Obstacle Avoidance and Proscriptive Bayesian Programming

Carla Koike, Cédric Pradalier, Pierre Bessiere, Emmanuel Mazer

▶ To cite this version:

HAL Id: hal-00019258
https://hal.archives-ouvertes.fr/hal-00019258

Obstacle Avoidance and Proscriptive Bayesian Programming

Carla Koike [a], Cédric Pradalier, Pierre Bessière, Emmanuel Mazer

`firstname.lastname@inrialpes.fr`

Inria[b] Rhône-Alpes & Gravir[c]
655 av. de l'Europe, Montbonnot, 38334 St Ismier Cedex, France

March 2, 2003

[a]Corresponding Author: INRIA Rhône-Alpes, 38334 Saint Ismier cedex France - Phone: +33.476.615348 - Fax: +33.476.615210

[b]Institut National de Recherche en Informatique et en Automatique.

[c]Lab. Graphisme, Vision et Robotique.

Abstract — Unexpected events and not modeled properties of the robot environment are some of the challenges presented by situated robotics research field. Collision avoidance is a basic security requirement and this paper proposes a probabilistic approach called Bayesian Programming, which aims to deal with the uncertainty, imprecision and incompleteness of the information handled to solve the obstacle avoidance problem. Some examples illustrate the process of embodying the programmer preliminary knowledge into a Bayesian program and experimental results of these examples implementation in an electrical vehicle are described and commented. A video illustration of the developed experiments can be found at *http://www.inrialpes.fr/sharp/pub/laplace*

Keywords — Bayesian programming, Obstacle avoidance, Command fusion

Obstacle Avoidance and Proscriptive Bayesian Programming

Carla Koike, Cédric Pradalier, Pierre Bessière, Emmanuel Mazer
GRAVIR-INRIA-INPG Grenoble, France
Carla.Koike@inrialpes.fr, Cedric.Pradalier@inrialpes.fr
Pierre.Bessiere@inrialpes.fr, Emmanuel.Mazer@inrialpes.fr

Abstract

Unexpected events and not modeled properties of the robot environment are some of the challenges presented by situated robotics research field. Collision avoidance is a basic security requirement and this paper proposes a probabilistic approach called Bayesian Programming, which aims to deal with the uncertainty, imprecision and incompleteness of the information handled to solve the obstacle avoidance problem. Some examples illustrate the process of embodying the programmer preliminary knowledge into a Bayesian program and experimental results of these examples implementation in an electrical vehicle are described and commented. A video illustration of the developed experiments can be found at *http://www.inrialpes.fr/sharp/pub/laplace*

I. Introduction

When expected to evolve safely in an unaffected environment, a mobile robot shall execute its tasks avoiding contact with obstacles, if possible following an optimal trajectory during displacements.

Specially in the context of vehicles, the presence of several obstacles at the same time can remarkably increase the complexity of signal processing and turn the obstacle avoidance unfeasible. Depending on the sensors technology some characteristics as size, distance, color and even texture of obstacle surface can change the precision and reliability of the avoidance method. One way to approach this complexity is to deal with the uncertainty in the avoidance reasoning, trying to model the imprecision and incompleteness of information handled.

This paper describes a probabilistic approach and how it can be used to solve the obstacle avoidance problem. Probabilities will be used to express uncertainty, and to represent the knowledge specific to obstacle avoidance, in the context of bayesian programming.

The rest of this paper is organized as follows. First, a short review of methods for obstacle avoidance is presented in section II, some of them using uncertainty representation. Then in section III, the Bayesian Programming Approach is briefly described. Section IV addresses different methods employed to solve the obstacle avoidance problem with Bayesian Programming and each proposed solution is illustrated with results and comments.

II. Obstacle Avoidance Related Work

In situated robotics, it is desired to have the robot evolving in non adapted and populated environment, requiring collision-free motion and fast reaction to unexpected events. This section presents a brief description of methods commonly used in obstacle avoidance in robotics.

In a rough manner, the methods for obstacle avoidance in robotics can be classified in global and local approaches ([7] and [4]). Global approaches are useful when the robot's environment is well known and the complete trajectory from the initial to the target configuration can be computed off-line and optimally. These techniques are closely related to path planning and not really adapted to an unadapted environment.

Distinctly, local approaches do not attempt to model the whole robot environment but they use the sensor values to infer the commands. Being simpler and computationally lighter, they are consequently more appropriate to fast responses in a changing environment. However, they cannot generate optimal solutions and some specific configurations of the obstacles can hinder the robot in local minima (also called traps). Considering the kind of relevant problem in situated (autonomous) robotics, only local approaches are discussed here.

In the specific situation of dynamic obstacles some particular methods were conceived ([10],[1]). When the trajectories are known a-priori, global approaches are usually considered. But in the general case these trajectories are not known and have to be inferred from current observations: obstacle avoidance being performed in reaction to these previsions, local approach methods are more suitable.

a) Potential Field: The general idea of potential field methods, firstly proposed by O.Khatib in 1986, is to have a function coding the navigation goals (which usually consist in reaching a target configuration) while avoiding the obstacles. This function is assumed to decrease near the target position and to increase away from the target and near the obstacles. The navigation problem is then a problem of function optimization: find the necessary commands to bring the robot in the global minimum value of this function. Often, this function is defined solely in relation to the target and obstacles, but other desired constraints in this process can be added as function terms or parameters.

This basic idea (to minimize a potential field function) is repeated in the VFF (Virtual Force Field) method [2], where the robot's work area is divided into cell elements of equal size. These cells form a grid, and each one of these cells have an associated certainty value that describes the measure of confidence that an obstacle exists within this cell area. Only a small window composed of 33×33 cells around the robot are considered in the calculation. Sensor values are used to project the Certainty Value of each cell and the force exerted by each cell repulsing the robot is proportional to the certainty value.

