Navigation intentionnelle d'un robot mobile

Cédric Pradalier

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N^o attribué par la bibliothèque
|_/_/_/_/_/_/_/_/_/_|

THÈSE

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : « Imagerie, Vision, Robotique »

préparée au laboratoire GRAVIR et à l'INRIA Rhône-Alpes, dans le cadre de
l'École Doctorale « Mathématiques, Sciences et Technologies de l'Information, Informatique »

présentée et soutenue publiquement par

Cédric PRADALIER

le 15/09/2004

Titre :

Navigation intentionnelle d'un robot mobile

Directeur de Thèse :

Christian LAUGIER

Composition du jury :

M.	James L. CROWLEY	Président
M.	Alessandro DE LUCA	Rapporteur
M.	Roland SIEGWART	Rapporteur
M.	Raja CHATILA	Examinateur
M.	Christian LAUGIER	Directeur de thèse
M.	Pierre BESSIÈRE	Co-directeur de thèse

Table des matières

Table des figures

Chapitre 1

Introduction

1.1 Situation

L'objectif de la robotique autonome est le contrôle, par un système informatique, d'un système mécanique non fixé à un support et doté de périphériques de perceptions et d'actions. Ce domaine de recherche appliquée se situe au carrefour de l'ingénierie, de l'intelligence artificielle, de la perception par ordinateur, de la planification de mouvements et de l'automatisme. Il s'agit donc d'une discipline qui doit faire la synthèse d'un ensemble de compétences de plus en plus vaste. Sans surprise, ce pluralisme est à l'origine d'une certaine complexité.

1.2 Objectifs et contributions

Inscrit dans le contexte de la robotique autonome, cette thèse se focalise sur l'étude de la navigation intentionnelle, c'est à dire le pilotage d'un robot mobile de façon à atteindre un but en tenant compte d'informations perceptives.

Pour atteindre cet objectif, nous allons nous poser trois questions fondamentales que nous préciserons par la suite :

- Comment définir une tâche de navigation intentionnelle (TNI) ?
- Comment réaliser une TNI ?
- Comment implanter une TNI ?

Nos réponses à ces questions s'articulent en une arborescence de problématiques que nous illustrons dans la figure 1.1. Nous allons maintenant détailler cette arborescence.

1.2.1 Définition d'une TNI

Le chapitre 3 nous permettra d'expliciter le sens que nous donnons à une TNI. De plus, nous profiterons de ce chapitre pour définir quelques notions qui nous seront utiles par la suite : les espaces utilisés pour la navigation et les notions de comportement et de trajet.

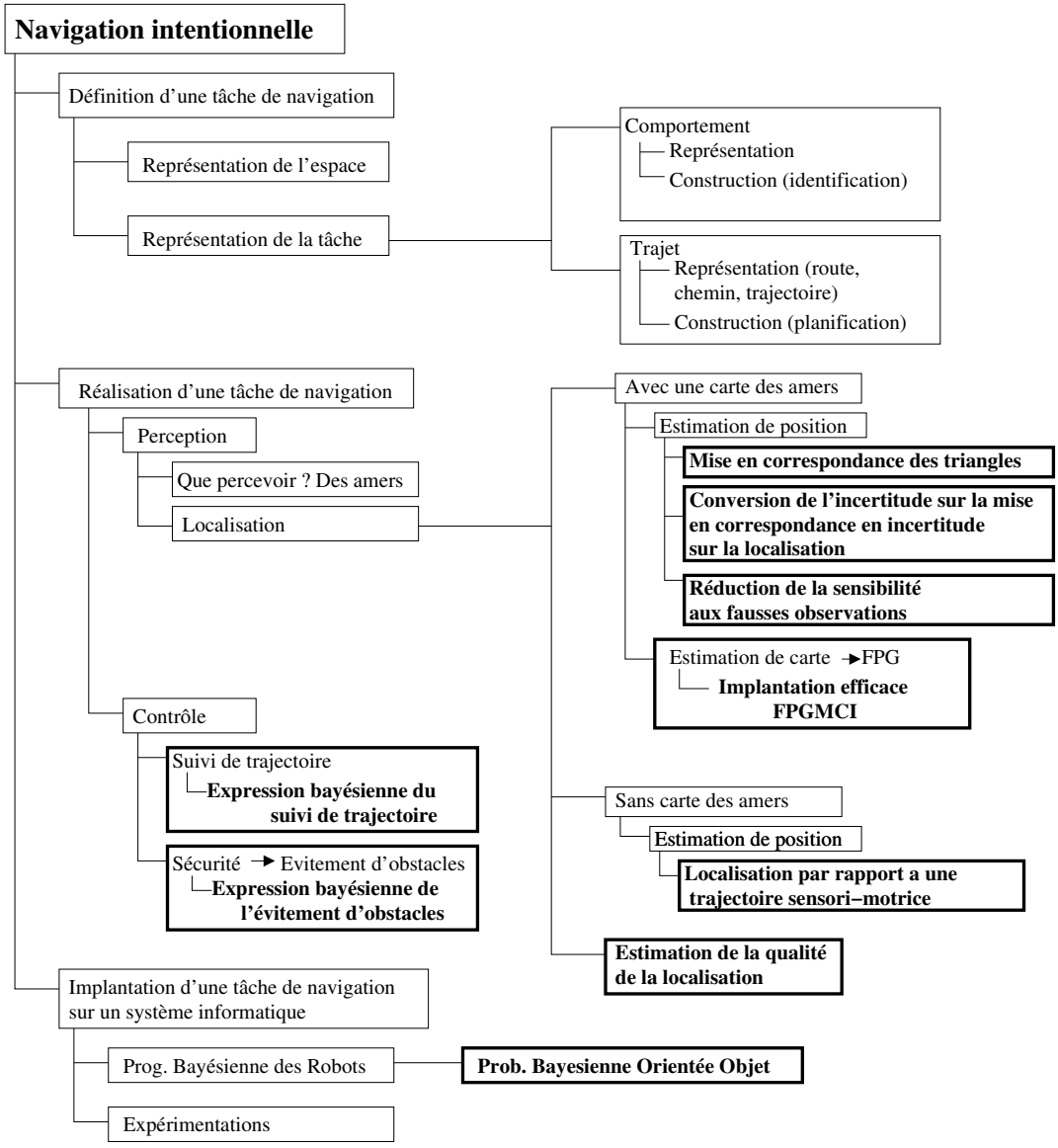


FIG. 1.1 – Problématiques traitées dans cette thèse. Nos contributions sont encadrées en gras.

1.2.2 Réalisation d'une TNI

C'est en répondant à la question de réalisation d'une TNI que nous proposerons la majorité de nos contributions. Nous considérons que cette réalisation passe nécessairement par la mise en place d'une boucle localisation-contrôle, ou boucle sensorimotrice. Nous étudierons donc d'abord les techniques de localisation et ensuite le contrôle d'un robot mobile.

a) Localisation

La localisation est l'élément central de ce travail. Pour simplifier ce problème vaste et complexe, nous avons choisi de ne pas traiter le problème de l'extraction d'amers naturels à partir des perceptions brutes. Nous considérerons donc dans tout ce travail que notre robot est équipé d'un détecteur d'amers et, en pratique, nous installerons des amers artificiels dans notre environnement.

Ce choix des amers étant fait, une question se pose : désire-t-on se localiser par rapport à une carte absolue ou non ? Dans la suite de ce document, l'adjectif *absolu* qualifiera une estimation par rapport à un repère fixe, indépendant du mouvement du robot.

*1.***Localisation par rapport à une carte** *[Pradalier & Sekhavat 2002a; 2004]* **:** Dans le cas où l'on choisit de se localiser par rapport à une carte, il faut être capable non seulement d'estimer la localisation du robot, mais aussi de construire une représentation interne de la carte.

Estimation de position La localisation peut se faire par triangulation. Dans ce cas, si les amers ne sont pas immédiatement identifiables, il est nécessaire d'utiliser un algorithme d'identification des amers observés. Cet algorithme doit être à la fois robuste et efficace. Dans cette optique, nous proposerons une optimisation de la mise en correspondance par graphes de correspondances, fondée sur l'identification des triangles formés par trois amers.

Cependant, même avec cette optimisation, les techniques d'identification explicite ne permettent pas de convertir l'incertitude sur la mise en correspondance en incertitude sur la localisation. Nous allons donc montrer comment obtenir ce résultat en décrivant un modèle de localisation probabiliste dont la mise en correspondance est implicite.

Nous allons aussi montrer comment améliorer encore la robustesse de cette localisation probabiliste aux fausses observations. Pour cela, nous utiliserons des modèles particuliers : les modèles de diagnostic.

Construction de carte Pour ce qui est de la construction de carte, nous avons choisi d'utiliser le filtre à projection géométrique (FPG) : un filtre qui estime l'état d'une carte d'amers grâce à l'estimation des distances inter-amers. Pour fonctionner, ce filtre a lui aussi besoin d'un algorithme de mise en correspondance. Nous montrerons donc comment intégrer efficacement la mise en correspondance par graphes de correspondances avec le FPG, et obtenir ainsi le FPGMCI – Filtre à Projection Géométrique avec Mise en Correspondance Intégrée.

