Submitted to the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems
October 27-31, 2003. Las Vegas, NV (US)

Expressing Bayesian Fusion as a Product of Distributions: Applications in Robotics

Cédric Pradalier and Francis Colas
Inria[a] Rhône-Alpes & Gravir[b]
655 av. de l'Europe, Montbonnot, 38334 St Ismier Cedex, France

March 2, 2003

[a]Institut National de Recherche en Informatique et en Automatique.
[b]Lab. Graphisme, Vision et Robotique.

Abstract — More and more fields of applied computer science involve fusion of multiple data sources, such as sensor readings or model decision. However incompleteness of the models prevent the programmer from having an absolute precision over their variables. Therefore bayesian framework can be adequate for such a process as it allows handling of uncertainty. We will be interested in the ability to express any fusion process as a product, for it can lead to reduction of complexity in time and space. We study in this paper various fusion schemes and propose to add a consistency variable to justify the use of a product to compute distribution over the fused variable. We will then show application of this new fusion process to localization of a mobile robot and obstacle avoidance.

Keywords — Bayesian programming, Data fusion, Command fusion

Acknowledgements — This work was sponsored by the European Project BIBA and the French ministry of research.

Expressing Bayesian Fusion as a Product of Distributions: Applications in Robotics

Cédric Pradalier and Francis Colas
GRAVIR-INRIA-INP Grenoble, France
firstname.lastname@inrialpes.fr

***Abstract*—More and more fields of applied computer science involve fusion of multiple data sources, such as sensor readings or model decision. However incompleteness of the models prevent the programmer from having an absolute precision over their variables. Therefore bayesian framework can be adequate for such a process as it allows handling of uncertainty. We will be interested in the ability to express any fusion process as a product, for it can lead to reduction of complexity in time and space. We study in this paper various fusion schemes and propose to add a consistency variable to justify the use of a product to compute distribution over the fused variable. We will then show application of this new fusion process to localization of a mobile robot and obstacle avoidance.**

I. Introduction

Data fusion is a common issue of mobile robotics, computer assisted medical diagnosis or behavioral control of simulated character for instance. This includes *estimation* of some state variable with respect to some sensory readings, *fusion* of experts' diagnosis or *action selection* among various module opinions.

In principle, fusion of multi-model data provides significant advantages over single source data. In addition to the statistical advantage gained by combining same-source data (obtaining an improved estimate of a physical phenomena via redundant observations), the use of multiple types of models may increase the accuracy with which a phenomenon can be observed and characterized. Applications for multi-model, and specifically multi-sensor, data fusion are widespread, both in military and civilian areas. Ref. [3] provides an overview of multi-sensor data fusion technology and its applications.

Besides, as sensory readings, opinions from experts or motor commands can not be known with arbitrary precision, pure logic can not manage efficiently a fusion process. Such issue can therefore be formalized in the bayesian framework, in order to confront different knowledge in an uncertain environment. This is illustrated for example in previous works by Lebeltel[5] and Coue[2]. The CyberMove project is precisely involved of robotics and in particular in probabilistic programming. This paradigm is applied for car-like robots in the framework of bayesian theory as depicted in [4]. As general bayesian inference problem has been shown to be NP-Hard [1], much work is dedicated to applicability and complexity reduction of the inference.

We are interested in evaluating a variable V knowing other variables $V_1 \dots V_n$: $P(V \mid V_1 \dots V_n)$. In the case of multi-sensor fusion, V could stand for the pose of a robot and $V_1 \dots V_n$, the values of its sensors. In this case, the programmer may specify each sensor model: $P(V_k \mid V)$. This is called a *direct* model and Bayes' rule can be applied to infere directly the fused distribution. Additionnally we will show in section II that $P(V \mid V_1 \dots V_n)$ is proportionnal to the product of $P(V \mid V_k)$ the opinion from each model. This property is interesting as it can lead to time and memory effective computation.

However, in the case of command fusion, V could stand for the command to apply. The simplest to specify for the programmer is now usually the influence of each sensor on the actual command: $P(V \mid V_k)$. This is called *inverse* programming and require an inversion of each submodel to build the joint distribution. We will address this fusion scheme in section III and show that the resulting distribution is no longer the product of each underlying distribution.

Section IV will thus present a new way of specifying a model using a consistency variable that will allow the fusion to be written as a product even in the case of command fusion. Finally two robotic implementations of such a scheme will be detailed in section V.

All along this paper, these conventions will be used:

- V: for a variable;
- $\vec{V}$: for any set of variable $\{V_k\}$, $\vec{V} = V_1 \dots V_n$;
- v: for any value of the variable V.

Furthermore, we will use variables with the following semantic:

- A: *Opinion* of some expert, or fusion of opinions about a problem (the pose of a robot for instance, or some motor command);
- D or D_k: Measured data;
- π_f or π_k: *A priori* knowledge.

Finally, we will consider a probabilistic program as formalized in [5] in order to explicit every assumption we make. Such a program is composed of:

- the list of *relevant variables*;

- a *decomposition* of the joint distribution over these variables;
- the *parametrical form* of each factor of this product;
- the *identification* of the parameters of these parametrical forms;
- a *question* in the form of probability distribution inferred from the joint distribution.

II. Bayesian fusion with "direct models"

In order to be in good conditions for the following of this paper, it seems necessary to understand the classical bayesian fusion mechanism, as presented in [5].