In order to solve some of the short points in the VFF approach, the VFH (Vector Field Histogram) method, initially described in [3], and extensions as VFH+ in [16] and as VFH* lately in [17] were proposed, basically searching to found the best path to reach the target configuration among the safe ones.

b) Steering Angle Field: Steering Angle Field Method, proposed by Feiten et al in 1994, uses the curvature tangent to obstacle to constrain a continuous space (one dimensional steering angle space) and speed control is an iterative negotiation process between the pilot module and the local obstacle avoidance module.

One of the first approaches based on the Steering Angle Field Method was published in [15]. It describes the Curvature-velocity method, which consists of formulating the local obstacle avoidance problem as a constrained optimization in the velocity space (the set of controllable velocities, as the translation and the rotation velocities for example).

Two extensions of this method are the Lane-Curvature Method (LCM) proposed in [9] and the modified-LCM, presented in [13]. In LCM, the environment is divided into lanes and the best lane is chosen to optimize the heading direction. In modified-LCM, a method is used to choose the lane considering the socially preferred turned direction.

c) Dynamic Window: The dynamic window approach was first described in [7], and proposes to avoid the obstacles by searching in the velocities space in order to maximize an objective function. The terms of this function include the measure of the progress toward the target position, the position of the nearest obstacles and the forward velocity of the robot. This method is directly derived from the motion dynamics of the robot, being well-suited for high speed motion.

The problem of computation explosion for this search is solved by reducing the search space using the constraints of the robot dynamics (limited velocities and acceleration and circular trajectories) and also the velocities that are considered safe with respect to the obstacles.

The constraints that reduce the search space are also called *Hard Constraints*, because they must be respected. On the other hand, the objective function terms model the preferences of the robot movement, and these are called *Soft Constraints*.

In a hybrid method developed lately [8], a sensor based dynamic window approach is responsible for avoiding detected obstacles. Additionally, informations about obstacles probably not detected by the sensors could be acquired in a map, so that collision is avoided. Localization in the map is estimated using a metric version of the Markov Localization algorithm.

In [4], a generalization of the dynamic window approach is proposed, combining methods of motion planning in an attempt to create a global reactive behavior at high speeds.

III. BAYESIAN PROGRAMMING BASIC CONCEPTS
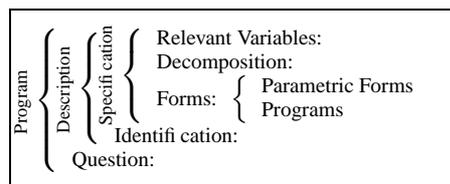
Fig. 1. Bayesian Program Structure

When programming a robot, the programmer builds an abstract representation of its environment, which is basically composed of geometric, analytic and symbolic notions. In a way, the programmer imposes on the robot his or her own abstract conception of the environment, supposing that the environment can be fully described by the abstract concepts used.

Due to the irreducible incompleteness of the model, it is not always easy to link these abstract concepts with the robot's raw signals (either obtained from the robot's sensors or being

sent to the robot's actuators). Controlling the environment is the usual answer to these difficulties, but it may not be desirable or possible when the robot must act in an environment not specifically designed for it, populated, or subject to unexpected and unattended events.

Probabilistic methodologies and techniques offer possible solutions to the incompleteness and uncertainty difficulties when programming a robot. The basic programming resources are *probability distributions*, which are generated placing side by side the programmer's conception task (actually, the expression of his or her preliminary knowledge about the robot, the environment and the task) and experimental data obtained from the environment.

The Bayesian Programming approach was first proposed in [12], originally as a tool for robotic programming, but nowadays it is used in a wider scope of applications (see [6], [14], and [5]).

In this approach, a probability distribution is associated with the uncertainty of a logical proposition value. The usual notion of a *Logical Proposition* (either true or false) and its operators (negation, conjunction and disjunction) are used when defining a *Discrete Variable*.

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 $i \neq j$, $x_i \wedge x_j$ is false) and exhaustive (at least one of these propositions x_i is true).

The probability distributions assigned to logical propositions are always defined according to some preliminary knowledge, identified as π: so $P(x_i \mid \pi)$ gives the probability distribution of the variable X having the value x_i, knowing π.

Most of the time, probabilities will be manipulated using the conjunction rule, also known as Bayes rule:

$$P(XY \mid \pi) = P(X \mid \pi)P(Y \mid X \pi) = P(Y \mid \pi)P(X \mid Y \pi). \tag{1}$$

Most details about the inference postulates and rules for carrying out probabilistic reasoning in this context can be found in [11]. They are the basis for the definition of a generic formalism to specify probabilistic models called *Bayesian Programs*.

The elements of a Bayesian Program are presented in figure 1.

- A program consists of a description, which composes a knowledge base (declarative part), and a question, that restores via inference some of this knowledge (procedural part).
- A description is built in 2 phases: a specification phase where the programmer expresses its preliminary knowledge and an identification (or learning) phase where the experimental data are taken into account.
- Preliminary knowledge specification is constructed from a set of pertinent variables, a decomposition of the joint distribution into a product of simpler terms, and a set of forms, one for each term.
- Forms are either parametric forms (given a priori or learned from the experimental data) or questions to other Bayesian Program.

Each of the these elements is briefly outlined in the rest of this section.

A. Description

The purpose of a description is to specify an effective method to compute a joint distribution on a set of relevant variables $\{X^1, X^2, \ldots, X^n\}$, given a set of experimental data δ and a preliminary knowledge π.

In the specification phase of the description, it is necessary to:

- Define a set of relevant variables $\{X^1, X^2, \ldots, X^n\}$, on which the joint distribution shall be defined;
- Decompose the joint distribution into simpler terms, following the conjunction rule. Given a partition of $\{X^1, X^2, \ldots, X^n\}$ into k subsets, the variables $\{L^1, L^2, \ldots, L^k\}$, each being the conjunction of the variables in each of these subsets, the conjunction rule leads to the following decomposition of the joint distribution:

$$P(L^1 \ L^2 \ \ldots \ L^n \mid \delta \ \pi) = P(L^1 \mid \delta \ \pi) \times P(L^2 \mid L^1 \ \delta \ \pi) \times \ldots \times P(L^k \mid L^{k-1} \ \ldots L^2 \ L^1 \ \delta \ \pi) \tag{2}$$

Conditional independence hypotheses can allow further simplification, and such a simplified decomposition of the joint distribution is called decomposition.

