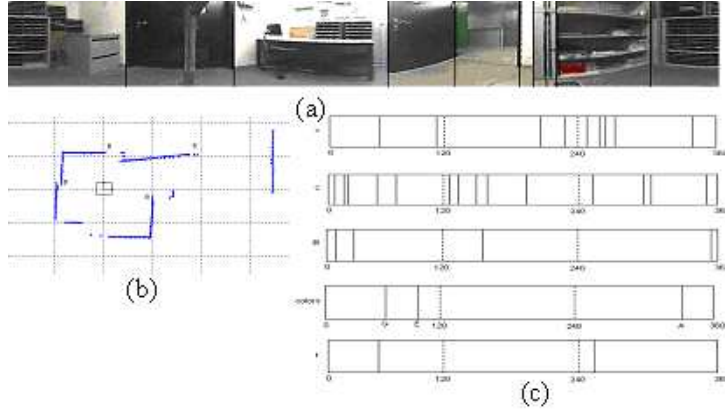


Fig. 5. Fingerprint generation. (a) panoramic image with the vertical edges 'v' and color patches detection, (b) laser scan with extracted corners 'c' and beacons 'b', (c) the first four images depict the position (0° to 359°) of the vertical edges, the corners, the beacons and the colors (G-green, E-light green, and A-red) respectively. The fifth image describes the correspondence between the vertical edge features and the corner features. By regrouping all this results together and by adding the empty space features, the final fingerprint is: *cbccbnfGcnEnvccncbcvncnnfvvvnccAcb*.

the string lengths to vary significantly. The approach adopted previously in the fingerprint approach for sequence matching is inspired by the minimum energy algorithm used in stereo-vision for finding pixels in two images that correspond to the same point of a scene [14, 10, 11]. Our current approach is a combination of the global alignment algorithm and the Bayesian formalism and it is described below.

Probabilistic fingerprint matching algorithm

The new approach comprises two steps. The first step is the phase of supervised learning where the robot inspects several locations, denoted by Loc . From each location $loc \in Loc$ the robot extracts the fingerprint data [11] and stores it along with the name of the location in a database, denoted by the symbol π . The second step is the phase of application, where we want the robot to localize itself in the environment. To answer at the question "Where am I?", the robot will extract the fingerprint fp of its current surroundings and solve the basic formula of probabilistic localization:

$$loc^* = \arg \max_{loc \in Loc} P(loc | fp \pi).$$

This means that if fingerprints are associated to each location, then the actual location of the robot may be recovered by comparing the fingerprint fp with the data of known locations and choosing the location loc^* which has the

highest probability. In what follows we show how $P(loc | fp \pi)$ can be solved by applying Bayesian Programming.