*2.***Localisation sans carte** *[Pradalier et al. 2004a]* Si on choisit de ne pas utiliser de carte absolue, nous allons montrer qu'il est tout de même possible de construire un modèle de localisation probabiliste pour se localiser par rapport à un trajet sensorimoteur, trajet défini par une séquence de perceptions.

*3.***Qualité de la localisation** *[Pradalier & Bessière 2004]* Que l'on choisisse de se localiser par rapport à une carte ou non, il est toujours important d'être capable d'évaluer la qualité de ses hypothèses de localisation. Pour obtenir ce résultat en temps réel, nous proposerons un modèle probabiliste qui estimera la confiance du système. Cette confiance sera définie à partir de la cohérence entre les observations réelles et les observations prédites grâce à un modèle de l'environnement et aux hypothèses de localisation.

b) Contrôle

Après avoir considéré la question de la localisation, nous traiterons le problème de la génération de commandes en fonction de la localisation.

*1.***Suivi de trajectoire** Lorsque la TNI est définie par une trajectoire, elle ne peut être réalisée avec succès sans un algorithme de suivi de trajectoire. Pour répondre à ce besoin, nous montrerons que ce problème de suivi de trajectoire peut être exprimé comme un problème d'inférence bayésienne.

*2.***Évitement d'obstacles** *[Koike et al. 2003a;b]* Pour garantir la sécurité du robot et de son environnement pendant la réalisation de la TNI, nous serons nécessairement confrontés au problème de l'évitement d'obstacles. Comme pour le suivi de trajectoire, nous montrerons que ce problème peut s'exprimer sous la forme d'un problème d'inférence bayésienne et nous mettrons l'accent sur la richesse sémantique qui en découle.

1.2.3 Implantation d'une TNI *[Hermosillo et al. 2003a; 2004; 2003b, Pradalier et al. 2003c; 2004b]*

Finalement, le chapitre 7 présentera notre réponse à notre dernière question : l'implantation d'une TNI sur un système informatique. Pour cela, nous avons choisi de nous placer dans le cadre du formalisme de la programmation bayésienne. Nous montrerons que ce formalisme souffre d'un manque de capacité de généralisation et d'abstraction. Nous proposerons donc une extension du formalisme inspirée des méthodes de programmation orientées objet : la Programmation Bayésienne Orientée Objet (PBOO).

1.3 Plan de lecture

Notre document traite la problématique de la navigation intentionnelle à travers huit chapitres.

Le chapitre 2 introduit les éléments mathématiques et le formalisme de base de l'inférence bayésienne. La notion de programme bayésien y est aussi introduite ainsi que quelques exemples.

Dans le chapitre 3 nous définissons la notion de tâche de navigation intentionnelle.

Les trois chapitres suivants décrivent les compétences nécessaires à la réalisation d'une tâche de navigation intentionnelle.

Le chapitre 4 présente ensuite nos contributions dans le cadre de la localisation :

- la localisation par triangulation et l'identification explicite des observations grâce à l'identification de triplés d'amers,
- les principes de la localisation bayésienne avec mise en correspondance implicite pour transformer l'incertitude sur l'identification des observations en incertitude sur la localisation,
- la localisation sur une trajectoire définie sous une forme sensorimotrice,
- et la construction d'un estimateur de confiance pour évaluer la qualité de la localisation.

Dans le chapitre 6, nous décrivons les stratégies de commande du robot mobile que nous avons développées. En particulier, nous montrons comment nous avons exprimé les problèmes de suivi de trajectoire et d'évitement d'obstacles sous la forme de problèmes d'inférence bayésienne.

Nous présentons ensuite l'intégration de cet ensemble de compétences robotiques dans le chapitre 7. Nous décrivons d'abord notre extension de la programmation bayésienne pour la modélisation orientée objet d'un système complexe, et nous l'utilisons ensuite pour deux applications particulières : une première qui utilise un planificateur pour construire et exécuter des trajectoires dans un environnement modérément dynamique, et une seconde qui donne au CyCab la capacité de se déplacer sur des trajectoires définies uniquement en termes sensorimoteurs.

La conclusion de ce travail et nos pistes de recherche pour l'avenir sont proposées au chapitre 8.

Remarquons finalement que l'annexe A décrit nos réflexions sur la notion de fusion de données par inférence bayésienne *[Pradalier et al. 2003a;b]*.

Chapitre 2

Introduction à la programmation bayésienne

Nous présentons dans ce chapitre le minimum théorique et mathématique nécessaire pour la compréhension des résultats présentés dans les chapitres suivants. Notre objectif est de présenter brièvement les principes du calcul bayésien comme une méthode de modélisation et d'inférence permettant de prendre en compte les incertitudes. Pour une présentation détaillée, nous renvoyons aux travaux de Pierre Bessière *[Bessière et al. 1998a;b]* et d'Olivier Lebeltel *[Lebeltel 1999, Lebeltel et al. 2004]*.

Nous nous plaçons plus précisément dans le cadre d'une théorie du raisonnement probabiliste appelée *Probability as Logic* (PaL) proposée par le physicien E.T.Jaynes *[Jaynes 2003]*. Cette théorie considère le calcul des probabilités comme une généralisation de la logique formelle (booléenne). Elle permet d'étendre la logique à des propositions dont la vérité n'est pas connue avec certitude et de formaliser la notion de raisonnement plausible. Dans cette théorie, la notion de probabilité est utilisée non pas dans son sens *fréquentiste* où elle caractérise la *fréquence* d'une mesure physique donnée, mais plutôt dans son sens *subjectiviste* où elle exprime un *état de connaissance* représentant les *informations* dont on dispose sur le phénomène en question. Cette utilisation des probabilités découle directement de la formalisation de la notion intuitive de *plausibilité* proposée par Cox *[Cox 1946; 1961]*. En se fixant un ensemble de desiderata définissant la notion de plausibilité pour une proposition logique, Cox montre que l'unique façon de manipuler cette notion (en restant fidèle aux desiderata de départ) est donnée par la théorie des probabilités. La notion de plausibilité est ainsi formalisée par la notion mathématique bien définie de *probabilité*.

2.1 Définitions fondamentales

2.1.1 Probabilité d'une proposition logique

Une proposition logique est un énoncé qui peut être vrai ou faux. Nous définissons la plausibilité (probabilité) $\mathrm{P}(\mathcal{A})$ d'une proposition $\mathcal{A}$ comme le degré de certitude accordé à sa véracité.

En réalité, toute probabilité d'une proposition $\mathcal{A}$ ne peut être donnée qu'au vu d'un ensemble de *connaissances préalables* que l'on dénotera π. Il est alors plus convenable d'expliciter dans les notations les connaissances π qui ont permis d'assigner une valeur $\mathrm{P}_{\mathcal{A}}$ à une proposition $\mathcal{A}$ en écrivant $\mathrm{P}(\mathcal{A} \mid \pi) = \mathrm{P}_{\mathcal{A}}$. Toutefois, dans le but d'alléger les notations, nous nous dispensons d'écrire ces connaissances π dans les formules que nous manipulons dans le reste de ce document. En effet à un instant donné nous ne nous intéressons qu'à un seul ensemble de connaissances préalables π.

Nous utiliserons, dans ce qui suit, la notation $\mathcal{AB}$ pour la conjonction de $\mathcal{A}$ et $\mathcal{B}$, $\mathcal{A} + \mathcal{B}$ pour leur disjonction et $\neg\mathcal{A}$ et $\neg\mathcal{B}$ pour leurs négations respectives.

2.1.2 Règles de calcul

À partir des desiderata définissant le raisonnement plausible, Cox montre que ce raisonnement doit utiliser 2 règles fondamentales qui sont à la base de toute inférence probabiliste. Étant données les propositions logiques $\mathcal{A}$ et $\mathcal{B}$, ces deux règles sont la règle du produit et la règle de normalisation.

a) Règle du produit

La règle du produit donne la probabilité d'une conjonction :

$$\mathrm{P}(\mathcal{A}\,\mathcal{B}) = \mathrm{P}(\mathcal{A})\mathrm{P}(\mathcal{B} \mid \mathcal{A}) = \mathrm{P}(\mathcal{B})\mathrm{P}(\mathcal{A} \mid \mathcal{B}). \tag{2.1}$$

b) Règle de normalisation

La règle de normalisation exprime le fait que la somme des probabilités d'une proposition et de sa négation est égale à 1 :

$$\mathrm{P}(\mathcal{A}) + \mathrm{P}(\neg\mathcal{A}) = 1. \tag{2.2}$$

2.1.3 Autres définitions et notations utiles

a) Probabilité d'une disjonction

L'utilisation des règles (2.1)et (2.2) permet d'écrire la probabilité d'une disjonction comme suit :

$$\mathrm{P}(\mathcal{A} + \mathcal{B}) = \mathrm{P}(\mathcal{A}) + \mathrm{P}(\mathcal{B}) - \mathrm{P}(\mathcal{A}\,\mathcal{B}). \tag{2.3}$$

Cette dernière règle s'écrit dans le cas où A et B sont indépendantes :

$$\mathrm{P}(\mathcal{A} + \mathcal{B}) = \mathrm{P}(\mathcal{A}) + \mathrm{P}(\mathcal{B}). \tag{2.4}$$

b) Variable discrète - distribution de probabilité discrète

Jusqu'ici, seule la probabilité d'une proposition logique a été définie. Supposons que l'on cherche à assigner des probabilités aux différentes valeurs numériques d'une variable discrète X. Nous devons dans ce cas définir un ensemble de propositions logiques mutuellement exclusives $\mathcal{A}_n \equiv$ "$X = n$", où n prend toutes les valeurs possibles de X, et considérer les probabilités $\mathrm{P}(\mathcal{A}_n) = \mathrm{f}(n)$. La fonction f représente une *distribution de probabilité discrète* sur la plage de variation de la variable X.