- Define the forms for each term $P(L^i | L^{i-1} \ \ldots L^2 \ L^1 \ \delta \ \pi)$ in the decomposition; i.e. each term is associated with either a parametric form (as a function $f_\mu(L^i)$) or to another Bayesian Program. In general, μ is a vector of parameters that may depend on $L^{i-1} \ \ldots L^2 \ L^1$ or δ or both. Learning take place when some of these parameters are computed using the data set δ.

Given a description $P(X^1\ X^2\ \ldots\ X^n\ \big|\ \delta\ \pi)$, a question is obtained by partitioning the variables $\{X^1, X^2, \ldots, X^n\}$ into three sets: *Searched*, *Known* and *Unknown* variables. A question is defined as the distribution:

$$P(\mathrm{S_{earch}}\ \big|\ \mathrm{K_{nown}}\ \delta\ \pi).$$

In order to answer this question, the following general inference is used:

$$P(\mathrm{S_{earch}}\ \big|\ \mathrm{K_{nown}}\ \delta\ \pi) = \sum_{\mathrm{U_{nknown}}} P(\mathrm{S_{earch}\ U_{nknown}}\ \big|\ \mathrm{K_{nown}}\ \delta\ \pi) \tag{3}$$

$$= \frac{\sum_{\mathrm{U_{nknown}}} P(\mathrm{S_{earch}\ U_{nknown}\ K_{nown}})}{P(\mathrm{K_{nown}})} = \frac{\sum_{\mathrm{U_{nknown}}} P(\mathrm{S_{earch}\ U_{nknown}\ K_{nown}})}{\sum_{\mathrm{U_{nknown}, S_{earch}}} P(\mathrm{S_{earch}\ U_{nknown}\ K_{nown}})}$$

## IV. OBSTACLE AVOIDANCE USING BAYESIAN PROGRAMMING

Bayesian programming offers a wide range of possibilities for solving robotics problems and incorporating the programmer's preliminary knowledge in the specification of the program description part (choice of relevant variables, joint distribution decomposition and parametric forms).

When stating the elements in the Bayesian program specification phase, the choice of the relevant variables and the joint distribution decomposition has to follow some restrictions: the relevant variables are highly problem dependent and the decomposition is limited to those imposed by conjunction (Bayes) rule. On the other hand, the parametric forms choice is free and depends a lot on the programmer's experience and his/her knowledge of the problem to be solved.

In order to find the most appropriate way to embody the preliminary knowledge about the obstacle avoidance and the Cycab vehicle in a Bayesian program, some solutions are described in this section, differing mostly in the specification of the program description part. They illustrate the process of evolving from simpler solution toward a more complex and efficient specification.

Before getting into the details of each version, a concise presentation of the experimental platform used is shown as it is essential in the choice of relevant variables choice as well as in the joint distribution decomposition. After that, the proposed solutions are presented in a increasing order of complexity.

A. Experimental platform: the Cycab
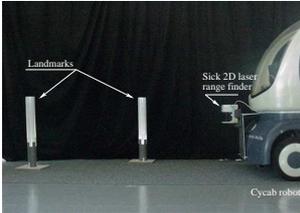
Our experimental platform is a robotic golf cab called Cycab. This mini-car is equipped with a Sick laser range finder with an efficient range of 20 meters and an uncertainty of about 5-10 centimeters. Obstacles in front of the vehicle are detected with the laser, but considering the restrictions due to the sensor technology and its position, some obstacles may not be detected (very small objects, objects out of the angle range, and material and color dependent surfaces).

Our laser range finder can detect objects in a range of -90 degrees to 90 degrees, with a half degree precision. The output value is the distance of a detected obstacle (from 0 to 8191 cm). If there is no obstacle, the maximal value is returned. As the precision is bigger in the range from 20 to 2000cm, the sensor raw values are preprocessed and discretized to have values from 0 (0cm) to 200 (2000cm or greater), with a precision of 10 cm.

Additionally, the 180 degrees range is divided in 8 regions called *Zones* and for each zone, just one obstacle distance is considered (defined as the smaller measure given by the Sick sensor in this angular range). We call *Zone 1* the first region at the vehicle left and *Zone 8* is the first region at the vehicle right.

The Cycab can be commanded in speed and steering angle. We discretized these inputs in speed from 0 to 5, and steering angle from -5 to 5. As we only have a sensor on the cycab front, we cannot have information about its back. So we decided to forbid backward motions.

Finally, we would like to emphasize that all experimentations took place in an unmodified private parking lot with parked cars and undisturbed traffic of cars and pedestrians.

B. Behavior Combination

This first version has a very simple and basic reasoning: each zone proposes values of speed and steering angle based on the distance to the nearest obstacle inside this zone.

