Expressing Bayesian Fusion as a Product of Distributions: Application to Randomized Hough Transform

Cédric Pradalier, Francis Colas and Pierre Bessière

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

August 1, 2003

---

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

Abstract — Data fusion is a common issue of mobile robotics, computer assisted medical diagnosis or behavioral control of simulated character for instance. However data sources are often noisy, opinion for experts are not known with absolute precision, and motor commands do not act in the same exact manner on the environment. In these cases, classic logic fails to manage efficiently the fusion process. Confronting different knowledge in an uncertain environment can therefore be adequately formalized in the bayesian framework.

Besides, bayesian fusion can be expensive in terms of memory usage and processing time. This paper precisely aims at expressing any bayesian fusion process as a product of probability distributions in order to reduce its complexity. We first study both direct and inverse fusion schemes. We show that contrary to direct models, inverse local models need a specific prior in order to allow the fusion to be computed as a product. We therefore propose to add a consistency variable to each local model and we show that these additional variables allow the use of a product of the local distributions in order to compute the global probability distribution over the fused variable. Finally, we take the example of the Randomized Hough Transform. We rewrite it in the bayesian framework, considering that it is a fusion process to extract lines from couples of dots in a picture. As expected, we can find back the expression of the Randomized Hough Transform from the literature with the appropriate assumptions.

Keywords — Bayesian programming, Data fusion, Hough Transform

Expressing Bayesian Fusion as a Product of Distributions: Application to Randomized Hough Transform

Cédric Pradalier*, Francis Colas* and Pierre Bessière*

*655 av. de l'Europe
38334 Montbonnot, St Ismier Cedex FRANCE
firstname.name@inrialpes.fr

Abstract. Data fusion is a common issue of mobile robotics, computer assisted medical diagnosis or behavioral control of simulated character for instance. However data sources are often noisy, opinion for experts are not known with absolute precision, and motor commands do not act in the same exact manner on the environment. In these cases, classic logic fails to manage efficiently the fusion process. Confronting different knowledge in an uncertain environment can therefore be adequately formalized in the bayesian framework.

Besides, bayesian fusion can be expensive in terms of memory usage and processing time. This paper precisely aims at expressing any bayesian fusion process as a product of probability distributions in order to reduce its complexity. We first study both direct and inverse fusion schemes. We show that contrary to direct models, inverse local models need a specific prior in order to allow the fusion to be computed as a product. We therefore propose to add a consistency variable to each local model and we show that these additional variables allow the use of a product of the local distributions in order to compute the global probability distribution over the fused variable. Finally, we take the example of the Randomized Hough Transform. We rewrite it in the bayesian framework, considering that it is a fusion process to extract lines from couples of dots in a picture. As expected, we can find back the expression of the Randomized Hough Transform from the literature with the appropriate assumptions.

Bayesian programming, Data fusion, Command fusion

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 applica-

tions.

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 issues 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[6] and Coue[2]. The CyberMove project is precisely involved in robotics and in particular in probabilistic programming. This paradigm is applied for car-like robots in the framework of bayesian theory as depicted in [5]. 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 $V_1 \ldots V_n$: $P(V \mid V_1 \ldots V_n)$. In the case of multi-sensor fusion, V could stand for the pose of a robot and $V_1 \ldots V_n$, 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 infer directly the fused distribution. Additionally we will show in section that $P(V \mid V_1 \ldots V_n)$ is proportional 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 sub-model to build the joint distribution. We will address this fusion scheme in section and show that the resulting distribution is no longer the product of each underlying distribution.

Section 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 in section , we will take the example of the Randomized Hough Transform [7] and rewrite it with this fusion scheme.

All along this paper, these conventions will be used:

- V: for a variable;
- $\vec{V}$: for any set of variable $\{V_k\}$, $\vec{V} = V_1 \ldots V_n$;
- 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 [6] in order to make 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;

1

• a *question* in the form of probability distribution inferred from the joint distribution.

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 [6].