First, we assume that we know how to express $P(D_k \mid A\,\pi_k)$, π_k being the set of *a priori* knowledge used by the programmer to describe the model k linking D_k and A. Then we are interested in $P(A \mid D_1 \ldots D_n\,\pi_f)$. In the context of mobile robotic, $P(D_k \mid A\,\pi_k)$ could be a sensor model, which, given the robot pose A, will predict a probability distribution over the possible observation D_k.

Using a modular programming paradigm, we start by defining sub-models which express *a priori* knowledge π_k. Practically, for each k, we use Bayes' rule to give the following joint distribution:

$$P(A\,D_k \mid \pi_k) = P(A \mid \pi_k)P(D_k \mid A\,\pi_k) \quad (1)$$

Then, we assume that we have no prior about A, so we have $P(A \mid \pi_k)$ uniform. In this case, we have, by direct application of Bayes' rule:

$$\begin{aligned} & P([A = a] \mid [D_k = d_k]\,\pi_k) \quad (2) \\ = \; & \frac{P([A = a] \mid \pi_k)P([D_k = d_k] \mid [A = a]\,\pi_k)}{P([D_k = d_k] \mid \pi_k)} \end{aligned}$$

Since we chose $P(A \mid \pi_k)$ uniform, and as $P([D_k = d_k] \mid \pi_k)$ does not depend on a, we get the following property:

$$\begin{aligned} & \exists c_k, \forall a, P([D_k = d_k] \mid [A = a]\,\pi_k) \quad (3) \\ = \; & c_k P([A = a] \mid [D_k = d_k]\,\pi_k) \end{aligned}$$

In order to shorten the notations, we will write the preceding equation as follows: $P(A \mid D_k\,\pi_k) \propto P(D_k \mid A\pi_k)$.

Using Bayes' rule and assuming the measured data independent, we can now express the complete joint distribution of the system:

$$P(A\,\vec{D} \mid \pi_f) = P(A \mid \pi_f) \prod_{k=1}^{n} P(D_k \mid A\,\pi_f) \quad (4)$$

In order to stay consistent with the sub-models, we choose to define $P(A \mid \pi_f)$ as a uniform distribution, and we set $P(D_k \mid A\,\pi_f) = P(D_k \mid A\,\pi_k)$.

We now come back to the distribution we were interested in:

$$\begin{aligned} & P([A = a] \mid [\vec{D} = \vec{d}]\,\pi_f) \quad (5) \\ = \; & \frac{P([A = a]\,[\vec{D} = \vec{d}] \mid \pi_f)}{P([\vec{D} = \vec{d}] \mid \pi_f)} \\ = \; & \frac{P([A=a] \mid \pi_f) \prod_{k=1}^{n} P([D_k = d_k] \mid [A=a]\,\pi_f)}{P([\vec{D} = \vec{d}] \mid \pi_f)} \end{aligned}$$

As $P([\vec{D} = \vec{d}] \mid \pi_f)$ does not depend on a, the proportionality which was true for the sub-models still holds for the complete model:

$$\begin{aligned} & \exists \kappa, \forall a, P([A = a] \mid [\vec{D} = \vec{d}]\,\pi_f) \quad (6) \\ = \; & \kappa \prod_{k=1}^{n} P([D_k = d_k] \mid [A = a]\,\pi_k) \end{aligned}$$

Finally, by substituting equation 4, we get

$$\begin{aligned} & \exists K = \kappa c_1 \ldots c_n, \forall a, \\ & P([A = a] \mid [\vec{D} = \vec{d}]\,\pi_f) \quad (7) \\ = \; & K \prod_{k=1}^{n} P([A = a] \mid [D_k = d_k]\,\pi_k) \end{aligned}$$

The probability distribution on the opinion A, resulting from the observation of n pieces of data d_k, is proportional to the product of probability distributions resulting from the individual observation of each data.

This result is intuitively satisfying for at least two reasons:

- First, if only one expert is available, the result of the fusion of his unique opinion is indeed his opinion. So the fusion process does not introduce additional knowledge.
- Second, if the dimension of A is greater than 1, and if each expert brings informations about one dimension of A, the projection of the fusion result on one dimension will be the opinion of the corresponding expert. This property is well illustrated in [2].

III. Bayesian fusion with "inverse models"

For instance, in a context of localization, we can usually predict sensor output given the position, and we are interested in the position. The joint distribution can be written using Bayes' rule: $P(\text{Pose Sensor1 Sensor2}) = P(\text{Pose})P(\text{Sensor1} \mid \text{Pose})P(\text{Sensor2} \mid \text{Pose})$. This is a direct model since we can build it directly from what we can express. On the other hand, in a context of command fusion, we can express a command distribution given the sensor reading $P(\text{Command} \mid \text{Sensor1})$, and we are interested in the command distribution $P(\text{Command} \mid \text{Sensor1 Sensor2})$. Unfortunately, there is no way to build a joint distribution $P(\text{Command Sensor1 Sensor2})$ using Bayes' rule only once. So we will have to build several sub-models and to inverse them.

Formally, let us assume that we know how to express $P(A \mid D_k\,\pi_k)$ instead of $P(D_k \mid A\,\pi_k)$, and that we still are interested in the evaluation of $P(A \mid \vec{D}\,\pi_f)$.