One Bayesian program is necessary to each zone in order to express the different actions in relation to the detected obstacle position. An additional Bayesian program is required to combine the actions of all zones.
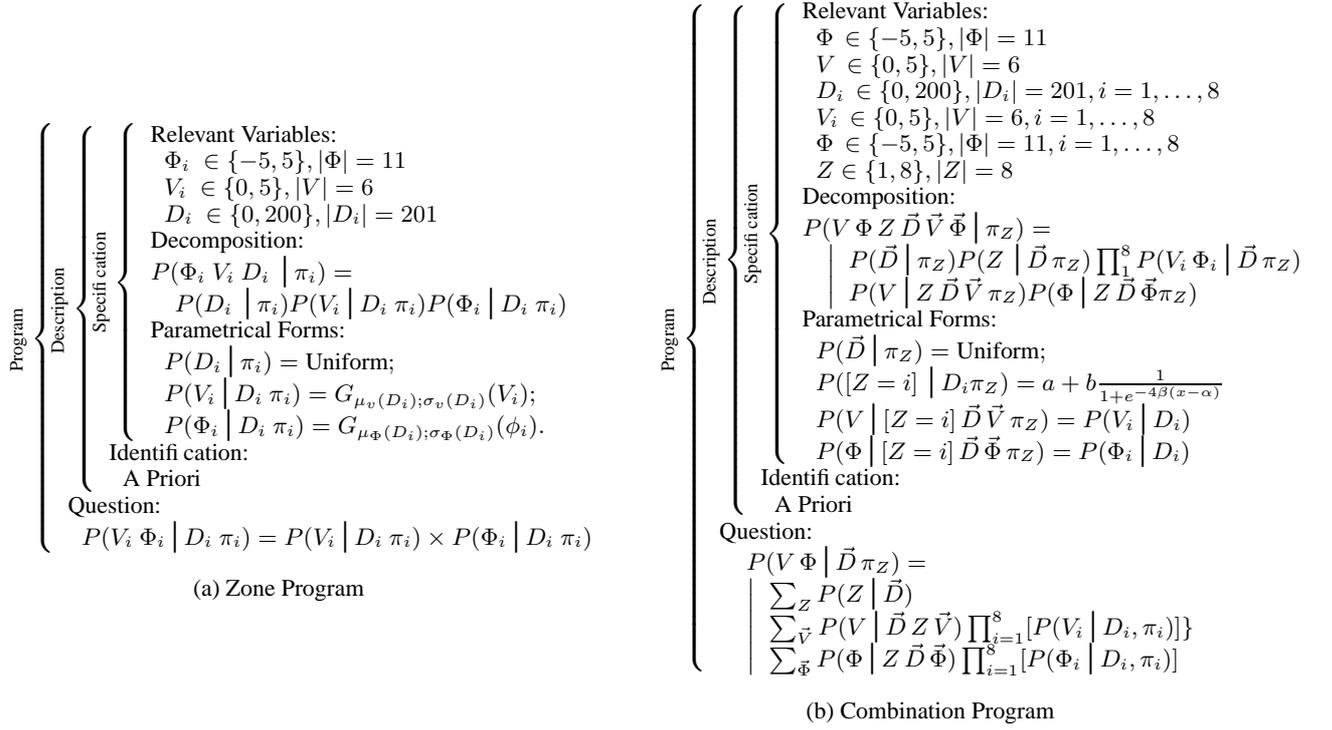
Fig. 2. Bayesian programs for behavior combination

(a) Zone Program

Program { Description { Specification {

Relevant Variables:
$\Phi_i \in \{-5,5\}, |\Phi| = 11$
$V_i \in \{0,5\}, |V| = 6$
$D_i \in \{0,200\}, |D_i| = 201$

Decomposition:
$P(\Phi_i \, V_i \, D_i \mid \pi_i) =$
$\quad P(D_i \mid \pi_i) P(V_i \mid D_i \, \pi_i) P(\Phi_i \mid D_i \, \pi_i)$

Parametrical Forms:
$P(D_i \mid \pi_i) = \text{Uniform};$
$P(V_i \mid D_i \, \pi_i) = G_{\mu_v(D_i); \sigma_v(D_i)}(V_i);$
$P(\Phi_i \mid D_i \, \pi_i) = G_{\mu_\Phi(D_i); \sigma_\Phi(D_i)}(\phi_i).$

Identification:
A Priori

Question:
$P(V_i \, \Phi_i \mid D_i \, \pi_i) = P(V_i \mid D_i \, \pi_i) \times P(\Phi_i \mid D_i \, \pi_i)$

(b) Combination Program

Program { Description { Specification {

Relevant Variables:
$\Phi \in \{-5,5\}, |\Phi| = 11$
$V \in \{0,5\}, |V| = 6$
$D_i \in \{0,200\}, |D_i| = 201, i = 1, \ldots, 8$
$V_i \in \{0,5\}, |V| = 6, i = 1, \ldots, 8$
$\Phi \in \{-5,5\}, |\Phi| = 11, i = 1, \ldots, 8$
$Z \in \{1,8\}, |Z| = 8$

Decomposition:
$P(V \, \Phi \, Z \, \vec{D} \, \vec{V} \, \vec{\Phi} \mid \pi_Z) =$
$\quad P(\vec{D} \mid \pi_Z) P(Z \mid \vec{D} \, \pi_Z) \prod_1^8 P(V_i \, \Phi_i \mid \vec{D} \, \pi_Z)$
$\quad P(V \mid Z \, \vec{D} \, \vec{V} \, \pi_Z) P(\Phi \mid Z \, \vec{D} \, \vec{\Phi} \pi_Z)$

Parametrical Forms:
$P(\vec{D} \mid \pi_Z) = \text{Uniform};$
$P([Z = i] \mid D_i \pi_Z) = a + b \frac{1}{1 + e^{-4\beta(x-\alpha)}}$
$P(V \mid [Z = i] \, \vec{D} \, \vec{V} \, \pi_Z) = P(V_i \mid D_i)$
$P(\Phi \mid [Z = i] \, \vec{D} \, \vec{\Phi} \pi_Z) = P(\Phi_i \mid D_i)$

Identification:
A Priori

Question:
$P(V \, \Phi \mid \vec{D} \, \pi_Z) =$
$\quad \sum_Z P(Z \mid \vec{D})$
$\quad \sum_{\vec{V}} P(V \mid \vec{D} \, Z \, \vec{V}) \prod_{i=1}^8 [P(V_i \mid D_i, \pi_i)]\}$
$\quad \sum_{\vec{\Phi}} P(\Phi \mid Z \, \vec{D} \, \vec{\Phi}) \prod_{i=1}^8 [P(\Phi_i \mid D_i, \pi_i)]$

Figure 3(a) shows the Bayesian program written for each zone. In addition to the obstacle distance D_i, each zone includes the vehicle speed variable V_i and the steering angle Φ_i as relevant variables. In the variables description, $V \in \{0,5\}$ means that V values ranges from 0 to 5 and $|V| = 6$ indicates that V has 6 discretized values.

The joint distribution decomposition assumes independence between V_i and Φ_i: the vehicle forward speed and steering angle are decided based only in the obstacle distance value. These distributions are considered gaussian curves, whose average represents an ideal value and whose standard deviation adds uncertainty in relation to this ideal value choice. Practically, the standard deviation can also be considered as either the strength of a constraint, or the confidence in a command proposition.