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 robotics, $P(D_k \mid A\pi_k)$ 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:

$$P(AD_k \mid \pi_k) = P(A \mid \pi_k)P(D_k \mid A\pi_k) \tag{1}$$

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

$$P([A = a] \mid [D_k = d_k]\pi_k) \tag{2}$$
$$= \frac{P([A = a] \mid \pi_k)P([D_k = d_k] \mid [A = a]\pi_k)}{P([D_k = d_k] \mid \pi_k)}$$

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:

$$\exists c_k, \forall a, P([D_k = d_k] \mid [A = a]\pi_k) \tag{3}$$
$$= c_k P([A = a] \mid [D_k = d_k]\pi_k)$$

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:

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

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:

$$P([A = a] \mid [\vec{D} = \vec{d}]\pi_f) \tag{5}$$
$$= \frac{P([A = a] \mid \pi_f) \prod_{k=1}^{n} P([D_k = d_k] \mid [A = a]\pi_f)}{P([\vec{D} = \vec{d}] \mid \pi_f)}$$

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:

$$P(A \mid \vec{D}\,\pi_f) \propto \prod_{k=1}^{n} P(D_k \mid A\,\pi_k) \tag{6}$$

Finally, by substituting equation 4, we get

$$P(A \mid \vec{D}\,\pi_f) \propto \prod_{k=1}^{n} P(A \mid D_k\,\pi_k) \tag{7}$$

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].

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:

$$P(AD_k \mid \pi_k) = P(D_k \mid \pi_k)P(A \mid D_k\,\pi_k) \tag{8}$$

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:

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

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

$$P(A \mid \vec{D}\,\pi_f) \tag{10}$$

$$\propto \quad P(A \mid \pi_f)\prod_k \frac{P(D_k \mid \pi_k)P(A \mid D_k\,\pi_k)}{P(A \mid \pi_k)}$$

3

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

$$P(A \mid \vec{D}\pi_f) \not\propto \prod_k P(A \mid D_k \pi_k) \tag{11}$$

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

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

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

Practically, this corresponds to

$$P(A \mid \pi_f) \quad \propto \quad \prod_k \sum_{D_k} P(A \mid D_k \pi_k) P(D_k \mid \pi_k) \tag{13}$$

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.

BAYESIAN FUSION WITH DIAGNOSIS

Definitions

In this section, we introduce a new variable:

- $I\!I$ or $I\!I_k$: boolean variable which **I**ndicates if the opinion A is *consistent* with the measure D_k.

We now express the following sub-model:

$$P(A D_k I\!I_k \mid \pi_k) = P(A \mid \pi_k) P(D_k \mid \pi_k) P(I\!I_k \mid A D_k \pi_k) \tag{14}$$

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

$$P([I\!I_k = 1] \mid A D_k \pi_k) = \exp\left(-\frac{1}{2}\left(\frac{A - D_k}{\sigma}\right)^2\right) \tag{15}$$

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

$$P(A \mid \vec{D}\vec{I\!I}\pi_f) \propto \prod_k P(A \mid D_k I\!I_k \pi_k) \tag{16}$$

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.

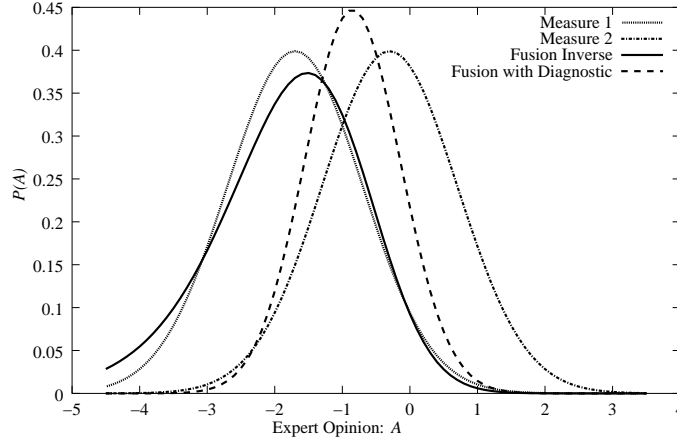[1] This is true when $I\!I$ is not considered.