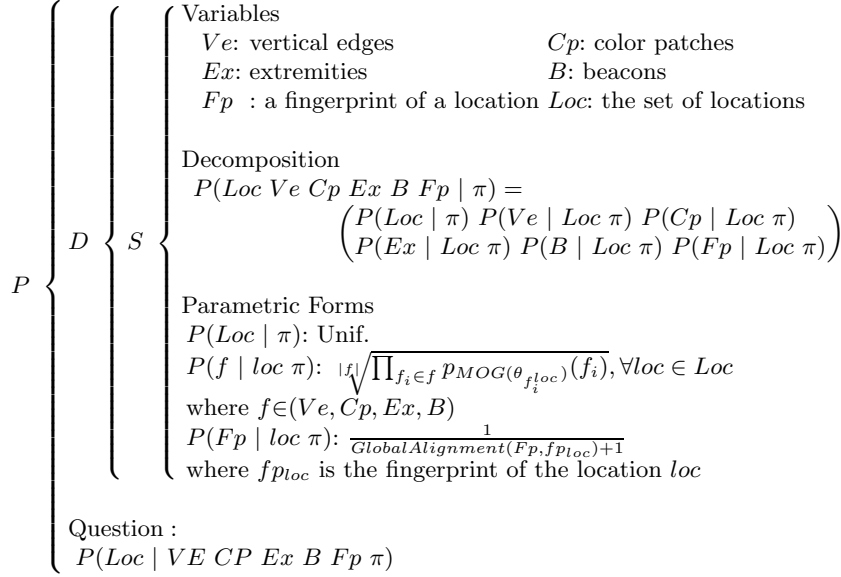


Fig. 6. The fingerprint matching formalism written in BP

Figure 6 shows the Bayesian Program used for the fingerprint matching. The features are denoted by: Ve the set of vertical edges and Cp the set of color patches extracted by the omni-directional camera, Ex the set of line extremities and B the set of beacons extracted from the data given by the laser scanner. For the fingerprint of a location, which is encoded as a circular string the notation Fp is used, and for the set of known (learned) locations the notation Loc is employed. We assume that the variables Ve , Cp , Ex , B and Fp are independent from one another. We consider that the features (Ve , Cp , Ex , B) are dependent of the location and these dependencies lead to the decomposition described in the Bayesian Program (see Figure 6). Since we have no a priori information about locations, we consider each location to be equally probable and consequently we express the probability of a location given all the prior knowledge as a uniform distribution. To determine the probability of one feature f , where $f \in Ve, Cp, Ex, B$, given the location and all the a priori knowledge, we suggest to express this probability as the likelihood of the new feature data f with respect to the distribution of the same feature as encountered at the given location during the learning phase. We calculate the distribution as a mixture of Gaussians (MOG) in angle space, optimizing the mixture parameters $\theta_{f_i^{loc}} = \{w_{f_i^{loc}}, \mu_{f_i^{loc}}, \sigma_{f_i^{loc}}\}$ (where $w_{f_i^{loc}}$ is

the weight, $\mu_{f_i^{loc}}$ is the mean and $\sigma_{f_i^{loc}}$ is the standard deviation of the f_i -th mixture component), by making use of the Expectation Maximization (EM) algorithm [15]. We illustrate the distribution of a MOG with an example. In Figure 7 the set of 13 occurrences of some angular feature and the MOG calculated for it is depicted. The parameters θ of the MOG were chosen to maximize this set of data. It can be seen on the Figure 7 that a MOG is a probability density function. To calculate the probability of the fingerprint sequence given the location and all the prior knowledge: we will use the global alignment algorithm [16] used usually for the alignment of DNA sequences and so let $GlobalAlignment(Fp, fp_{loc})$ be a function yielding the minimal cost of the global alignment algorithm of two fingerprint strings. The three equations from the parametric forms will solve the basic question described in the Bayesian Program.

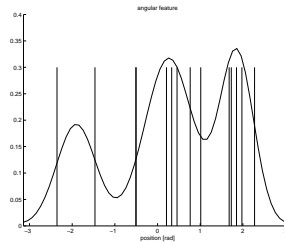


Fig. 7. Example of a MOG.

Experimental Results

The approach has been tested in ten rooms, in a $50 \times 25 m^2$ portion of our institute building. For the experiments, Donald Duck (see Figure 8), a fully autonomous mobile robot, has been used.

In order to validate the probabilistic fingerprint approach, for each of the ten rooms, fingerprints have been extracted. This experiment has been repeated six times for each room, with a variation of the robot locations within the room. Four fingerprints per room have been included in a database. The others 20 fingerprints (2 per room) have been matched to the database for testing the localization algorithm. For a given observation (fingerprint), a match is successful if the best match with the database (highest probability) corresponds to the correct room. Since the number of occurrences of the beacon and color patch feature was too small to give significant results, they were omitted for the MOG calculations, but they were used for the fingerprint strings. The results yield a percentage of successful matches of 82.4%. The method presented does not always lead to a perfect success rate, but it