In Figure 3(b), the Bayesian program for the zone combination is displayed. The influence of V_i and Φ_i in the vehicle V and Φ is determined by a probabilistic variable Z whose value indicates the zone with the nearest obstacle. The probability distribution of Z values is a sigmoid curve: the probability of $Z = i$ increases with decreasing values of distance of the obstacle in zone i. The value of Z is used to decide which V_i and Φ_i will be used to compose the probability distribution of V and Φ. For simplicity of notation, let $\forall A, \vec{A} = A_1 \ldots A_8$.

Finally, we want to know which values shall take V and Φ knowing the values of $\vec{D}$ and the expression that solves this question is shown in the bottom of figure 3(b).

This solution was implemented, tested and executed in the Cycab vehicle and the results were coherent, but adjusting the functions f_i for each zone was very troublesome. In addition to that, using the variable Z makes this solution a weighted switching between zones, when the desired is to combine the proposition of all zones and estimate a reasonable value of V.

For instance, a left zone estimates that due to a potential danger the translation speed should be small, for example $V = 1$. At the same time instant, a right zone sees no obstacle and proposes for translation speed $V = 5$. The resulting value of V will be greater than 1, showing that collision danger is not taken into account properly. Furthermore, all these nested sums that have to be calculated mean a lot of computational time.

The standard deviation of the Gaussian in the definition of $P(V_i \mid D_i, \pi_i)$ and $P(\Phi_i \mid D_i, \pi_i)$ for each zone is not taken into account, meaning there is redundancy of information.

C. Prescriptive Behavior Fusion

Basically, this version wishes to reduce the redundancy, using the standard deviation values of each Gaussian in the definition of $P(V_i \mid D_i, \pi_i)$ and $P(\Phi_i \mid D_i, \pi_i)$ for each zone as the measure of influence of each speed V_i and angle Φ_i in the variables V and Φ: a command fusion approach. Neither the variable Z nor the functions f_i for each zone will be necessary any more.
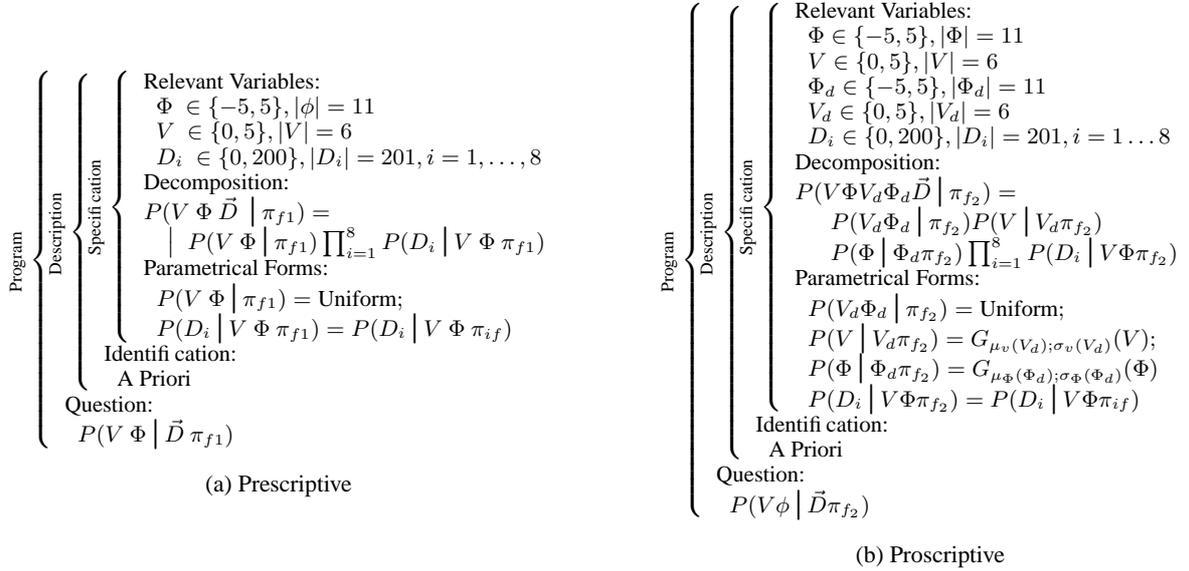
The Bayesian program of each zone is similar to the one presented in figure 3(a), but instead of having their own V_i and Φ_i variables, all zones use unique variables V and Φ. Furthermore, due to the changes in the command fusion program, the question asked to each zone program changes to:

$$P(D_i \mid V \, \Phi \, \pi_{if})$$ (4)

$$= \frac{P(D_i \mid \pi_{if})P(V \mid D_i \, \pi_{if})P(\Phi \mid D_i \, \pi_{if})}{\sum_{D_i} P(V \mid D_i \, \pi_{if})P(\Phi \mid D_i \, \pi_{if})}$$

As in the solution proposed in the previous section, each zone proposes values of speed and steering angle based on the distance to the nearest obstacle inside this zone. However, instead of combining the different propositions from each zone, a command fusion is performed with the variables V and Φ. The Bayesian program for the command fusion is shown in the figure 4(a), where the relation with the zone program is given by the terms $P(D_i \mid V \, \Phi \, \pi_{if})$, whose expression is shown in equation 4.