Nous utilisons la notation $\mathrm{P}(X)$ pour désigner une distribution de probabilité sur une variable discrète X.

Les deux règles (2.1) et (2.2) continuent à s'appliquer dans le contexte des variables discrètes et s'écrivent respectivement, pour deux variables X et Y, comme suit :

$$\mathrm{P}(X\,Y) = \mathrm{P}(X)\mathrm{P}(Y \mid X) = \mathrm{P}(Y)\mathrm{P}(X \mid Y)$$

$$\sum_X \mathrm{P}(X) = 1.$$

c) Variable continue - densité de probabilité

Afin de considérer la notion de probabilité d'une variable continue X tout en utilisant les résultats obtenus pour les propositions logiques, nous reprenons le raisonnement donné par Jaynes :

Commençons par définir les deux propositions suivantes :

$$\mathcal{F}_1 \equiv (X \leq q) \tag{2.5}$$

$$\mathcal{F}_2 \equiv (X > q) \tag{2.6}$$

Ces deux propositions sont exhaustives et mutuellement exclusives. La probabilité de $\mathcal{F}_1$ va dépendre de q en définissant une fonction G telle que :

$$\mathrm{P}(\mathcal{F}_1) = \mathrm{G}(q).$$

Cette fonction G est bien évidemment croissante.

Intéressons nous maintenant à la proposition $\mathcal{W} \equiv (a < X \leq b)$ (la valeur de X est dans l'intervalle $]a, b]$). Notons $\mathcal{A}$ et $\mathcal{B}$ les propositions suivantes :

$$\mathcal{A} \equiv (X \leq a); \tag{2.7}$$

$$\mathcal{B} \equiv (X \leq b). \tag{2.8}$$

Nous avons l'égalité $\mathcal{B} = \mathcal{A} + \mathcal{W}$. De plus, comme $\mathcal{A}$ et $\mathcal{W}$ sont mutuellement exclusives, l'application de la règle de la somme permet d'écrire :

$$\mathrm{P}(\mathcal{B}) = \mathrm{P}(\mathcal{A}) + \mathrm{P}(\mathcal{W}),$$

ce qui donne finalement :

$$\mathrm{P}(a < X \leq b) = p(\mathcal{W}) = \mathrm{G}(b) - \mathrm{G}(a).$$

Si G est différentiable, nous pourrons écrire :

$$\mathrm{P}(a < X \leq b) = \int_a^b \mathrm{g}(X)dX$$

avec $\mathrm{g}(X) = \frac{d\mathrm{G}(X)}{dX}$.

La fonction g est appelée *densité de probabilité* ou bien *distribution de probabilité* sur la variable X. Nous l'appellerons dans le reste de ce document *distribution de probabilité* et nous utiliserons la notation $\mathrm{P}(X)$ pour dénoter la distribution sur une variable continue.

Les deux règles fondamentales (2.1) et (2.2) restent valides pour les cas des variables continues et s'écrivent respectivement pour deux variables X et Y comme suit :

$$\mathrm{P}(X\ Y) = \mathrm{P}(X)\mathrm{P}(Y \mid X) = \mathrm{P}(Y)\mathrm{P}(X \mid Y);$$

$$\int \mathrm{P}(X)dX = 1.$$

La distribution $\mathrm{P}(X\ Y)$ est appelée la *distribution conjointe* de X et Y.

d) Marginalisation - Distribution marginale

Soit X et Y deux variables numériques. On appelle *distribution marginale* de X par rapport à Y la distribution de probabilité :

- $\mathrm{P}(X) = \sum_Y \mathrm{P}(X\ Y) = \sum_Y \mathrm{P}(Y)\mathrm{P}(X \mid Y)$ dans le cas où Y est discrète ;
- $\mathrm{P}(X) = \int \mathrm{P}(X\ Y)dY = \int \mathrm{P}(Y)\mathrm{P}(X \mid Y)dY$ dans le cas où Y est continue.

e) Formule de Bayes

Une réécriture directe de la règle du produit permet d'obtenir la formule dite de *Bayes*

$$\mathrm{P}(X \mid Y) = \frac{\mathrm{P}(X\ Y)}{\mathrm{P}(Y)} = \frac{\mathrm{P}(X)\mathrm{P}(Y \mid X)}{\mathrm{P}(Y)} = \frac{\mathrm{P}(X)\mathrm{P}(Y \mid X)}{\sum_X \mathrm{P}(X)\mathrm{P}(Y \mid X).}$$

Lorsque Y est connu, $\mathrm{P}(Y)$ est une constante indépendante de X. On sait donc que $\mathrm{P}(X \mid Y)$ est proportionnel à $\mathrm{P}(X)\mathrm{P}(Y \mid X)$, et on peut déduire le coefficient de proportionnalité de la règle de normalisation.

La relation de proportionnalité entre distributions sera noté $\propto$:

$$\mathrm{P}(X \mid Y) \propto \mathrm{P}(X)\mathrm{P}(Y \mid X)$$

2.2 Méthode de programmation bayésienne

Dans cette section, nous présentons brièvement la méthode appelée “Programmation Bayésienne”, qui sera utilisée dans le reste du document. Cette méthode se base sur un objet formel, la *description [Lebeltel 1999, Lebeltel et al. 2004]*.

f) Structure générale

La définition d’un *programme bayésien* à l’aide d’une description se décompose en trois phases que nous expliciterons dans les prochains paragraphes :

– la spécification des connaissances préalables ;
– l’identification des valeurs des paramètres des distributions de probabilité ;
– l’utilisation de la description.

Nous allons maintenant définir le concept de description et décrire plus précisément chacune de ces trois phases. La figure 2.1 illustre la structure générale que nous utiliserons pour définir un programme bayésien.

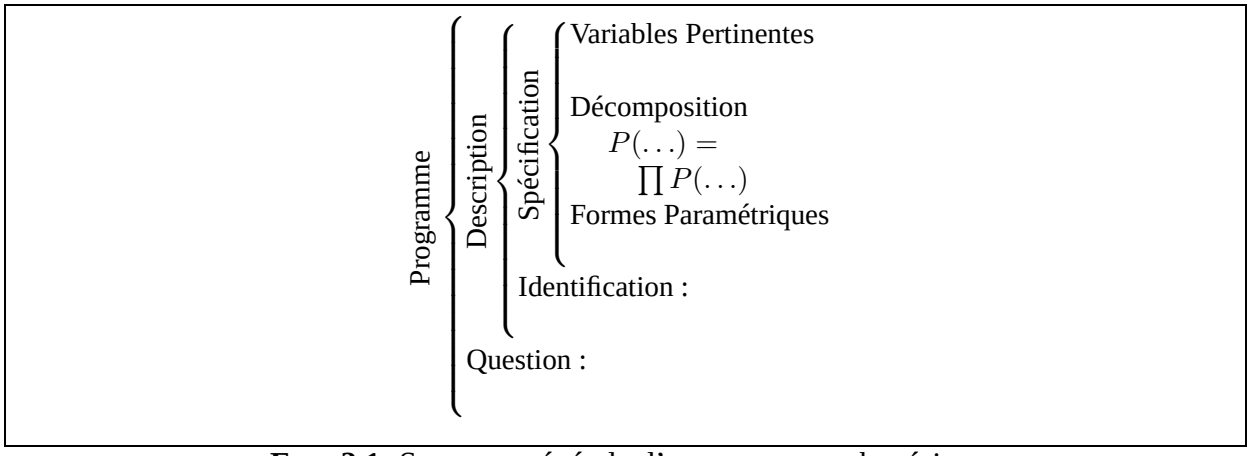


FIG. 2.1: Structure générale d’un programme bayésien.

2.2.1 Définition de la notion de *description*

Une description est dénotée formellement par la distribution de probabilité conjointe d’un ensemble de variables $V_1, \ldots, V_n$: $P(V_1 \ldots V_n \mid \pi)$ déterminée au vu des connaissances préalables π dont une partie est spécifiée par le programmeur et le reste peut provenir d’un ensemble de données expérimentales.

2.2.2 Spécification

La phase de spécification est la partie la plus délicate du travail du programmeur. Au cours de cette phase, il doit énoncer *explicitement* les connaissances qu’il apporte à la description et celles qui résultent d’un processus adaptatif dépendant d’un jeu particulier de données expérimentales.