FIGURE 1. Comparison of fusion processes

Proof of equation 16

First, we can write Bayes' rule as follows:

$$P([A = a] \mid [\vec{D} = \vec{d}][\vec{I} = \vec{\imath}]\,\pi_f) \tag{17}$$

$$= \frac{P([A = a]\,[\vec{D} = \vec{d}] \mid \pi_f)}{P([\vec{D} = \vec{d}]\,[\vec{I} = \vec{\imath}] \mid \pi_f)} \times$$

$$P([\vec{I} = \vec{\imath}] \mid [A = a]\,[\vec{D} = \vec{d}]\,\pi_f)$$

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:

$$P([A = a] \mid [\vec{D} = \vec{d}][\vec{I} = \vec{\imath}]\,\pi_f) \tag{18}$$

$$\propto \quad P([\vec{I} = \vec{\imath}] \mid [A = a]\,[\vec{D} = \vec{d}]\,\pi_f)$$

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

$$P([A = a] \mid [\vec{D} = \vec{d}]\,[\vec{I} = \vec{\imath}]\,\pi_f) \tag{19}$$

$$\propto \quad P([\vec{I} = \vec{\imath}] \mid [A = a]\,[\vec{D} = \vec{d}]\,\pi_f)$$

$$\propto \quad \prod_k P([I_k = i_k] \mid [A = a]\,[D_k = d_k]\,\pi_k)$$

Another application of Bayes' rule leads to:

$$P([A = a] \mid [\vec{D} = \vec{d}]\,[\vec{I} = \vec{\imath}]\,\pi_f) \tag{20}$$

$$\propto \quad \prod_k \left[\frac{P([D_k = d_k]\,[I_k = i_k] \mid \pi_k)}{P([A = a]\,[D_k = d_k] \mid \pi_k)} \right.$$

$$\left. \times P([A = a] \mid [D_k = d_k]\,[I_k = i_k]\,\pi_k) \right]$$

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.

Properties

Generally, we are interested in the case where we assume that our experts are competent, and so $\vec{\mathbb{1}} = \vec{1}$. 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 $\vec{\mathbb{1}} = \vec{1}$.

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.

APPLICATION TO HOUGH TRANSFORM

The Hough transform is a generic method to evaluate some parameters given noisy data samples. It was first proposed in [4] for extracting lines from an image. We will first remind briefly about the algorithm and one of its upgrade the randomized Hough Transform as presented in [7]. Then we will replace this algorithm in a bayesian framework and see that it is mainly a fusion process.

Hough Transform and Randomized Hough Transform

Randomized Hough Transform (RHT) has been proposed in [7] as an upgrade for plain Hough Transform. The goals of this two algorithms are the same. They are to find the lines in an image from the edge pixels. The basic idea is to find a good parameter space for the line extracted. Then they build an histogram in this space, each pixel contributing to the set of lines it may belong to. Finally the line(s) having received the more contributions are decided to be the one looked for.

Hough Transform

More formally, let the lines be represented in the $(\rho, \theta) \in \mathbb{R}^+ \times\,]-\pi; \pi]$ space, where ρ is the distance of the line from the origin and θ its bearing as shown in figure 2. For a given pixel (x, y) labeled as an edge in the picture, we can write a relation between the distance and bearing of the lines to characterize those passing by this point. We have indeed:
$$\rho = \sqrt{x^2 + y^2} \sin(\theta)$$

Then we are interested in the number of pixels in the source image that belong to each line. The Hough Transform thus adds a contribution for each point to the counters of all the lines it belongs to. This accumulator array is build according to the characterization of the lines. . This is illustrated by algorithm .