Fig. 3.   Command Fusion Bayesian Program

Relevant Variables:
$\Phi \in \{-5, 5\}, |\phi| = 11$
$V \in \{0, 5\}, |V| = 6$
$D_i \in \{0, 200\}, |D_i| = 201, i = 1, \ldots, 8$
Decomposition:
$P(V \, \Phi \, \vec{D} \mid \pi_{f1}) =$
$\quad \mid P(V \, \Phi \mid \pi_{f1}) \prod_{i=1}^{8} P(D_i \mid V \, \Phi \, \pi_{f1})$
Parametrical Forms:
$\quad P(V \, \Phi \mid \pi_{f1}) = $ Uniform;
$\quad P(D_i \mid V \, \Phi \, \pi_{f1}) = P(D_i \mid V \, \Phi \, \pi_{if})$
Identification:
$\quad$ A Priori
Question:
$\quad P(V \, \Phi \mid \vec{D} \, \pi_{f1})$

(a) Prescriptive

Relevant Variables:
$\Phi \in \{-5, 5\}, |\Phi| = 11$
$V \in \{0, 5\}, |V| = 6$
$\Phi_d \in \{-5, 5\}, |\Phi_d| = 11$
$V_d \in \{0, 5\}, |V_d| = 6$
$D_i \in \{0, 200\}, |D_i| = 201, i = 1 \ldots 8$
Decomposition:
$P(V \Phi V_d \Phi_d \vec{D} \mid \pi_{f2}) =$
$\quad P(V_d \Phi_d \mid \pi_{f2})P(V \mid V_d \pi_{f2})$
$\quad P(\Phi \mid \Phi_d \pi_{f2}) \prod_{i=1}^{8} P(D_i \mid V \Phi \pi_{f2})$
Parametrical Forms:
$\quad P(V_d \Phi_d \mid \pi_{f2}) = $ Uniform;
$\quad P(V \mid V_d \pi_{f2}) = G_{\mu_v(V_d); \sigma_v(V_d)}(V);$
$\quad P(\Phi \mid \Phi_d \pi_{f2}) = G_{\mu_\Phi(\Phi_d); \sigma_\Phi(\Phi_d)}(\Phi)$
$\quad P(D_i \mid V \Phi \pi_{f2}) = P(D_i \mid V \Phi \pi_{if})$
Identification:
$\quad$ A Priori
Question:
$\quad P(V \phi \mid \vec{D} \pi_{f2})$

(b) Proscriptive

The parametric forms for $P(V \mid D_i \, \pi_{if})$ and $P(\Phi \mid D_i \, \pi_{if})$ defined in each zone implement the relation between the vehicle movement and the distance of the detected obstacles. Figure 5(a) shows $P(\Phi \mid d_4 \, \pi_{4f})$ in the curve called *Zone 4* for a detected obstacle ahead, slightly at the vehicle left at 2 meters distance. As an obstacle detected in this zone indicates that the vehicle shall turn right to avoid it, it can be seen that for positive values of Φ the probability values are higher than in the case of negative values.

Supposing that this is the only obstacle detected, all other zones will give for $P(\Phi \mid d_i \, \pi_{if})$ curves near uniform as shown in figure 5(a), curve *Zone 1*, and the command fusion results a curve for $P(\Phi \mid \vec{d} \, \pi_{f1})$ similar to $P(\Phi \mid d_4 \, \pi_{4f})$, not shown in the figure. A similar reasoning is used in the case of the translation speed, as the curves have analogous definition.
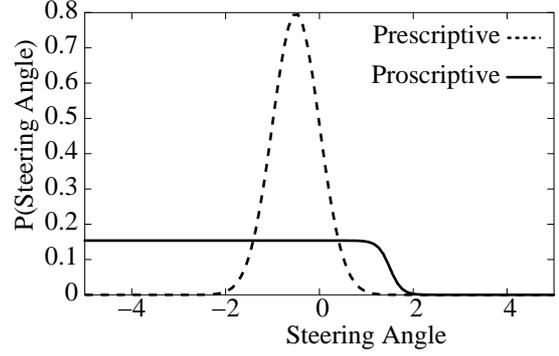
This version was also implemented, tested and executed on the Cycab. The results are quite the same as the version presented in section IV-B, but adjusting the parameters was easier, as only the standard deviation curves should be tunned to express the relation between the importance of the estimates of the zones. The transition between the zones seemed smoother, both in speed and steering angle changes.

However, the command fusion result does not show reasonable results when two zones indicate inconsistent commands, as in the example shown in figure 5(a). One obstacle of considerable size standing in front of the

Fig. 4. Model curves
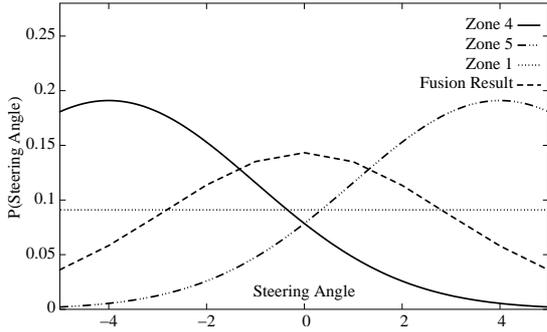
(a) Prescriptive fusion result   (b) Prescriptive and Proscriptive Strategy

vehicle is detected by zones 4 (slightly left) and 5 (slightly right). The curve of $P(\Phi \mid d_4 \, \pi_{4f})$ (called *Zone 4* in the figure) indicates a positive steering angle and the curve of $P(\Phi \mid d_5 \, \pi_{5f})$ (called *Zone 5*) proposes a negative one. If there are no other obstacles detected, the resulting curve of $P(\Phi \mid \vec{d} \, \pi_{f1})$ points out that the vehicle should go straight, running right into the obstacle, as illustrated in curve *Fusion Result* at figure 5(a).

Besides, it is not possible to change the desired movement in absence of obstacles, not allowing the obstacle avoidance to be executed jointly with other tasks.

D. Proscriptive Behavior Fusion

The problems detected in the previous section are due to the choice of $P(V \mid D_i \, \pi_{if})$ and $P(\Phi \mid D_i \, \pi_{if})$: in order to avoid an obstacle, it is not absolutely essential to model the necessary action the vehicle shall take. It is possible to indicate only the prohibited actions (the ones that can lead the vehicle directly into the obstacle), and let the fusion process decide which command represents the best vehicle action between the desired and the possible ones.

This reasoning introduces the main idea of prescriptive and proscriptive programming approaches. A prescriptive approach to obstacle avoidance indicates *what to do* in order not to bump with the obstacle: for example, in order to avoid an object located at its right side, the vehicle shall turn left. Oppositely, a proscriptive approach point out *what shall not be done* to avoid the collision: if the object is at vehicle right side, then it shall not go to the right.