As before, using a modular probabilistic programming paradigm, we start by specifying sub-models which express the π_k. First:

$$P(A\,D_k \mid \pi_k) = P(D_k \mid \pi_k)P(A \mid D_k\,\pi_k) \quad (8)$$

with $P(D_k \mid \pi_k)$ uniform. From this sub-models, using Bayes' rule, we can express $P(D_k \mid A\,\pi_k)$.

Then, we can introduce this expression in a global model:

$$P(A\,\vec{D} \mid \pi_f) = P(A \mid \pi_f) \prod_k P(D_k \mid A\,\pi_f) \quad (9)$$

where we let $P(D_k \mid A\,\pi_f) = P(D_k \mid A\,\pi_k)$.

Then, no matter what $P(A \mid \pi_f)$ is, we get

$$\begin{aligned} & P(A \mid \vec{D}\,\pi_f) \quad (10) \\ \propto \quad & P(A \mid \pi_f) \prod P(D_k \mid A\,\pi_k) \\ \propto \quad & P(A \mid \pi_f) \prod \frac{P(D_k \mid \pi_k) P(A \mid D_k\,\pi_k)}{P(A \mid \pi_k)} \end{aligned}$$

In the general case ($P(A \mid \pi_f)$ unspecified, uniform...), this leads to

$$P(A \mid \vec{D}\,\pi_f) \not\propto \prod P(A \mid D_k\,\pi_k) \quad (11)$$

Thus, this result does not correspond to the intuition of the bayesian fusion process we got in section II.

Nevertheless, it exists a way to come back to the proportionality: we just have to specify $P(A \mid \pi_f)$ such that

$$\frac{P(A \mid \pi_f)}{\prod P(A \mid \pi_k)} = \text{cste} \quad (12)$$

Practically, this corresponds to

$$\begin{aligned} P(A \mid \pi_f) \quad & \propto \quad \prod P(A \mid \pi_k) \quad (13) \\ & \propto \quad \prod \sum_{D_k} P(A \mid D_k\,\pi_k) P(D_k \mid \pi_k) \end{aligned}$$

Using this probability distribution, we effectively obtain an intuitive fusion process, but the understanding of the "physical meaning" of $P(A \mid \pi_f)$ becomes rather challenging.

IV. Bayesian fusion with diagnosis

A. Definitions

In this section, we introduce a new variable:

- $\mathbb{I}$ or $\mathbb{I}_k$: boolean variable which **I**ndicates if the opinion A is *consistent* with the measure D_k.

We now express the following sub-model:

$$P(A\,D_k\,\mathbb{I}_k \mid \pi_k) = P(A \mid \pi_k) P(D_k \mid \pi_k) P(\mathbb{I}_k \mid A\,D_k\,\pi_k) \quad (14)$$

with A and D_k independent and uniform[1]. The conditional distribution over $\mathbb{I}_k$ is to be specified by the programmer. For instance, he may choose:

$$P([\mathbb{I}_k = 1] \mid A\,D_k\,\pi_k) = \exp(-\frac{1}{2}\left(\frac{A - D_k}{\sigma}\right)^2) \quad (15)$$

The main interest of this model is due to the fact that it provides us with a way to express

$$P(A \mid \vec{D}\vec{\mathbb{I}}\pi_f) \propto \prod_k P(A \mid D_k\,\mathbb{I}_k\,\pi_k) \quad (16)$$

This is illustrated in figure 1 that compares the results of inverse fusion and fusion with diagnosis. It shows that in some cases, inverse fusion leads to a counterintuitive response whereas the product sticks to an expected result.

[1] This is true when $\mathbb{I}$ is not considered.

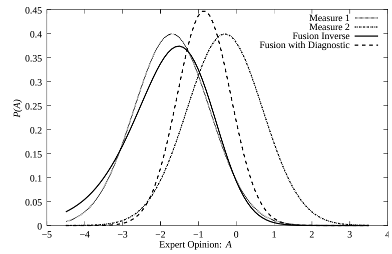


Fig. 1. Comparison of fusion processes

B. Proof of equation 16

First, we can write Bayes' rule as follows:

$$\begin{aligned} & P([A = a] \mid [\vec{D} = \vec{d}][\vec{\mathbb{I}} = \vec{\imath}]\,\pi_f) \quad (17) \\ = \quad & \frac{P([A = a]\,[\vec{D} = \vec{d}] \mid \pi_f)}{P([\vec{D} = \vec{d}]\,[\vec{\mathbb{I}} = \vec{\imath}] \mid \pi_f)} \times \\ & P([\vec{\mathbb{I}} = \vec{\imath}] \mid [A = a]\,[\vec{D} = \vec{d}]\,\pi_f) \end{aligned}$$

But, due to the hypothesis contained in the π_k, A and $\vec{D}$ are independent and uniformly distributed, hence $P([A = a]\,[\vec{D} = \vec{d}] \mid \pi_f)$ does not depend on a and we can write:

$$\begin{aligned} & P([A = a] \mid [\vec{D} = \vec{d}][\vec{\mathbb{I}} = \vec{\imath}]\,\pi_f) \quad (18) \\ \propto \quad & P([\vec{\mathbb{I}} = \vec{\imath}] \mid [A = a]\,[\vec{D} = \vec{d}]\,\pi_f) \end{aligned}$$