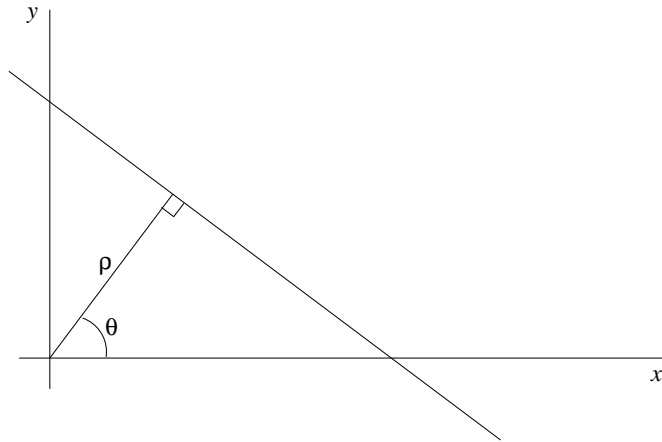Finally, we can apply a threshold or get the line which holds the greatest number of edge pixels.

FIGURE 2. Characterization of a line

Algorithm 1 Hough Transform.

---
  for all Pixel **do**
    for all Theta **do**
      Rho ← sqrt(Pixel.x^2+Pixel.y^2)*sin(Theta)
      Count[Rho, Theta] ← Count[Rho, Theta] + 1
    end for
  end for

Randomized Hough Transform

This former algorithm is well known but may be improved concerning its complexity or its robustness to noise (or outliers). One of its derivative is the Randomized Hough Transform defined in [7]. We will present it shortly in this section.

Hough Transform loops through all the edge points in the image and each point contributes to the whole set of lines that pass through this pixel. In Randomized Hough Transform, rather than having a single point contributing for every lines, we take two pixels that add credit for the line defined by them.

Picking two pairs of points in the same line will give very consistent results whereas the probability of having reinforcement in the belief on one line given pairs of pixels from different lines is very low.

It is easy to see this stochastic algorithm converges much faster than the previous one for usual signal to noise ratios (few off-line edge dots) and a small number of lines.

Bayesian Hough Transform

We have presented so far two deterministic algorithms to extract lines from an image treated by edge detection. In this section we will design a bayesian program based on a fusion process in order to accomplish the same task. We will see that RHT can be seen

7

as an approximation of such a fusion. Finally, we will put forth the hypothesis of this approximation.

Variables and uncertainty

We will consider the picture, an array of pixel, as a discretization of the continuous image we are interested in. In this respect, an edge pixel detected by the preprocessing will denote the existence of a particular edge point in the limits of his pixel. This means that given the coordinate $(x, y) \in \mathbb{N}^2$ of a pixel, the real coordinates of the edge belong to $[x; x+1[\times [y; y+1[$. This uncertainty follows directly from the resolution of the camera in a real image or from the resolution of the rendering for a simulated scene for example. This uncertainty is precisely the reason why we may need a bayesian program to handle it.

We will consider a set of so-called observations $\{o_i\}_i$, each observation being a pair of dots $((x_i^1, y_i^1), (x_i^2, y_i^2))_i$. We will then be interested in the parameters (ρ, θ) of the single line to be extracted.

The variables are then:

- (P, Θ): the parameters;
- $\{O_i\}_i$: the set of observations.

It is interesting to note that we will end with a probability distribution over the parameters of one line. In case of several lines, this distribution will be multi-modal, each mode corresponding to a different line. We will then use multi-modality to weaken the one-line assumption we made. This is a common exploit of probability, even without explicit reference.

Local model

Having defined our variables, we are going to express our knowledge about the relation between the parameters of the line and the coordinates of the pixels. The usual way is to deduce the parameters from the coordinates: $(\rho, \theta) = f((x_i^1, y_i^1), (x_i^2, y_i^2))$ or specifying $P(P \Theta \mid X^1 Y^1 X^2 Y^2)$. This is then a case of what we called an inverse model. For the reasons given above, we may thus want to use fusion with diagnosis. Furthermore, this fusion scheme will also allow us to deal properly with outliers, i.e. observations not coherent with the line extracted.