Basically, the implementation of these approaches depends on the choice of $P(V \mid D_i \, \pi_{if})$ and $P(\Phi \mid D_i \, \pi_{if})$ inside each zone program. In a proscriptive version, the speed and steering curves in each zone show confidence only about the values the vehicle *cannot execute* in order to avoid the obstacle. Therefore, the values of speed and steering angle that bring the vehicle nearer to the obstacle are avoided, using a zero or very small probability. An uniform (or near uniform) distribution over all other values is created, so that in the fusion process the *allowed* values by the several zones will be the most probable. Figure 5(b) below illustrates the difference of $P(\Phi \mid d_4 \, \pi_{4f})$ for each approach, where the prescriptive approach curve is a gaussian (average value is chosen as the nominal value of Φ to avoid the obstacle) and the proscriptive approach curve is a sigmoid with very small probability values to the forbidden values of Φ.

With the purpose of implementing the proscriptive obstacle avoidance strategy, the parametric forms for $P(V \mid D_i \, \pi_{if})$ and $P(\Phi \mid D_i \, \pi_{if})$ inside each zone program are defined as:

$$\delta = \frac{d_i - d_{\min}}{d_{\max} - d_{\min}} v_{\max}, \; \gamma = \frac{d_i - d_{\min}}{d_{\max} - d\min} \Phi_{\max}$$

$$P_i(V|D_i) = \frac{1}{K} \left(1 - sigmoid_{\delta, \ln(3)}(v) \right)$$

$$P_i(\Phi|D_i) = \frac{1}{K} \left(1 - sigmoid_{\gamma, \ln(3)}(\Phi) \right)$$

Each zone has different parameters $d_{\min}$ and $d_{\max}$ in order to provide different levels of danger depending on the distance of the obstacle in this zone. Therefore, when the distance to an obstacle is smaller than $d_{\min}$, the vehicle shall stop, whatever the distance measured in the other zones. Conversely, an obstacle at a distance greater than $d_{\max}$ can be ignored. Obviously, the values of $d_{\min}$ are greatly distinct in the right hand zone and in the front zone, for example.

Considering no obstacle is at view, the resulting fusion curve would be uniform. Consequently, reference values for both speed and steering angle can be added to guide the vehicle in a desired trajectory, solving another handicap encountered in the previous versions.

The whole Bayesian program for the command fusion using proscriptive programming can be seen in figure 4(b). In the joint distribution, two new variables are added to express the desired values for the translation speed and steering angle (V_d and Φ_d). Φ_d is specially connected to the goal configuration and it shows the direction desired to follow in order to reach the target.

1) An alternative for $P(\Phi \mid D_i)$: It is important to notice that sigmoids are not the best function for the distribution of both $P(V \mid D_i\ \pi_{if})$ neither $P(\Phi \mid D_i\ \pi_{if})$. Actually, when an object is detected in front of the car, we have two solutions: either steering left or right. Both solutions are equally good, thus the distribution would better look like an inversed gaussian, shown in figure 6(a) or mathematically:

$$P(\Phi \mid D_i\ \pi_{if}) = \frac{1}{K}\left(1 - e^{(\frac{\Phi - \mu}{\sigma})^2}\right). \tag{5}$$

Fig. 5. Proscriptive models
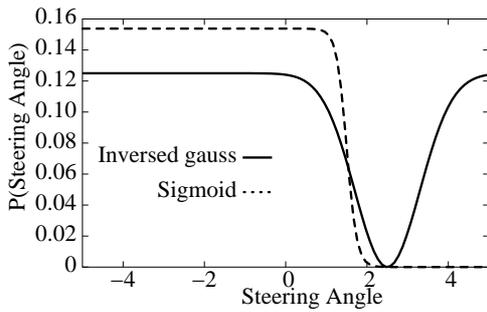
(a) Proscriptive models for steering angle
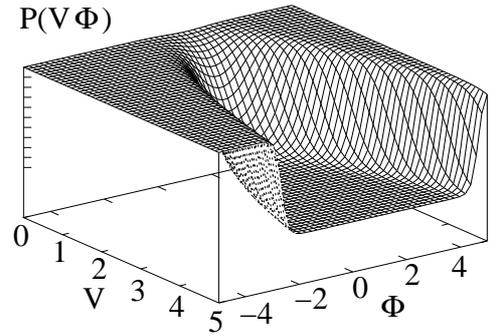
(b) Linked model for speed and steering

Another drawback in the program of the previous section is the initial assumption of independence between Φ and V. Clearly, the steering angle necessary to avoid an obstacle depends on the vehicle translation speed, as well as the choice of the translation speed has a close relation with the steering angle (it is not sensible to go forward at full speed and steer fully at right at the same time instant).

Both disadvantages can be improved changing the specification part of the zone Bayesian program described in figure 3(a). Initially, in order to add the dependency between Φ and V a unique probability distribution $P_i(V\Phi|D_i\ \pi_{if})$ is built, and neither $P(\Phi \mid D_i\ \pi_{if})$ nor $P(V \mid D_i\ \pi_{if})$ are no longer necessary. When building this term, the shape shown in the figure 6(a) and in the equation IV-D.1 shall be used rather than the sigmoids to express the similarity between two equally good solutions (steering left or right, for example).

When it is feasible to build an adequate kinematic model of the vehicle, it is possible to estimate for any Φ and any distance to an obstacle in any zone, the maximum speed $v_{\max}$ which grants a safety minimal time (two seconds, for instance) before hitting the obstacle.

The robot dynamics can then be added to the obstacle avoidance as constraints in the way of previously allowed pair of values of V, Φ. These allowed values can be related to:

- The maximal values of acceleration that result in a range of possible values greater or smaller than the present value of speeds;
- The relation between the distance to the obstacle and the time the robot needs to react properly;

- The robot system behavior when in high speeds.