Then we will assume that all the sensor models are independent:

$$\begin{aligned} & P([A = a] \mid [\vec{D} = \vec{d}]\,[\vec{\mathbb{I}} = \vec{\imath}]\,\pi_f) \quad (19) \\ \propto \quad & P([\vec{\mathbb{I}} = \vec{\imath}] \mid [A = a]\,[\vec{D} = \vec{d}]\,\pi_f) \\ \propto \quad & \prod_k P([\mathbb{I}_k = i_k] \mid [A = a]\,[D_k = d_k]\,\pi_k) \end{aligned}$$

Another application of Bayes' rule leads to:

$$\begin{aligned} & P([A = a] \mid [\vec{D} = \vec{d}]\,[\vec{\mathbb{I}} = \vec{\imath}]\,\pi_f) \quad (20) \\ \propto \quad & \prod_k \Bigg[\frac{P([D_k = d_k]\,[\mathbb{I}_k = i_k] \mid \pi_k)}{P([A = a]\,[D_k = d_k] \mid \pi_k)} \\ & \quad \times P([A = a] \mid [D_k = d_k]\,[\mathbb{I}_k = i_k]\,\pi_k) \Bigg] \end{aligned}$$

Again, as A and D_k are independent and uniform, $P([A = a]\,[D_k = d_k] \mid \pi_k)$ does not depend on a. This then lead to equation 16. Therefore we can write:

$$\begin{aligned} & P([A = a] \mid [\vec{D} = \vec{d}][\vec{\mathbb{I}} = \vec{\imath}]\,\pi_f)] \quad (21) \\ \propto \quad & \prod_k P([A = a] \mid [D_k = d_k]\,[\mathbb{I}_k = i_k]\,\pi_k) \end{aligned}$$

C. Expression of $P(\mathbb{I}_k \mid A\, D_k\, \pi_k)$

It is sometimes easier to express something which looks like $P(A \mid D_k \pi_k)$ or $P(D_k \mid A \pi_k)$ than directly $P(\mathbb{I}_k \mid A\, D_k\, \pi_k)$. How can we deduce $P(\mathbb{I}_k \mid A\, D_k\, \pi_k)$ from this?

Actually, variable $\mathbb{I}_k$ provides us with more knowledge than the genuine knowledge of $P(A \mid D_k\, \pi_k)$. It gives us a way to describe the output of a sensor which does not work, or an inconsistent behavior given some measures. In order to reckon $P(\mathbb{I}_k \mid A\, D_k\, \pi_k)$, we have to be able to express $P(A \mid D_k\, [\mathbb{I}_k = 1]\, \pi_k)$ ("usually" written as $P(A \mid D)$) **and** $P(A \mid D_k\, [\mathbb{I}_k = 0]\, \pi_k)$.

In this case, due to equation 20, it exists c_0 and c_1 such that

$$
\begin{aligned}
&P([\mathbb{I}_k = 0] \mid [A = a]\, D_k\, \pi_k) \\
&\quad = c_0\, P([A = a] \mid [\mathbb{I}_k = 0]\, D_k\, \pi_k) \\
&P([\mathbb{I}_k = 1] \mid [A = a]\, D_k\, \pi_k) \\
&\quad = c_1\, P([A = a] \mid [\mathbb{I}_k = 1]\, D_k\, \pi_k) \\
&P([\mathbb{I}_k = 0] \mid [A = a]\, D_k\, \pi_k) \\
&\quad = 1 - P([\mathbb{I}_k = 1] \mid [A = a]\, D_k\, \pi_k)
\end{aligned}
$$

So,

$$c_0 + c_1 = \frac{P([\mathbb{I}_k = 0]\, D_k)}{P(A)P(D_k)} + \frac{P([\mathbb{I}_k = 1]\, D_k)}{P(A)P(D_k)} = \frac{1}{P(A)}$$

Thus, defining

$$
\begin{aligned}
P_0 &= P([A = a] \mid [\mathbb{I}_k = 0]\, D_k\, \pi_k) \\
P_1 &= P([A = a] \mid [\mathbb{I}_k = 1]\, D_k\, \pi_k) \\
U_A &= P([A = a] \mid \pi_k) \leftarrow \text{Uniform value on } A
\end{aligned}
$$

we get a linear system in c_0 and c_1 from which we find:

$$
\begin{aligned}
P([\mathbb{I}_k = 0] \mid [A = a]\, D_k\, \pi_k) &= \frac{P_0}{U_A} \frac{U_A - P_1}{P_0 - P_1} \\
P([\mathbb{I}_k = 1] \mid [A = a]\, D_k\, \pi_k) &= \frac{P_1}{U_A} \frac{U_A - P_0}{P_1 - P_0}
\end{aligned}
$$

First note: The preceding results is singular when $P_0(a) = P_1(a)$. A way to remove the singularity is to set $P_0(a) = P_1(a) = U_A$ in these cases. $P(\mathbb{I} \mid [A = a]\, D_k)$ can then be given a value on this point, but this value does not depend anymore on the relative value of $P_0(a)$ and $P_1(a)$, but rather on their derivatives. To give a sense to these values becomes rather complicated.

Second note: Knowing P_1, how can P_0 be defined so that $P(\mathbb{I}_k \mid A D_k)$ be continuous? It may exist numerous ways to reach this goal, but we can show that in particular, for any C greater than $\max(P_1)$, $P_0 \propto C - P_1$ satisfy our conditions. If C is lesser than $\max(P_1)$, then $C - P_1$ is sometimes negative, which makes it incompatible with a probability distribution.