We can get an exact expression of $P(\Theta \mid O_k)$ and $P(P \mid O_k) = \sum_\theta P(P \mid \theta) \times P(\theta \mid O_k)$ if the dots belong to the line. This way, we can define $P(I\!I_k \mid P \Theta O_k)$ for $I\!I_k = 1$. For $I\!I_k = 0$, these are defined as uniform distribution. Finally, we are able to end up with a specification of our local model. We can then compute:

$$
\begin{aligned}
P(P \Theta \mid O_k) \;=\; & P([I\!I_k = 1]) \times P(P \Theta \mid O_k [I\!I_k = 1]) \\
& +\; P([I\!I_k = 0]) \times P(P \Theta \mid O_k [I\!I_k = 0])
\end{aligned}
\tag{21}
$$

8

This distribution is illustrated figure 3 by the distribution $P(\theta \mid o_k)$ of the angle of the line knowing a pair of dots.
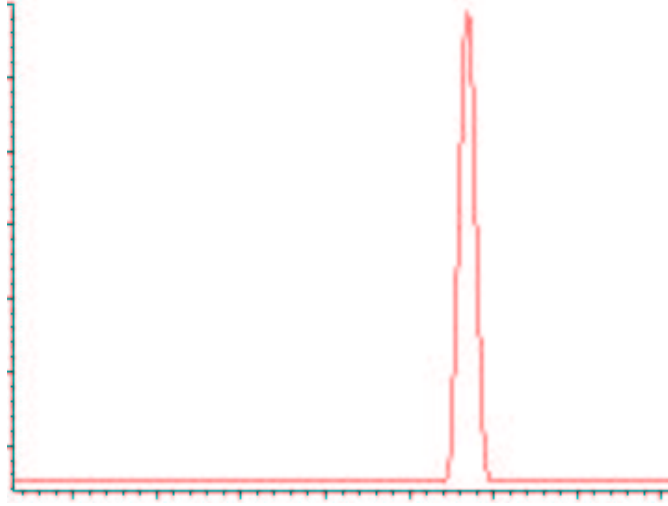

FIGURE 3. Distribution over the angle: $P(\theta \mid o_k)$

Inference

Once the local model is defined we can use the results of the fusion with diagnosis. Indeed, from equation 16 we can show directly:

$$P(A \mid \vec{D} \pi_f)) \propto \prod_k P(A \mid D_k \pi_k) \qquad (22)$$

Furthermore we have shown earlier that $\forall a \in A, P(a \mid D_k \pi_k) \neq 0$. We can therefore take the logarithm of equation 22:

$$\log P(A \mid \vec{D} \pi_f)) = \sum_k \log P(A \mid D_k \pi_k) + \kappa \qquad (23)$$

This way, we end up with a summation process. We can however note that the support of $P(A \mid D_k [\mathbb{I}_k = 1] \pi_k)$ is the only place where $\log P(A \mid D_k \pi_k) \neq c = \log P([\mathbb{I}_k = 0] \mid \pi_k) \times P(A \mid D_k [\mathbb{I}_k = 0] \pi_k)$ As c does not depend on either the value of A or model k, this constant can be included in κ and neglected in a peak search.

We end up with only a local accumulation as in randomized Hough transform. The only assumption we need then to get the exact same computation, is that the discretization must be larger than the width of $P(A \mid D_k [\mathbb{I}_k = 1] \pi_k)$. This way, the whole update is limited to a single value. This assumption may however be false for pairs of dots in a close neighborhood but it is usually sound enough.

This section presented a way to rewrite Randomized Hough Transform in a bayesian framework. We were able to find back the usual accumulation algorithm by using a fusion with diagnostic process. Finally, this allows us to give a bayesian interpretation of this line extraction algorithm.

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.

ACKNOWLEDGMENTS

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. P.V.C. Hough. Method and means for recognizing complex patterns. In *US Patent*, 1962.
5. E. T. Jaynes. Probability theory: the logic of science. Unprinted book, available on-line at `http://bayes.wustl.edu/etj/prob.html`, 1995.
6. O. Lebeltel, P. Bessière, J. Diard, and E. Mazer. Bayesian robots programming. Research Report 1, Les Cahiers du Laboratoire Leibniz, Grenoble (FR), May 2000.
7. L. Xu, E. Oja, and P. Kultanen. A new curve detection method: Randomized hough transform. *PRL*, 11:331–338, 1990.