Fig. 8. System used for the experimentation: The fully autonomous robot Donald Duck and the panoramic vision system. The camera has a 640×480 pixel resolution and an equiangular mirror is used so that each pixel in the image covers the same view angle.

still delivers valuable information for false-matched rooms. When the room is successfully matched, the probabilistic matching algorithm gives a high probability: 0.79 in average (between 0.62 and 0.89). Even if it detects the correct room with the second or third highest probability, a Bayesian localization approach, like for example a Partially Observable Markov Decision Process (POMDP) [17, 18] can use this information in its observation function. An amelioration of the results can be expected with the augmentation of the number of components of Mixture of Gaussians (MOG) and of the number of observations of a feature [19].

4 Conclusion and open problems on Bayesian Programming

The main interest of Bayesian Programming is its ability to describe real-world models with partial and incomplete knowledge about the world. Bayesian Programming is a promising framework and a lot of exciting open problems still exists. To progress toward more robust and sophisticated robotics control systems, these problems need innovative and original solutions. Apart from robotics, those problems are common to other artificial intelligence related fields. It was shown before that it is impossible to completely represent an environment and the strength of Bayesian Programming is to deal with this incompleteness by transforming it into uncertainty. However, the more knowledge is used, the more accurate is the behavior of the robot. Therefore, the problem of making realistic and robust behaviors can be summarized as follow:

- how to make a well-adapted Bayesian Program?
- how to know that a program fit perfectly into a particular task?
- how to learn unknown parameters from real data and experiences?
- how to efficiently use a complex program with many variables and many probabilistic forms?

The answer to those questions is not obvious and leads to more general and exciting questions : learning and inference. How to learn a Bayesian Program instead of making it by hand and how to use the data provided by sensors in order to extract and learn a program? It is out of the scope of this paper to present details about state-of-the-art research on algorithm for Bayesian Programming, but we give here a few facts on this:

- inference is a NP-hard problem for a general Bayesian Program, but solutions exists for particular problems. For example, a state-space model Bayesian filter is usually dealt with using Kalman filter or the well-known Forward-Backward algorithm. If the time series analysed by the filter is stationary gaussian, then Durbin-Levison approaches are technically efficient [20],
- inference on regular lattices of variables can be solved using suited algorithms. For instance, factorial hidden Markov models represent a complex stochastic process decomposed into several independent Markov chains given observations. The inference problem is intractable but the use of a variational approximation helps to overcome the computational cost of exact inference [21],
- probabilistic forms are usually discrete or gaussian. However, Bayesian Programming aims at representing whatever probability distributions where probabilistic forms are numerous or even unknown. Numerous approaches exists for dealing with other probabilistic forms, like Mixtures of Gaussians or exponential forms [22],
- complexity of probabilistic forms is sometime a bottleneck for robotics applications. Some techniques aims at reducing the memory footprint of those forms by approximating the distribution leading to a more efficient internal representation [23],
- making versatile programs is hard, but making small programs is quite easier. Does it exist a similar way as object software engineering to link and join small Bayesian Programs into a larger one. Several approaches have been developed: relational probabilistic models [24] or active learning [25] in the context of expert systems.

These techniques and approaches have been designed for particular purposes in the field of statistical learning and artificial intelligence and solve specific problems. They can be adapted to robotics and lead will to more efficient robots systems being able to deal with more complex environments as those of the real world.

References

1. T. Lozano-Pérez, J. Jones, E. Mazer, and P. O'Donnell, *HANDEY, A Robot Task Planner*. Cambridge, Massachussets, USA: The MIT Press, 1992. ISBN 0-262-12172-7.