So we can *easily* express $P(\mathbb{I}_k \mid A\, D_k\, \pi_k)$ from the knowledge of $P(A \mid [\mathbb{I}_k = 1]\, D_k\, \pi_k)$. Since all these expressions are symmetric in A and D_k, the knowledge of $P(D_k \mid A\, [\mathbb{I}_k = 1]\, \pi_k)$ would be satisfying all the same.

D. Properties

Generally, we are interested in the case where we assume that our experts are competent, and so $\vec{\mathbb{I}} = \vec{1}$. Hence, when we compute $P(A \mid \vec{D}) \propto \prod_k P(A \mid D_k)$, we are implicitly in the context of equation 16, with $\vec{\mathbb{I}} = \vec{1}$.

Another interesting point with this form of bayesian fusion appears when we use only one expert, the resulting opinion is the expert opinion. So the fusion does not introduce some additional knowledge. In the general case, we have seen that this cannot be guaranteed without introducing the $\mathbb{I}$ variable. As far as we understand it, this "failure" of the classical fusion might be due to the fact that it does not describe the output of a failing sensor, i.e. $P(A \mid D_k\, [\mathbb{I}_k = 0])$.

V. Applications

A. Obstacle avoidance

1) Situation: The robot we use is a car-like robot. It can be commanded through a speed V and a steering angle Φ, and it is equipped with 8 sensors. These sensors measure the distance to the nearest object in some fixed angular sector (see figure 2). We will call $D_k, k = 1 \ldots 8$ the probabilistic variables corresponding to these measures.

Besides, we will assume that this robot is commanded by some high-level system (trajectory following for instance) which provides him with a pair of desired commands (V_d, Φ_d).

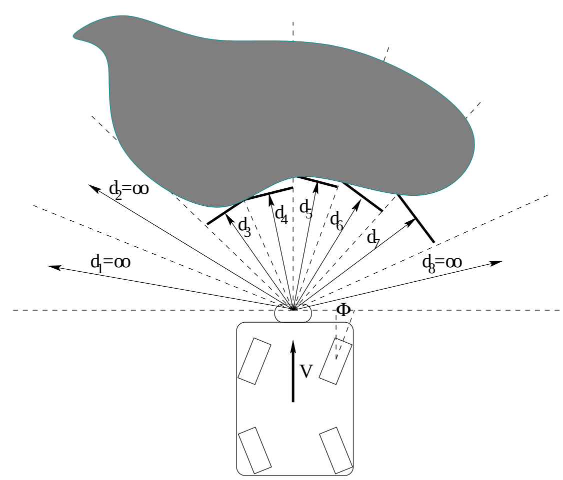


Fig. 2. Obstacle avoidance: situation

Our goal is to find commands to apply to the robot, guarantying the vehicle security while following the desired command as much as possible.

2) Sub-models definition: Before to define our sub-models, it seems necessary to identify, in the preceding paragraph, the variables which correspond to A and D_k.

- The opinion on which each sub-model will have to express itself is the vehicle command. So $A \equiv U = (V, \Phi)$.
- The data which will be used by the sub-models are the eight distances $D_1, \ldots, D_8$ and the desired command $U_d = (V_d, \Phi_d)$.

In each sub-model, the variable $\mathbb{I}_k$ will describe the *compatibility* between a model and a given measure. In this case, we define compatibility in term of agreement with the desired commands or in term of security guarantee.

Two types of sub-model will be used:

- **Desired Command Following**: Figure 3.
- **Elementary Obstacle Avoidance**: Figure 4.

Program / Description / Specification:

Relevant Variables:
$U \equiv (V, \Phi)$: Robot commands
$U_d \equiv (V_d, \Phi_d)$: Desired commands
$\mathbb{I}_{cf}$: Compatibility between U and U_d

Decomposition:
$P(U\, U_d\, \mathbb{I}_{cf} \mid \pi_{cf}) =$
$P(U \mid \pi_{cf}) P(U_d \mid \pi_{cf}) P(\mathbb{I}_{cf} \mid U\, U_d\, \pi_{cf})$

Parametric Forms:
$P(U \mid \pi_{cf})$uniform
$P(U_d \mid \pi_{cf})$uniform
$P(\mathbb{I}_{cf} \mid U\, U_d\, \pi_{cf}) = e^{-\frac{1}{2}(U-U_d)^T P^{-1} (U-U_d)}$

Identification:
P fixed *a priori* .

Question:
$P(U \mid [U_d = u_d]\, [\mathbb{I}_{cf} = 1]\, \pi_{cf})$

Fig. 3: Desired Command Following

Program / Description / Specification:

Relevant Variables:
$U \equiv (V, \Phi)$: Robot commands
D_k : Distance measured by sensor k
$\mathbb{I}_k$: compatibility between U and D_k

Decomposition:
$P(U\, D_k\, \mathbb{I}_k \mid \pi_k) =$
$P(U \mid \pi_k) P(D_k \mid \pi_k) P(\mathbb{I}_k \mid U\, D_k\, \pi_k)$

Parametric Forms:
$P(U \mid \pi_k)$uniform
$P(D_k \mid \pi_k)$uniform
$P(\mathbb{I}_k \mid U\, D_k\, \pi_k) = \frac{1}{1+\exp(-4\beta(V - V_{max}(\Phi, D_k)))}$