On peut distinguer trois types de connaissance : le choix des variables pertinentes, l'expression des dépendances entre les variables retenues sous la forme d'un produit de distributions élémentaires, et enfin la forme paramétrique associée à chacune de ces distributions.

a) Connaissances préalables structurelles : le choix des variables pertinentes

Nous appelons *connaissances préalables structurelles* les connaissances permettant de définir l'ensemble des variables $V_1, \ldots, V_n$ pour la description. Toutes les autres variables sont ainsi supposées non pertinentes pour le problème considéré.

En robotique, nous pouvons classer ces variables naturellement en trois sous-ensembles :

- les variables sensorielles extéroceptives et proprioceptives ;
- les variables motrices ;
- les variables internes, qui permettent de représenter les états internes du robot ou de ses capteurs.

b) Connaissances préalables de dépendance : le choix d'une décomposition de la distribution conjointe

Comme énoncé précédemment, la description sur les variables $V_1, \ldots, V_n$ a pour but la définition de la distribution conjointe $P(V_1 \cdots V_n \mid \delta\, \pi)$. Cette forme mathématique est une distribution de probabilité sur n dimensions. La règle du produit (2.1) nous permet de décomposer cette expression, en l'exprimant sous forme de produit de distributions.

Prenons en exemple une distribution conjointe $\mathrm{P}(X\ Y\ Z)$ de 3 variables X, Y et Z. L'application de la règle du produit permet d'écrire :

$$\mathrm{P}(X\ Y\ Z) = \mathrm{P}(Z)\mathrm{P}(Y \mid Z)\mathrm{P}(X \mid Y\ Z).$$

Cette seconde étape de la spécification permet également d'exprimer les relations de dépendance, ou d'indépendance, entre les variables. Ces indépendances permettent de réduire fortement les dimensions des termes apparaissant dans la décomposition.

Si l'on suppose dans notre exemple que les variables X et Y sont indépendantes sachant la valeur de Z, on aura :

$$\mathrm{P}(X\ Y\ Z) = \mathrm{P}(Z)\mathrm{P}(Y \mid Z)\mathrm{P}(X \mid Z).$$

c) Connaissances préalables d'observation : le choix des formes paramétriques

Il faut maintenant associer à chacun des termes apparaissant dans la décomposition choisie à l'étape précédente une forme paramétrique. Par ces choix, nous allons fournir des *a priori* sur les valeurs des distributions de probabilité et la manière dont ces valeurs seront modifiées par l'expérience. Ce dernier point est composé d'un ensemble de valeurs initiales pour les paramètres, et d'un mécanisme de mise à jour au vu de données expérimentales (ce mécanisme peut éventuellement être vide si l'on veut figer la distribution lors de la présente phase de spécification). Une description dans laquelle tous les termes sont ainsi fixés est appelée *spécification (description) a priori*.

Les formes paramétriques sont en général des lois de probabilité classiques, par exemple des lois uniformes ou des lois normales. Une *question* à une autre description peut également être utilisée comme forme paramétrique, à la manière d'un sous-programme probabiliste.

2.2.3 Identification

Les formes paramétriques peuvent contenir des paramètres libres, comme les moyennes ou écarts types de distributions gaussiennes. Il est nécessaire de fixer les valeurs numériques de ces paramètres pour achever notre description. Ces valeurs peuvent soit être obtenues par un processus d'apprentissage, soit être fixées *a priori* par le programmeur.

2.2.4 Utilisation

Au terme des phases de spécification et d'identification, nous disposons d'une description complètement définie. La phase d'utilisation va consister à mettre en œuvre les descriptions par le biais de questions probabilistes.

a) Question (définition)

Poser une question consiste à chercher la distribution de probabilité d'un certain nombre de variables $\mathcal{E}_q$ de la description, connaissant les valeurs d'autres variables $\mathcal{E}_c$, et éventuellement ignorant les valeurs d'un troisième groupe de variables $\mathcal{E}_i$. Une question probabiliste est donc n'importe quelle expression de la forme :

$$\mathrm{P}(V_k \ \ldots \ V_l \mid v_m \ \ldots \ v_n), \tag{2.9}$$

où $\mathcal{E}_q = \{V_k, \ldots, V_l\} \neq \emptyset$, $\mathcal{E}_c = \{V_m, \ldots, V_n\}$, et $\mathcal{E}_i = \{V_o, \ldots, V_p\}$ est l'ensemble des variables n'apparaissant ni dans $\mathcal{E}_q$, ni dans $\mathcal{E}_c$. Ces trois ensembles doivent bien sûr former une partition de l'ensemble des variables considérées pour que la question ait un sens.

b) Inférence

La connaissance de la distribution conjointe $\mathrm{P}(V_1 \ \cdots \ V_n)$ et l'application des règles du produit (2.1) et de la marginalisation (2.2) nous permet de répondre à toute question de la forme (2.9). Appliquons d'abord la règle de Bayes :

$$\mathrm{P}(V_k \ \ldots \ V_l \mid v_m \ \ldots \ v_n) = \frac{\mathrm{P}(V_k \ \ldots \ V_l \ v_m \ \ldots \ v_n)}{\mathrm{P}(v_m \ \ldots \ v_n)}. \tag{2.10}$$

En appliquant maintenant la règle de marginalisation, nous pouvons exprimer le dénominateur et le numérateur de ce quotient en fonction de la distribution conjointe que l'on sait calculer :

$$\mathrm{P}(V_k \ \ldots \ V_l \mid v_m \ \ldots \ v_n) = \frac{\sum_{V_o \ldots V_p} \mathrm{P}(V_k \ldots V_l \, v_m \ldots v_n \, V_o \ldots V_p)}{\sum_{\substack{V_k \ldots V_l \\ V_o \ldots V_p}} \mathrm{P}(V_k \ldots V_l \, v_m \ldots v_n \, V_o \ldots V_p)}. \tag{2.11}$$

Du fait de la normalisation sur les variables $V_o \dots V_p$, cette méthode de résolution d'une question peut être coûteuse (l'inférence bayésienne en général est *NP-difficile [Cooper 1990]*). Cependant différentes formes de simplification peuvent apparaître lors de l'inférence, grâce, en particulier, aux indépendances formulées dans la distribution conjointe.

c) Décision

Le résultat de l'inférence fournit une distribution de probabilité sur les variables recherchées. Cette distribution de probabilité résume les connaissances préalables du programmeur sur le problème et les informations apportées par des observations extéroceptives par exemple.

Dans le cadre de la robotique, cette distribution de probabilité porte typiquement sur les variables motrices. Afin de contrôler le robot, il convient de choisir une valeur pour ces variables. Il s'agit donc d'un problème de décision. De nombreuses stratégies sont imaginables, le plus courant est de choisir la valeur correspondant au maximum de probabilité, ou de tirer les valeurs aléatoirement selon la distribution obtenue.

2.2.5 Exemple : la localisation unidimensionnelle

Afin d'illustrer la notion de programme bayésien et son utilisation, nous allons utiliser le système minimal suivant : un robot unidimensionnel se déplaçant sur l'axe des abscisses dans l'intervalle $[-10, 10]$ et capable de mesurer sa distance à l'origine (la figure 2.2 donne un aperçu de ce système). Nous utiliserons plusieurs fois un système similaire pour illustrer certaines notions qui seront présentées par la suite. Avec ce système, la question que nous nous posons est la question classique de localisation : quelle est la position du robot sachant qu'il a mesuré une distance de 5 mètres. Pour répondre à cette question nous allons construire, étape par étape, un programme bayésien dont le résumé est donné par la figure 2.3.

a) Question

La question "quelle est la position du robot sachant la distance qu'il a mesurée" se traduit par la question probabiliste $P(X \mid Z)$.

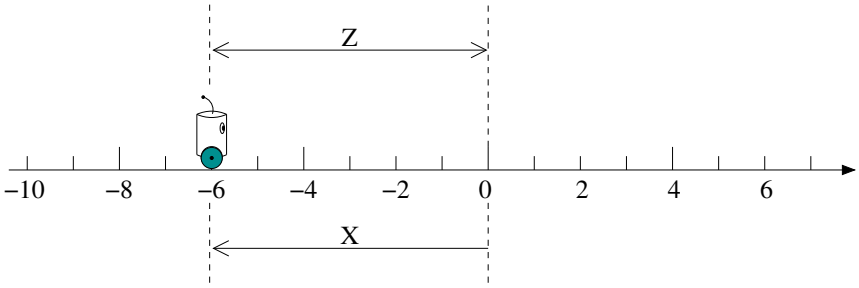
b) Spécification

*1.***Variables pertinentes** Au vu de la description de notre problème, le choix des deux variables pertinentes est assez immédiat :

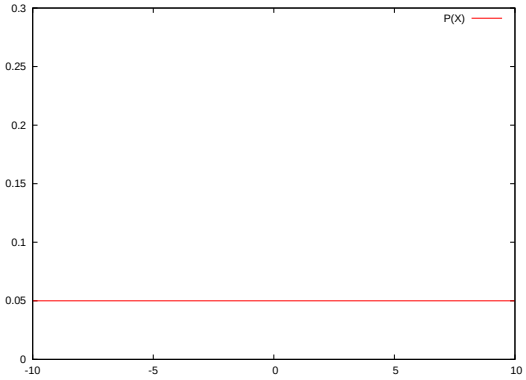
- la position X du robot sur l'axe des abscisses, variable réelle unidimensionnelle définie sur l'intervalle $[-10, +10]$;
- la distance Z mesurée par le robot, variable réelle positive unidimensionnelle.