2. O. Lebeltel, P. Bessière, J. Diard, and E. Mazer, “Bayesian robots programming,” *Autonomous Robot*, vol. 16, 1 2004.
3. J. Pearl, *Causality - Models, reasoning and inference*. Cambridge University Press, 2001.
4. I. Rish, *Efficient Reasoning in Graphical Models*. PhD thesis, University of California, Irvine, 1999.
5. M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul, “An introduction to variational methods for graphical models,” *Machine Learning*, vol. 37, no. 2, pp. 183–233, 1999.
6. D. J. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.
7. S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Artech House, 2000.
8. A. H. Jazwinsky, *Stochastic Processes and Filtering Theory*. New York : Academic Press, 1970.
9. C. Coué, C. Pradalier, and C. Laugier, “Bayesian programming for multi-target tracking: an automotive application,” in *Proceedings of the International Conference on Field and Service Robotics*, (Lake Yamanaka, Japan), 2003.
10. P. Lamon, I. Nourbakhsh, B. Jensen, and R. Siegwart, “Deriving and matching image fingerprint sequences for mobile robot localization,” in *Proceedings of the International Conference on Robotics and Automation*, vol. 2, (Seoul, Korea), pp. 1609–1614, May 2001.
11. P. Lamon, A. Tapus, E. Glauser, N. Tomatis, and R. Siegwart, “Environmental modeling with fingerprint sequences for topological global localization,” in *Proceedings of the International Conference on Intelligent Robots and Systems*, vol. 4, (Las Vegas, USA), pp. 3781–3786, October 2003.
12. A. Martinelli, A. Tapus, and R. Siegwart, “Multi-resolution slam for real world navigation,” in *11th International Symposium of Robotics Research*, (Siena, Italy), October 2003.
13. K. O. Arras and R. Siegwart, “Feature extraction and scene interpretation for map-based navigation and map building,” in *Proceedings of the Symposium on Intelligent Systems and Advanced Manufacturing*, (Pittsburgh, USA), October 1997.
14. T. Kanade and Y. Ohta, “Stereo by intra- and inter- scanline search dynamic programming,” *IEEE Transactions on pattern analysis and machine intelligence*, March 1985.
15. J. A. Bilmes, “A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models,” *ICSI-TR-97-021*, 1997.
16. S. Needleman and C. Wunsch, “A general method applicable to the search for similarities in the amino acid sequence of two proteins,” *Journal Molecular Biology*, vol. 48, 1970.
17. A. R. Cassandra, L. Kaelbling, and J. Kurien, “Acting under uncertainty: Discrete bayesian models for mobile robot navigation,” in *Proceedings of the International Conference on Robotics and Automation*, vol. 2, (Osaka, Japan), pp. 963 – 972, November 1996.
18. N. Tomatis, I. Nourbakhsh, and R. Siegwart, “Hybrid simultaneous localization and map building: a natural integration of topological and metric,” *Robotics and Autonomous Systems*, vol. 44, pp. 3–14, 2003.

19. A. Tapus, S. Heinzer, and R. Siegwart, “Bayesian programming for topological global localization with fingerprints,” in *International Conference on Robotics and Automation*, (New Orleans, USA), May 2004.
20. P. Brockwell and R. Davis, *Introduction to Time Series and Forecasting*. Springer, 2002.
21. Z. Ghahramani and M. Jordan, “Factorial hidden Markov models,” MIT Computational Cognitive Science Report Technical Report 9502, MIT, 1995.
22. M. Jordan, “Graphical models,” *Statistical Science (Special Issue on Bayesian Statistics)*, p. In press, 2002.
23. D. Bellot and P. Bessière, “Approximate discrete probability distribution representation using a multi-resolution binary tree,” in *ICTAI 2003*, (Sacramento, California, USA), 2003.
24. L. Getoor, N. Friedman, D. Koller, and B. Taskar., “Learning probabilistic models of relational structure,” in *Eighteenth International Conference on Machine Learning (ICML)*, (Williams College), 2001.
25. S. Tong and D. Koller, “Active learning for structure in bayesian networks,” in *Seventeenth International Joint Conference on Artificial Intelligence*, (Seattle, Washington), pp. 863–869, August 2001.