Identification:
β fixed *a priori*
$V_{max}()$ pre-calculated in simulation

Question:
$P(U \mid [D_k = d_k]\, [\mathbb{I}_k = 1]\, \pi_k)$

Fig. 4: Elementary Obstacle Avoidance k

Once these sub-models defined, a bayesian program such as the one presented in figure 5 provides us with a way to answer the question

$$P(V\, \Phi \mid D_1 \ldots D_8\, V_c\, \Phi_c\, [\vec{\mathbb{I}} = \vec{1}]\, \pi_f)$$

Program / Description / Specification:

Relevant Variables:
$U \equiv (V, \Phi)$: Robot commands
$\vec{D} = U_d D_1 \ldots D_8$: Obstacle avoidance data
$\vec{\mathbb{I}}$: Indicators from sec. IV

Decomposition:
$P(U\, \vec{D}\, \vec{\mathbb{I}} \mid \pi_f) =$
$P(U \mid \pi_f) P(\vec{D} \mid \pi_f) \prod_k P(\mathbb{I}_k \mid U\, D_k\, \pi_f)$

Parametric Forms:
$P(U \mid \pi_f)$uniform
$P(D_k \mid \pi_f)$uniform
$P(\mathbb{I}_k \mid U\, D_k\, \pi_f) = P(\mathbb{I}_k \mid U\, D_k\, \pi_k)$

Identification:
None

Question:
$P(U \mid [\vec{D} = \vec{d}]\, [\vec{\mathbb{I}} = \vec{1}] \pi_f)$

Fig. 5: Obstacle Avoidance

3) Results: We want to compute $P(U \mid U_d\ D_1 \ldots D_8\ [\vec{\mathbb{I}} = \vec{1}]\ \pi_f)$ with the data presented in table I. Table II shows how the sub-models gradually act on the resulting distribution $P(V\ \Phi \mid V_d\ \Phi_d\ D_1 \ldots D_8\, [\vec{\mathbb{I}} = \vec{1}]\ \pi_f)$. In this table, each cell shows one expert's opinion (left) and its accumulated influence on the global model (right). In our current implementation, evaluating the fusion distribution given desired commands and measured distances took about $40\mu s$ (1GHz PC).

Variable	Value	Variable	Value
V_d	$1.5\, m/s$	Φ_d	$0.2\, rad$
D_1	$4.5\, m$	D_2	$4.5\, m$
D_3	$2.5\, m$	D_4	$4.8\, m$
D_5	$3.7\, m$	D_6	$3.0\, m$
D_7	$5.0\, m$	D_8	$5.0\, m$

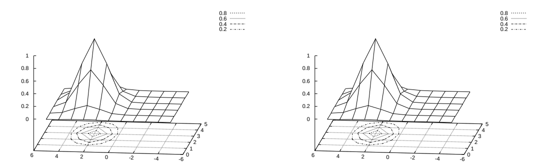
TABLE I

TEST DATA FOR OBSTACLE AVOIDANCE SYSTEM

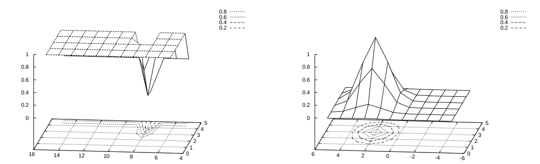
B. Localization

1) Situation: We consider now the case of a mobile robot whose configuration is $C = [x, y, \theta]$. This robot moves in an environment where some landmarks can be found. The position of one such landmark will be noted $L_j = [x_j, y_j]$, and will be assumed known. Furthermore, a sensor is installed on this robot in such a way that its pose in the global frame is identical to the robot's pose. When a landmark is observed by our sensor, a pair of measure (distance, bearing) is returned. We will call such a measure an observation $O_k = [d_k, \alpha_k]$ of the landmark. From the measure only, we cannot identify which landmark has been observed.

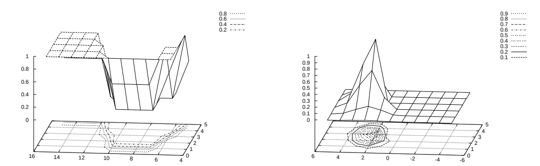
In these condition our goal is to compute a probability distribution over the robot configurations, knowing a set of observations $\vec{O}$. We will show that this goal can be efficiently achieved using fusion with diagnosis.



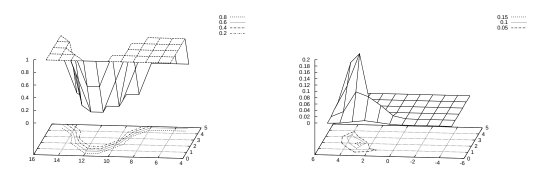
Desired Command Following



Obstacle avoidance knowing D_2



Obstacle avoidance knowing D_4



Obstacle avoidance knowing D_6

TABLE II

PROGRESSIVE COMPUTATION OF $P(V\,\Phi \mid V_d\,\Phi_d\,D_1 \ldots D_8\,\pi_f)$.