*2.***Décomposition** A partir de ces deux variables, deux décompositions sont possibles :

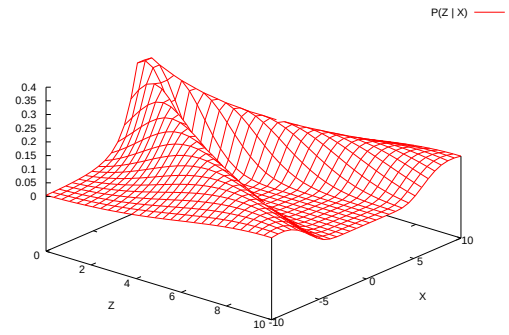
$$P(X\ Z) = P(X)P(Z \mid X) \text{ ou } P(X\ Z) = P(Z)P(X \mid Z)$$



Système minimal pour la localisation unidimensionnelle



Spécification de $P(X)$



Spécification de $P(Z \mid X)$

FIG. 2.2: Spécification du système minimal pour la localisation unidimensionnelle.

La première décomposition fait apparaître un terme appelé *modèle capteur* dans la littérature : $P(Z \mid [X = x])$. Ce terme exprime, sous la forme d'une distribution de probabilité, la mesure capteur attendue lorsque le robot se trouve dans une position x. C'est en définissant complètement ce terme que l'on exprimera complètement nos connaissances sur le capteur et sur le système {robot-capteur} en général.

La deuxième décomposition n'est pas souhaitable car elle fait apparaître la *question* que nous voulons poser à notre programme bayésien : $P(X \mid Z)$. Le but de notre programme étant de répondre à cette question, on peut supposer qu'on ne sait pas définir directement cette distribution et donc, qu'on ne peut l'utiliser dans la phase de spécification.

3.**Formes paramétriques** Nous devons définir ici les formes paramétriques associées aux distributions $P(X)$ et $P(Z \mid X)$. Comme nous n'avons aucun *a priori* sur la position du robot, nous spécifions $P(X)$ avec la forme paramétrique la moins riche en information (au sens de Shannon) : la distribution uniforme sur $[-10, +10]$. Quant à $P(Z \mid X)$, on la définit par une famille de distributions gaussiennes centrées sur $|X|$ et d'écart type $1 + k|X|$, k constante positive. Cette définition exprime non seulement la mesure prédite lorsque le robot se trouve à la position X, mais aussi la précision attendue : plus le robot est loin de l'origine, moins la mesure de distance est supposée être précise, ce qui s'exprime par une augmentation de l'écart type.

c) Identification

Le seul paramètre restant à identifier est le coefficient de proportionnalité k. Dans notre exemple, il sera fixé *a priori* à $k = \frac{1}{2}$.

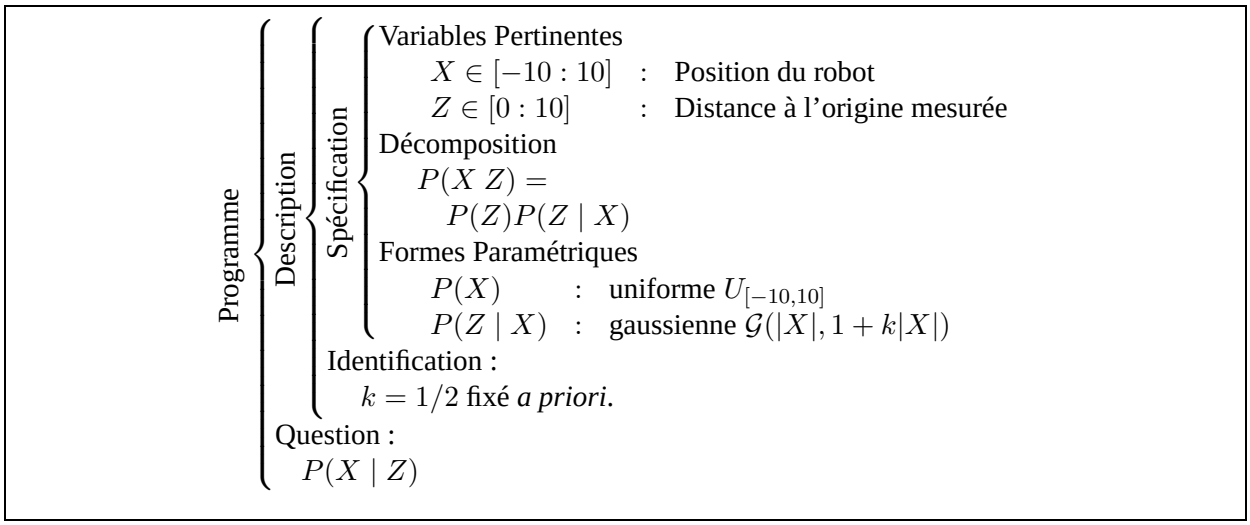


FIG. 2.3: Exemple de programme bayésien : la localisation unidimensionnelle.

La figure 2.4 donne la distribution résultant de l'observation d'une distance de 5 mètres : $P(X \mid [Z = 5])$. On constate que la distribution est bimodale, ce qui était attendu car la mesure de la distance à l'origine ne nous donne aucune information sur le signe de X. En observant cette courbe avec plus d'attention, on constate que les pics ne sont pas exactement centrés sur -5 et $+5$: les valeurs plus proches de l'origine sont en effet légèrement plus probables. Cette préférence est le résultat de notre modèle capteur gaussien et en particulier de l'expression de son écart-type : aussi étonnant que cela puisse paraître, il est plus probable d'avoir fait une mesure égale à 5 mètres alors que la position reelle était environ $\pm 4,50m$ que de faire cette même mesure en étant situé exactement à $\pm 5m$.

2.3 Collection d'outils bayésiens

Pour compléter ce chapitre sur la programmation bayésienne, nous allons maintenant présenter quelques outils dont nous aurons besoin par la suite. Notons que chacun est un cas particulier de programme bayésien.

2.3.1 Modèle de fusion

Définition 2 – Processus de fusion *Un processus de fusion de données cherche à estimer une variable X à partir d'un ensemble de $n \geq 2$ observations $\{Z_i\}_{i \in [1,n]}$.*

Intuitivement, un médecin fusionne les données issues de l'observation de divers indicateurs

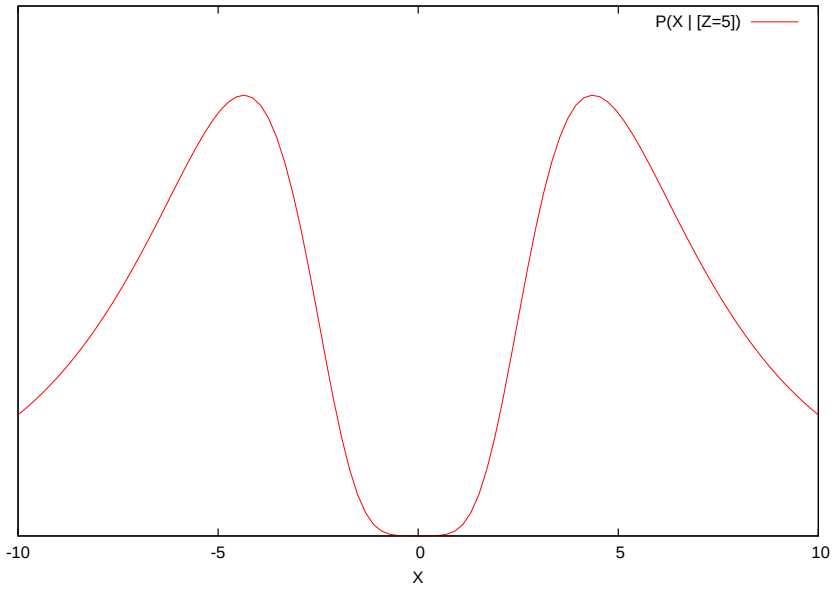


FIG. 2.4: Distribution de probabilité correspondant à la question $P(X \mid [Z = 5])$.

tels que la température, la respiration, le rythme cardiaque... pour déterminer la maladie dont souffre son patient.

L'avantage d'une estimation par fusion de données est l'augmentation de la qualité de la connaissance par l'accumulation des informations issues de chaque observation.

La figure 2.5 présente un programme bayésien correspondant à une fusion de donnée. On y remarque une hypothèse très courante pour ce type de programme : l'indépendance des observations connaissant la variable X.