Once we know $v_{\max}^i(d, \Phi)$, we can define a new joint distribution for zone Bayesian program:

$$P(V\Phi D_i \mid \pi_{if}) = P(D_i \mid \pi_{if})P(V\Phi \mid D_i \, \pi_{if}) \tag{6}$$

where $P(D_i \mid \pi_{if})$ remains uniform and $P(V\Phi \mid D_i \, \pi_{if})$ is some 2D sigmoid as in the equation 7 and also shown in figure 6(b).

$$P_i(V\Phi|D_i \, \pi_{if}) = \frac{1}{K}\left(1 - \frac{1}{1 + e^{-4\beta(v - v_{\max}(\Phi))}}\right) \tag{7}$$

Following the description in [8], the hard constraints are the ones that shall be obeyed while the soft constraints are expectations, which shall be optimized whenever possible. In order to assure the relation between these constraints, some constants are used when defining the objective function to be optimized or the relation between attractive and repulsive forces.

Using this approach, hard constraints are expressed by the zero probability values or Dirac in the curves modeling the joint distribution of each zone $P(V\Phi D_i)$. They represent the obstacles and the security about not touching them. The constraints based on the circular movement and the dynamic model of the robot can also be added when expressing this joint distribution.

On the other hand, soft constraints can be represented by the covariance values of the Gaussian probability distributions $P(V \mid V_d)$ and $P(\Phi \mid \Phi_d)$. By adequately choosing their standard deviation σ_V or σ_Φ, we can determine whether the robot would better turn or stop. The greater σ_Φ compared to σ_V, the more the robot will try to turn rather than stop.

2) Reference Following: As described in the previous sections, $P(V_d)$ and $P(\Phi_d)$ are probability distributions based on the desired vehicle trajectory. Regarding the obstacle avoidance presented here, they are considered as *a-priori* knowledge, but it is straightforward to consider them as connecting points in a structured framework. Higher levels specify the desired values of V_d and Φ_d, setting the distributions $P(V_d)$ and $P(\Phi_d)$ in a lower level (obstacle avoidance), which shall try to follow them, preserving the safety (avoiding the obstacles) as an essential behavior always in execution.

The higher levels can be composed of reactive behaviors based on environment features (phototaxis, thigmo-taxis or gradient following, for example) or a planner which generates the sequence based in mapping and self localization. Another simple example is to have the joystick as the generator of the desired direction to follow, but restrained by the obstacle avoidance.

V. Conclusions

In this paper we presented our work about probabilistic expression of an obstacle avoidance task. This work took place in the context of a new programming technique based on Bayesian inference, called *Bayesian Programming*.

We put the stress on comparing the different forms of probability distribution which may be used in a context of command fusion. We found that security constraints should not be expressed the same way as commands made to fulfill an objective. In general, security is guaranteed by specifying what the robot must not do | what we call *Proscriptive Programming*|, and objective is reached by representing what the robot should execute | called *Prescriptive Programming*.

Besides the advantages of *Bayesian Programming* | namely, a) a clear and well-defined mathematical back-ground, b) a generic and uniform way to formulate problems | the main interest of our approach relies in a) the augmented expression abilities offered by associating commands with probability distributions and using command bayesian fusion to mix them, b) its built-in modularity. Actually, this obstacle avoidance scheme is currently used in such applications as assisted driving, sensor servoing and trajectory execution from a path planner.

VI. Acknowledgments

VII. REFERENCES

[1] S. Blondin. Planification de mouvements pour véhicule automatisé en environnement partiellement connu. Mémoire de Diplôme d'Etudes Approfondies, Inst. Nat. Polytechnique de Grenoble, Grenoble (FR), June 2002.

[2] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, Sept/Oct 1989.

[3] J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, June 1991.

[4] O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Detroit, MI (US), May 1999.

[5] F. Colas. Modélisation analytique et bayésienne des neurones d'orientation de la tête. Mémoire de Diplôme d'Etudes Approfondies, Univ. Joseph Fourier, Grenoble (FR), June 2002.

[6] C. Coué, Th. Fraichard, P. Bessière, and E. Mazer. Using bayesian programming for multi-sensor data fusion in automotive applications. In *Proc. of the IEEE Intelligent Vehicle Symp.*, Versailles (FR), June 2002. Poster session.

[7] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33, March 1997.

[8] D. Fox, W. Burgard, S. Thrun, and A.B. Cremers. A hybrid collision avoidance method for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1238–43, Leuven, Belgium, May 1998.

[9] N. Y. Ko and R. Simmons. The lane-curvature method for local obstacle avoidance. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Victoria, BC (CA), October 1998.

[10] F. Large, S. Sekhavat, Z. Shiller, and C. Laugier. Towards real-time global motion planning in a dynamic environment using the NLVO concept. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Lausanne (CH), September-October 2002.

[11] O. Lebeltel, P. Bessière, J. Diard, and E. Mazer. Bayesian robots programming. Research Report 1, Les Cahiers du Laboratoire Leibniz, Grenoble (FR), May 2000.

[12] O. Lebetel. *Programmation Bayesienne de Robots*. Ph.D. Thesis, Inst. Nat. Polytechnique de Grenoble, Grenoble (FR), October 1999.

[13] Vicente Matellán and Reid Simmons. Implementing human-acceptable navigational behavior and a fuzzy controller for an autonomous robot. In *Proceedings of III Workshop de Agentes Físicos WAF-2002*, pages 113–120, Murcia, Spain, March 2002.

[14] K. Mekhnacha, E. Mazer, and P. Bessière. A robotic CAD system using a Bayesian framework. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Takamatsu (JP), November 2000.

[15] R. Simmons. The curvature-velocity method for local obstacle avoidance. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 3375–3382, Minneapolis, MN (US), April 1996.

[16] I. Ulrich and J. Borenstein. VFH+: Reliable obstacle avoidance for fast mobile robots. In *IEEE Int. Conf. Robotics Automation*, pages 1572–1577, Leuven, Belgium, May 1998.

[17] I. Ulrich and J. Borenstein. Vfh*: Local obstacle avoidance with look-ahead verification. In *IEEE Int. Conf. Robotics Automation*, pages 2505–2511, San Francisco, CA, April 2000.