2) Models: We first define a sensor model which evaluate the compatibility $I\!I_k$ of a given observation with a given configuration, knowing the landmarks positions. This model uses an observation predictor: $\tilde{O}_k = h(C, L_j)$.

$$P(M_k \mid C\,O_k\,L_j\,\pi_k) = \exp(\frac{1}{2}(O_k - h(C,L_j))^T P^{-1}(O_k - h(C,L_j))) \quad (22)$$

Yet, this model assumes that we know which landmark is observed. This assumption is false in our case. We could use some complex data association scheme, as available in the literature, but we would rather use a probabilistic approach to this problem. So we introduce another variable W_k (**W**hich), whose value indicates which landmark is observed for a given O_k. So we can express the following model:

$$P(I\!I_k \mid C\,O_k\,W_k\,\vec{L}\,\pi_k) = \exp(\frac{1}{2}(O_k - h(C,L_{W_k}))^T P^{-1}(O_k - h(C,L_{W_k}))) \quad (23)$$

From this model, we will build a set of bayesian programs: as many sub-models (see figure 6) as observations, and a global model, which makes a bayesian fusion of these models (see figure 7).

- Program
  - Description
    - Specification
      - Relevant Variables:
        - $C = (X, Y, \Theta)$: Robot configuration
        - $O_k = (D_k, A_k)$: Observation k
        - $\vec{L}$: Observable landmarks
        - $W_k \in [1..n]$: Landmark observed in O_k
        - $I\!I_k \in \{0, 1\}$: Compatibility between O_k et C
      - Decomposition:
        $P(M_k\,C\,O_k\,W_k\,\mathcal{L} \mid \pi_k) =$
        $P(C \mid \pi_k)P(\vec{L} \mid \pi_k)P(O_k \mid \pi_k)$
        $P(W_k \mid \pi_k)P(I\!I_k \mid C\,\vec{L}\,O_k\,W_k\,\pi_k)$
      - Parametric Forms:
        - $P(C \mid \pi_k)$uniform
        - $P(O_k \mid \pi_k)$uniform
        - $P(W_k \mid \pi_k)$uniform
        - $P(\vec{L} \mid \pi_k)$uniform
        - $P(I\!I_k \mid C\,\vec{L}\,O_k\,W_k\,\pi_k)$see equation 24
    - Identification:
      PFixed *a priori*
  - Question:
    $P(I\!I_k \mid [O_k = (d_k, \alpha_k)]\,[C = (x, y, \theta)][\vec{L} = \vec{l}]\pi_k)$

Fig. 6: Sensor model for observation k

- Program
  - Description
    - Specification
      - Relevant Variables:
        - $C = (X, Y, \Theta)$: Robot configuration
        - $\vec{O}$: Observation set
        - $\vec{L}$: Landmark set
        - $\vec{I\!I}$: Compatibilities between $\vec{O}$ and C
      - Decomposition:
        $P(C\,\vec{O}\,\vec{L}\,\vec{I\!I} \mid \pi_f) =$
        $P(C \mid \pi_f)P(\vec{L} \mid \pi_f)P(\vec{O} \mid \pi_f)$
        $\prod_k P(I\!I_k \mid C\,\mathcal{L}\,O_k\,\pi_f)$
      - Parametric Forms:
        - $P(C \mid \pi_f)$uniform
        - $P(\vec{O} \mid \pi_f)$uniform
        - $P(\vec{L} \mid \pi_f)$uniform
        - $P(I\!I_k \mid C\,\vec{L}\,O_k\,\pi_f) = P(I\!I_k \mid C\,\vec{L}\,O_k\,\pi_k)$
          Question to sub-model j (see figure 6)
    - Identification:
      None
  - Question:
    $P(C \mid [\vec{I} = \vec{1}]\,[\vec{O} = \vec{o}][\vec{L} = \vec{l}]\pi_f)$

Fig. 7: Localization program

3) Outliers management: As seen at the beginning of this article, the final expression of $P(C \mid \ldots)$ will be proportional to a product of $P(I\!I_k \mid C\,O_k\,W_k\,\vec{L}\,\pi_k)$. If observation O_k is an outlier[2], $P(I\!I_k \mid C \ldots)$ will be, in general, very small if C stands for the true position of the robot. So, when numerous outliers are present, correct position compatibility will be set to zero, due to the only presence of these measure.

[2]Observation which is not the result of the observation of some known landmark

A simple solution to this problem is to add a special value to variable W_k: when it is zero, observation O_k is assumed to be an outlier, and $P(I\!I_k \mid C O_k [W_k = 0] \vec{L} \pi_k)$ is set uniform. Another solution would consists in adding a new variable F_k (**F**alse) which indicates whether O_k is an outlier or not. Identification of parametric forms becomes then more complicates, but semantic is more satisfying.

a) Is there a reason to choose this model?: In the case of localization, it would be completely possible to use a classical bayesian fusion instead of fusion with diagnosis. The main interest of this method is the computational cost. Actually, we may note that, for each $x, y, \theta, d_k, \alpha_k, x_0, y_0$, we have:

$$\begin{aligned} & P(I\!I_k \mid [C = (x, y, \theta)] \, [O_k = (d_k, \alpha_k)] \\ & \quad [L_0 = (x_0, y_0)] \, [W_k = 0] \pi_k) \\ = \; & P(I\!I_k \mid [C = (x - x_0, y - y_0, \theta - \alpha_k)] \\ & \quad [O_k = (d_k, 0)][L_0 = (0, 0)][W_k = 0] \pi_k) \end{aligned} \tag{24}$$

So, by tabulating $P([I\!I_k = 1] \mid [C = (x, y, \theta)][O_k = (d_k, 0)][L_0 = (0, 0)][W_k = 0] \pi_k)$ we can compute $P(I\!I_k \mid \ldots)$ without computing neither a transcendental function nor the observation model h.