Programme
- Description
  - Spécification
    - Variables Pertinentes
      - X : Variable recherchée
      - $\{Z_i\}_{i\in[1,n]}$: Variables observées
    - Décomposition
      - $P(X\ Z_1 \ldots Z_n) =$
      - $P(X) \prod_i P(Z_i \mid X)$
    - Formes Paramétriques
      - $P(X)$: Souvent uniforme
      - $P(Z_i \mid X)$: Modèle capteur
  - Identification :
    - selon le contexte.
- Question :
  - $P(X \mid Z_1 \ldots Z_n)$

FIG. 2.5: Modèle de fusion de données bayésienne

2.3.2 Le filtre bayésien

a) Définitions

Nous allons maintenant définir une notion essentielle pour le chapitre 4 : le filtre bayésien. Avant de définir un filtre bayésien, il est nécessaire de définir deux notions :

Définition 3 – Processus markovien *Un processus dépendant du temps et caractérisé par un état $x(t)$ est dit markovien d'ordre 1 si son état courant $x(t)$ est indépendant de son état avant l'instant $t-1$. En d'autres termes, $x(t)$ ne dépend que de $x(t-1)$.*

Définition 4 – Variable observable *Une variable V est dite observable s'il est "physiquement" possible de la mesurer. Au contraire, une variable V est dite cachée si on ne peut pas la mesurer directement. A titre d'exemple, sur un robot mobile, la vitesse des roues est observable, mais la position est cachée.*

Définition 5 – Filtre bayésien *Un filtre bayésien permet d'estimer l'état X_t d'un processus de Markov caché au cours du temps, en fonction des observations déjà réalisées $Z_0 \dots Z_t$.*

Programme
- Description
  - Spécification
    - Variables Pertinentes
      - $\{X_i\}_{0 \leq i \leq t}$: États du système au cours du temps
      - $\{Z_i\}_{0 \leq i \leq t}$: Observations réalisées
    - Décomposition
      - $P(X_0 \dots X_t\, Z_0 \dots Z_t) =$
      - $P(X_0)P(Z_0 \mid X_0)$
      - $P(X_1 \mid X_0)P(Z_1 \mid X_1)$
      - $\vdots$
      - $P(X_t \mid X_{t-1})P(Z_t \mid X_t)$
    - Formes Paramétriques
      - $P(X_0)$: *a priori* initial, peut être uniforme
      - $P(X_i \mid X_{i-1})$: Modèle de transition
      - $P(Z_i \mid X_i)$: Modèle capteur
  - Identification :
    - En fonction du contexte, en général fixée *a priori*.
- Question :
  - $P(X_t \mid Z_t \dots Z_0)$

FIG. 2.6: Exemple de programme bayésien : le filtre bayésien.

La figure 2.6 présente le programme bayésien correspondant à un filtre bayésien. Plusieurs remarques peuvent être faites par rapport à ce problème. Tout d'abord, la définition du filtre

bayésien dépend complètement de la définition de deux modèles :

Le modèle de transition $P(X_i \mid X_{i-1})$ permet de prévoir l'état du système à partir de l'état précédent. En général, pour un système robotique réel, le mouvement introduit de l'incertitude, ce qui s'exprime en *ne spécifiant pas* $P(X_i \mid X_{i-1})$ à l'aide d'une distribution de Dirac[1].

Le modèle capteur $P(Z_i \mid X_i)$ permet de prédire l'observation Z_i connaissant l'état X_i du système.

D'autre part, la complexité de l'expression analytique correspondant à la question du programme bayésien est très importante. En marginalisant sur les variables $X_{t-1} \dots X_0$, on obtient en effet l'expression ci-dessous :

$$\begin{aligned} P(X_t \mid Z_t \dots Z_0) &\propto \int_{X_{t-1}} \cdots \int_{X_0} P(X_t \dots X_0\, Z_t \dots Z_0) \\ P(X_t \mid Z_t \dots Z_0) &\propto P(Z_t \mid X_t) \int_{X_{t-1}} P(X_t \mid X_{t-1}) P(Z_{t-1} \mid X_{t-1}) \\ &\dots \int_{X_0} P(X_1 \mid X_0) P(Z_0 \mid X_0) P(X_0) \end{aligned} \tag{2.12}$$

Toutefois, la popularité du filtre bayésien ne vient pas de cette expression impossible à manipuler, mais plutôt de sa forme récursive. On constate en effet que l'on peut exprimer la question à l'instant t en fonction de la question à l'instant $t-1$:

$$P(X_t \mid Z_t \dots Z_0) \propto P(Z_t \mid X_t) \int_{X_{t-1}} P(X_t \mid X_{t-1}) P(X_{t-1} \mid Z_{t-1} \dots Z_0) \tag{2.13}$$

Grâce à cette formule de récurrence, on peut calculer la meilleure estimation possible de l'état après chaque observation en se rappelant seulement l'estimation à l'état précédent. Cette propriété résulte immédiatement d'une des hypothèses de départ de notre modèle : il s'agit d'un modèle de Markov.

Cette propriété permet aussi d'exprimer l'estimation de l'état du système après une transition et avant une observation :

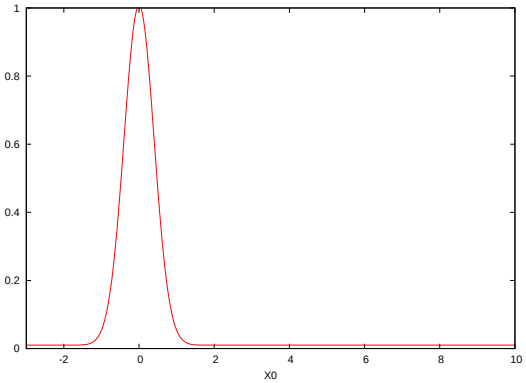
$$P(X_t \mid Z_{t-1} \dots Z_0) \propto \int_{X_{t-1}} P(X_t \mid X_{t-1}) P(X_{t-1} \mid Z_{t-1} \dots Z_0)$$

b) Illustration

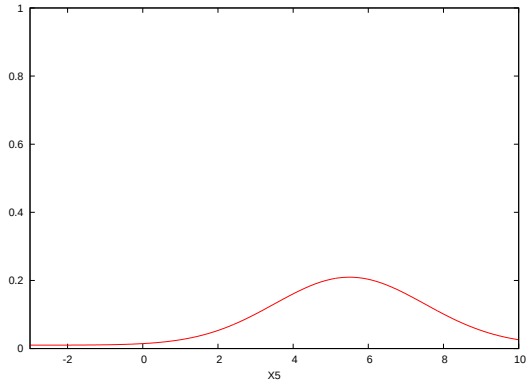
Pour illustrer le fonctionnement du filtre bayésien, nous reprenons l'exemple de notre robot unidimensionnel.

– Pour $t = 0$, figure 2.7.a, celui-ci se trouve en $X = 0$ avec une grande certitude, traduite par une distribution très piquée.

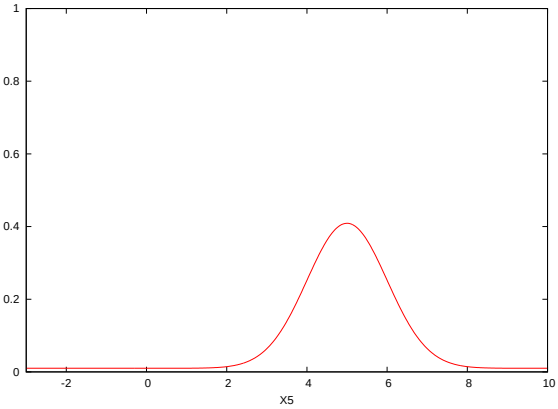
[1] Une distribution de Dirac est une distribution qui vaut 0 sur presque tout son domaine, sauf en un point.



a. État initial $P(X_0)$



b. État avant observation $P(X_1)$



c. État après observation $P(X_1 \mid Z_1)$

FIG. 2.7: Fonctionnement d'un filtre bayésien.

– Après un déplacement de 5 unités, le robot est prêt à faire un nouvelle mesure de sa position par rapport à l'origine. La figure 2.7.b montre la distribution très aplatie résultant de la question :

$$P(X_1) \propto \int_{X_0} P(X_1 \mid X_0) P(X_0)$$

Cet aplatissement est le résultat de l'incertitude introduite par le mouvement.
– Après la mesure, figure 2.7.c, la distribution s'est resserrée, ce qui traduit l'apport d'information dû à une observation.

Pour conclure, le filtre bayésien est une structure générique de programme bayésien, une sorte de méta-programme, qui permet d'estimer de façon très efficace l'état d'un système de Markov caché.

Finalement, on ne peut terminer cette section sur les filtres bayésiens sans en évoquer deux spécialisations particulièrement connue : le filtre de Kalman et le filtre à particules.

2.3.3 Le filtre de Kalman

Définition 6 – Filtre de Kalman *Le filtre de Kalman* [Crowley 1989, Kalman 1960] *est un filtre bayésien dont toutes les formes paramétriques sont des distributions gaussiennes et dont les modèles de transition et d'observation résultent de fonctions linéaires.*

A titre d'exemple, reprenons notre robot unidimensionnel et supposons maintenant qu'il est équipé d'un capteur de position. L'état E_t que nous allons estimer avec un filtre de Kalman est non seulement la position du robot, mais aussi sa vitesse. On note $X_t = {}^t[x_t\ v_t]$ et Z_t l'observation au temps t. Avec ces notations, on peut définir les modèles de transition et d'observation du système, et constater qu'ils sont linéaires :

$$X_{t+1} = \begin{pmatrix} x_{t+1} \\ v_{t+1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_t \\ v_t \end{pmatrix} = A.X_t$$

$$Z_t = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x_t \\ v_t \end{pmatrix} = C.X_t$$

Pour décrire les incertitudes du système, on pose d'une part $X_{t+1} = A.X_t + v$, où v est une variable aléatoire gaussienne centrée en 0 de covariance R. D'autre part $Z_t = C.X_t + w$, où w est une variable aléatoire gaussienne centrée en 0 de covariance Q.