It is possible to make a step further in the research for computational efficiency by precalculating directly $P(C \mid [I\!I_k = 1] \, [O_k = (d, 0)] \, [L_0 = (0, 0)] \, [W_k = 0] \, \pi_k)$. In this case, due to equation 20, we have:

$$\begin{aligned} f(C) &= \prod_k \sum_{W_k} P(C \mid O_k \, \vec{L} \, [I\!I_k = 1] \, W_k \, \pi_k) P(W_k \mid \pi_k) \\ &\propto P(C \mid \vec{O} \, \vec{L} \, [\vec{I\!I} = \vec{1}] \, \pi_f), \end{aligned} \tag{25}$$

Thus it is possible to compute $P(C \mid \ldots)$ locally without taking care of normalization. For instance, if an estimation of the robot current pose is available, we can only compute $f(C)$ in some neighborhood of this pose. Then, we might search a peak of $f(C)$ in this neighborhood. Due to the proportionality this peak of $f(C)$ will also be a peak of $P(C \mid \ldots)$.

Furthermore, using the above equation we can see that the fusion distribution can be evaluated by evaluating exactly $sizeof(\vec{O}) \times sizeof(\vec{L})$ distributions.

4) Results: Graphs in table III give a discretized estimation of some probability distribution $P(C \mid [I\!I = \vec{1}] \, \vec{L} \, \ldots)$. The arrows orientation correspond to the variable θ, and their length is proportional to the probability of the configuration which corresponds to their base position. To add some readability to this complex graphs, lines of maximum likelihood were superimposed to the plot. A peak of the likelihood clearly appears around the correct configuration (rounded in plain lines). Two lesser maxima (rounded in dashed lines) are also present, expressing the possible positions, should one observation be an outlier.

Note that in this example, the building of the probability distribution corresponding to the potential matching of one observation with one landmark took about $0.5ms$ (1GHz PC). So the complete probability distribution evaluation took about $4.5ms$.

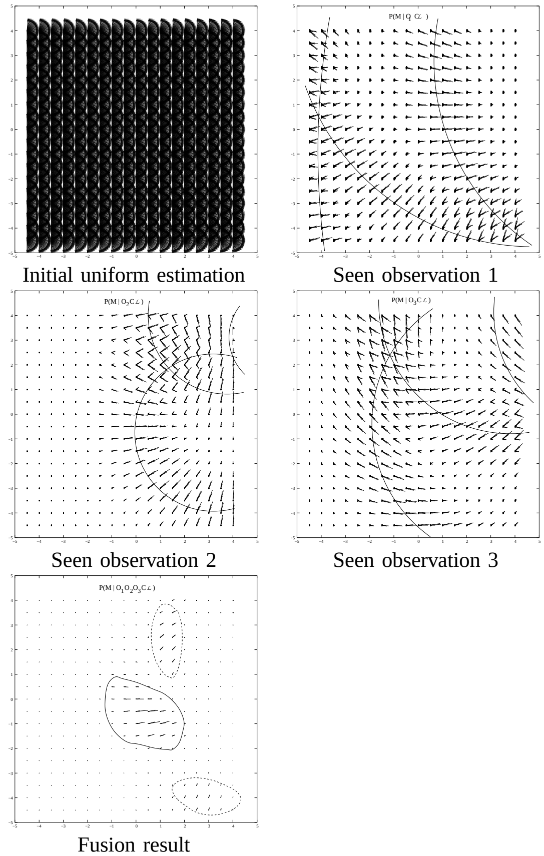


TABLE III

FUSION WITH DIAGNOSIS IN LOCALIZATION

VI. CONCLUSION

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

We put the stress on the fact that bayesian fusion cannot be always expressed as a product of probability distributions. Specifically, we found that, in such case as command fusion where the fusion should semantically result in a product, we have to use specific descriptions. The models we use should express rather a *consistency* between variables ($P(I\!I \mid A \, B \, \pi)$) than an expectation ($P(A \mid B \, \pi)$).

We have also shown that, using fusion with diagnosis instead of classical fusion could lead to computationally more efficient solutions.

The main advantages of our approach is that it provides us with a new way to perform data fusion in the context of Bayesian Programming. So we keep the advantages of Bayesian Programming, namely: a) a clear and well-defined mathematical background, b) a generic and uniform way to formulate problems. And we add the following specific advantages: a) a model expression which is symmetric in input and output variables, b) a fusion scheme which can always be expressed in term of a product of probability distributions, c) a mathematical soundness for "currently running experiments" expressed as products of probability distributions.

As for future works, we will use our increased expression abilities with Bayesian Programming in order to build a complete modular robotic application based on bayesian inference: obstacle avoidance, localization, trajectory following, sensor servoing...

VII. Acknowledgments

This work was supported by the European Project BIBA and the French ministry of research.

VIII. References

[1] G. Cooper. The computational complexity of probabilistic inference using bayesian belief network. *Artificial Intelligence*, 42(2-3), 1990.

[2] C. Coué, Th. Fraichard, P. Bessière, and E. Mazer. Multi-sensor data fusion using bayesian programming: an automotive application. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Lausanne (CH), September-October 2002.

[3] D. Hall and J. Llinas. An introduction to multisensor data fusion. *IEEE Proceedings*, 85(1), January 1997.

[4] E. T. Jaynes. Probability theory: the logic of science. Unprinted book, available on-line at `http://bayes.wustl.edu/etj/prob.html`, 1995.

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