La force du filtre de Kalman vient du fait que, dans ces conditions, si l'état initial du système est décrit par une variable gaussienne, la distribution $P(X_t \mid Z_t \dots Z_0)$ est gaussienne quel que soit t et ses paramètres – moyenne $\bar{X}_t$ et covariance P_t – peuvent être exprimés directement par les formules de récurrence suivantes :

$$\begin{aligned} X_{pred} &= A.\bar{X}_t \\ P_{pred} &= A.P_t.\,{}^tA + Q \\ K &= P_{pred}.\,{}^tC\,.(C.P_{pred}.\,{}^tC + R)^{-1} \\ \bar{X}_{t+1} &= X_{pred} - K(B.X_{pred} - Z_{t+1}) \\ P_{t+1} &= (I - K.B)P_{pred} \end{aligned}$$

Dans les équations ci-dessus, X_{pred} et P_{pred} sont les paramètres prédits à partir du modèle de transition seul, K est appelé gain de *Kalman* et $\bar{X}_{t+1}$ et P_{t+1} sont les paramètres résultant de la fusion de de la prédiction et du processus d'observation.

*1.***Filtre de Kalman Étendu** La linéarité des modèles est une hypothèse de base du filtre de Kalman. Il est possible de manipuler des modèles non-linéaires à condition de les linéariser préalablement. On parle alors de *Filtre de Kalman Étendu* (EKF). Alors que les formules du filtre de Kalman fournissent exactement le résultat du filtre bayésien général (eq. 2.13), l'EKF n'en fournit qu'une approximation. Pour que cette approximation ne soit pas trop grossière, il est nécessaire que l'approximation linéaire des modèles soit suffisamment précise.

2.3.4 Le filtre à particules

Le filtre à particules *[Arulampalam et al. 2002]* ou filtre à condensation, est lui aussi un filtre bayésien. Contrairement au filtre de Kalman, il ne pose aucune contrainte sur les formes paramétriques mais spécifie que celles-ci seront représentées par un ensemble de particules. Ce nuage de particules constitue un échantillonage de la distribution de probabilité qu'il représente.

Dans le cas d'un filtre estimant la localisation d'un véhicule automobile dans le plan, les particules seront placées dans un espace de dimension 3 (2 coordonnées de position dans le plan et une variable d'orientation).

Intuitivement, on peut voir chaque particule comme une hypothèse de localisation pour le véhicule. Les parties de l'espace où de nombreuses particules sont présentes correspondent aux localisations très probables alors que les espaces vides de particules sont considérés très improbables.

Le filtre à particules réalise son inférence en trois étapes :

1. Prédiction : pour calculer $P(X_t \mid Z_{t-1} \dots Z_0)$ à partir de $P(X_{t-1} \mid Z_{t-1} \dots Z_0)$, le modèle de transition est appliqué directement à chaque particule.
2. Observation : après une observation Z_t, on calcule, pour chaque particule p_i, un poids égal à $\alpha(i) = P(Z_t \mid X_t = p_i)$.
3. Ré-échantillonage : on sélectionne les particules de la nouvelle génération par tirage aléatoire de l'ancienne génération selon une distribution $P(i)$ proportionnelle à $\alpha(i)$.

Par rapport au filtre de Kalman, le filtre à particules fournit une représentation moins précise des distributions de probabilité sur X_t. Cependant, il a l'avantage de ne pas nécessiter de modèle linéaire ou linéarisable, et surtout, d'être capable de représenter des distributions quelconques, éventuellement unimodales.

2.3.5 Modèle de diagnostic

Nous souhaitons terminer ce chapitre sur la programmation bayésienne en présentant un type de modèle que nous avons introduit pour modéliser des relations de cohérence entre variables : les modèles de diagnostic.

Définition 7 – Variable de diagnostic *Une variable de diagnostic $\mathbb{I}_A^B$ est une variable booléenne qui indique si la variable A est* cohérente *avec la variable B. Les variables A et B seront nommées variables diagnostiquées par $\mathbb{I}_A^B$.*

Définition 8 – Modèle de diagnostic *Un modèle de diagnostic sur A et B est constitué d'une variable de diagnostic I_A^B et d'une distribution conjointe $P(\mathbb{I}_A^B\ A\ B)$ se décomposant en*

$$P(\mathbb{I}_A^B\ A\ B) = P(A)P(B)P(\mathbb{I}_A^B \mid A\ B)$$

avec $P(A)$ et $P(B)$ uniformes.

Il peut paraître surprenant que les variable A et B apparaissent comme des variables indépendantes dans cette distribution. Cela signifie deux choses :

- D'une part, toute la connaissance sur la relation entre A et B est contenue dans la distribution $P(\mathit{I}_A^B \mid A\ B)$.
- D'autre part, sans savoir si A et B sont cohérentes l'une avec l'autre (i.e. I_A^B non spécifiée), il n'y a aucune raison *a priori* pour que ces variables dépendent l'une de l'autre.

a) Exemple

Si on reprend le cas d'un robot unidimensionnel capable de mesurer sa distance à l'origine, on peut définir un modèle de diagnostic pour la position du robot X et la distance mesurée Z :

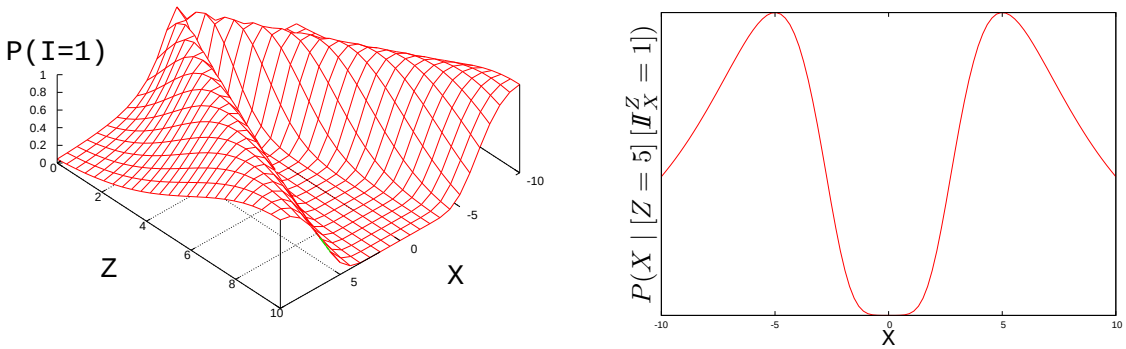
$$P(X\ Z\ \mathit{I}_X^Z) = P(X)P(Z)P(\mathit{I}_X^Z \mid X\ Z)$$

en posant

$$P([\mathit{I}_X^Z = 1] \mid X\ Z) = e^{-\frac{1}{2}\left(\frac{Z-|X|}{k|X|}\right)^2}$$

$P([\mathit{I}_X^Z = 1] \mid X\ Z)$ vaut 1 lorsque $Z = |X|$, c'est à dire lorsque l'observation Z correspond exactement à ce qu'on attend du capteur. Au contraire, plus Z est différent de $|X|$ plus $P([\mathit{I}_X^Z = 1] \mid X\ Z)$ est proche de 0. Cette probabilité traduit donc un diagnostic de cohérence entre l'observation réelle et son modèle. La partie gauche de la figure 2.8 présente la valeur de la probabilité $P([\mathit{I}_X^Z = 1] \mid X\ Z)$ en fonction X et Z.

Dans un contexte de localisation, on s'intéressera à la question $P(X \mid Z\ [\mathit{I}_X^Z = 1])$.



$P([\mathit{I}_X^Z = 1] \mid X\ Z)$ $\qquad$ $P(X \mid [Z = 5]\ [\mathit{I}_X^Z = 1])$

FIG. 2.8: Modèles de diagnostic : illustration dans le cas d'un robot unidimensionnel.

A droite de la figure 2.8, on peut observer le résultat de la question $P(X \mid [Z = 5]\,[\mathit{I}_X^Z = 1])$. On constate que l'on retrouve le même graphe que dans la figure 2.4. Dans la plupart des cas, un modèle de diagnostic exprimera les mêmes connaissances qu'un modèle classique (cf. fig. 2.3). Nous montrerons la spécificité et l'intérêt des modèles de diagnostic dans la section 4.3.2.

b) Intérêt des modèles de diagnostic

*1.***Symétrie** Tout d'abord, la spécification du modèle est complètement symétrique par rapport aux variables diagnostiquées. Ce n'est pas vrai en général avec une spécification du type $P(A)P(B \mid A)$. En effet, même si l'on choisit de ne pas spécifier d'*a priori* sur A en posant $P(A)$ uniforme, $P(B) = \int_A P(A)P(B \mid A)$ n'est pas uniforme en général. De par la modélisation, on a donc introduit un *a priori* sur B. Les variables de diagnostic sont particulièrement utiles dans les cas où l'on souhaite éviter cet *a priori* .

*2.***Expression de l'ignorance et robustesse** Nous montrerons dans la section 4.3 que l'utilisation d'un modèle de diagnostic permet une expression de l'ignorance plus satisfaisante, ce qui permet d'écrire un processus de fusion de données plus robuste aux données incohérentes.

*3.***Efficacité calculatoire** Finalement nous montrons dans *[Pradalier et al. 2003a;b]* et dans l'annexe A qu'une fusion de données à partir de modèles de diagnostic peut toujours s'écrire comme un produit de distributions, ce qui est intéressant en terme de coût calculatoire. Cette propriété sera illustrée dans les sections 6.3 et 6.4.

2.4 Conclusion sur la programmation bayésienne

Cette méthode de programmation a initialement été appliquée à la programmation de tâches "réflexes" simples sur des robots mobiles *[Coué & Bessière 2001, Lebeltel 1999]*, et à la CAO-Robotique *[Mekhnacha 1999a, Mekhnacha et al. 2001]*. Depuis, son utilisation s'est étendue à d'autres domaines de la robotique, comme la programmation des bras manipulateurs *[García-Ramírez 2003]*, et la localisation *[Diard 2003b]*.

Dans cette thèse, la programmation bayésienne sera utilisée comme un outil. La plupart de nos raisonnements seront exprimés dans ce formalisme. Au chapitre 7, nous en définirons une extension inspirée des concepts de la programmation objet. Cette extension nous permettra de structurer les applications de robotique intégrée que nous présenterons dans ce chapitre. Finalement l'annexe A présentera le résultat de nos réflexions sur l'expression des problèmes de fusion de données dans le cadre de la programmation bayésienne.

Chapitre 3

Caractérisation d'une tâche de navigation

3.1 Introduction

Lorsque l'on est confronté au problème de la navigation intentionnelle d'un robot mobile dans un environnement, la première question qui se pose naturellement est la caractérisation de cette tâche.

Intuitivement, on parlera de navigation intentionnelle lorsqu'un robot se déplace de façon à atteindre un objectif. Pour éclairer cette définition intuitive, il va être nécessaire d'expliciter la notion de déplacement.

Dans la suite de ce chapitre, nous allons commencer par rappeler les différents espaces utilisés dans le contexte de la robotique mobile. Nous verrons ensuite qu'une tâche de navigation peut être représentée soit sous la forme d'un comportement à appliquer, soit sous la forme d'un trajet à parcourir.

3.2 Espaces pour la navigation

*1.***Espace de tâche, ou espace de travail, noté** $\mathbb{W}$ Cet espace est celui dans lequel l'environnement de travail du robot est défini, et en particulier, les obstacles. Pour un véhicule automobile ou un robot évoluant dans les couloirs d'un laboratoire, l'espace de tâche est généralement considéré plan et bidimensionnelle. Un humain, dans ses tâches quotidiennes, évolue dans un espace de travail tridimensionnel dans lequel il se déplace en évitant les obstacles et en manipulant des objets.

*2.***Espace des perceptions, noté** $\mathbb{O}$ Les mesures fournies par les capteurs embarqués sur le robot sont définies dans cet espace. Pour un robot équipé d'une caméra, l'espace d'observation est un espace de dimension très élevé dans lequel les images sont définies. Une des problématiques de la vision par ordinateur est de simplifier cet espace en sélectionnant et en condensant l'information pertinente dans un espace de dimension réduite.

*3.***Espace des configurations, noté** $\mathbb{C}$ Cet espace, introduit dans le domaine de la planification de trajectoire *[Latombe 1995]*, permet de décrire l'état complet du robot. Pour un bras manipulateur, il s'agira de toutes les positions articulaires. Pour un véhicule automobile tel que notre plate-forme expérimentale, une configuration est en général définie par la position d'un point de référence du véhicule et par son orientation. On y ajoute parfois l'angle de braquage.

*4.***Espace de contrôle, noté** $\mathbb{U}$ Cet espace est celui dans lequel les commandes pouvant être appliquées sur le système sont définies. Pour un véhicule automobile, il peut s'agir, par exemple, de l'espace du couple $<$angle de braquage,vitesse$>$.

Remarque : Dans le contexte d'une tâche de navigation, il arrive que l'on ne manipule pas directement un des espaces de base ci-dessus, mais que l'on utilise une différence de valeurs, des dérivées ou encore des primitives. Pour ne pas alourdir les notations, nous assimilerons l'espace de ces valeurs à leur espace de base, et nous n'utiliserons que les quatre symboles $\mathbb{W}, \mathbb{O}, \mathbb{C}$ et $\mathbb{U}$.

3.2.1 Mode de représentation d'une tâche de navigation

Commençons par noter D une variable prenant, selon les cas, sa valeur parmi les espaces $\mathbb{W}, \mathbb{C}$ ou $\mathbb{O}$. En utilisant cette variable, nous allons maintenant considérer la notion de déplacement d'un robot mobile. Deux compréhensions sont possibles :

- Soit on décrit le déplacement par une méthode qui associe une commande à tout point de D. Nous désignerons cette représentation par le terme *comportement.*
- Soit on décrit le déplacement par une séquence de point dans D. Nous utiliserons alors le terme *trajet*. La nature exacte du trajet dépendra de la nature des index de la séquence.

Dans les deux sections suivantes, nous allons examiner plus en détails ces deux hypothèses de représentation, et en particulier, expliciter l'influence du choix de la valeur de D sur la représentation.

3.3 Les comportements

Définition 9 – Comportement *Un* comportement C *est une fonction définie sur l'espace* D *et permettant de générer des commandes. Formellement :*

$$C : D \longrightarrow \mathbb{U}$$

3.3.1 Exemples de comportements selon la valeur de D

Selon la nature de l'espace D, on peut distinguer plusieurs types de comportement.

– Un comportement de suivi de trajectoire (section 6.3) permet de générer des commandes à partir d'une erreur de suivi, sous la forme d'une différence de configurations : $D = \mathbb{C}$.
– Un comportement d'évitement d'obstacles (section 6.4) ou de suivi de mur, génère ses commandes directement à partir des observations. Dans ce cas, on a $D = \mathbb{O}$. De façon générique, un comportement qui réagit uniquement aux données perceptives est appelé *comportement réactif [Arkin 1991]*.
– Finalement si on choisit $D = \mathbb{W}$, on définit un comportement qui associe à chaque position du robot une action. On a donc un plan de navigation pour tout l'espace. Les techniques utilisant des fonctions de potentiel *[Khatib 1985]* peuvent fournir ce type de comportement.

3.3.2 Principaux modes de construction

Parmi l'infinité de possibilités de définition de C, deux approches sont couramment utilisées.

a) Apprentissage non-supervisé

Parmi les approches non-supervisées, la plus représentative et la plus utilisée est l'apprentissage par renforcement (cf. livre de Sutton *[Sutton & Barto 1998]*). Le principe de ce type d'apprentissage est d'explorer l'espace des comportements de façon à en construire un qui soit optimal par rapport à un critère spécifique. A titre d'exemple, il est possible d'utiliser un apprentissage par renforcement, dans un environnement de type grille, pour construire un comportement permettant d'atteindre un but en minimisant les risques de collisions (cf. figure 3.1).

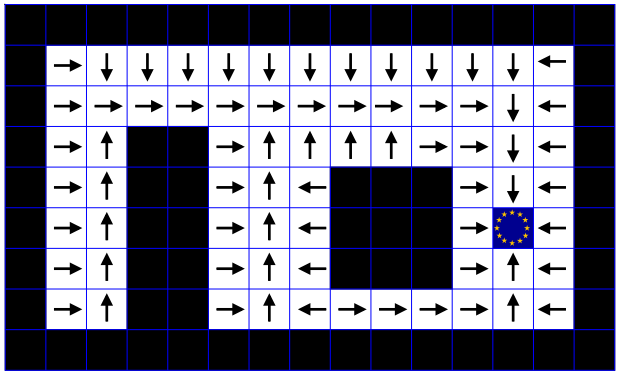


FIG. 3.1: Comportement (mouvement en fonction de la position) dans un environnement grille : le but est d'atteindre le drapeau européen en minimisant les risques de collisions.

Pour des espaces d'état et de contrôle plus complexes, la complexité de l'espace de fonctions de comportement rend la mise en oeuvre directe des algorithmes d'apprentissage par renforcement difficile. Il est alors nécessaire d'utiliser de représentations synthétiques approchées de ces fonctions (réseaux de neurones par exemple *[Sutton & Barto 1998]*) ou de choisir un ensemble restreint de fonctions de comportement (approche présentée dans *[Bagnell & Schneider 2001]* pour contrôler un hélicoptère).

b) Apprentissage supervisé

Contrairement à l'apprentissage non-supervisé, l'apprentissage supervisé n'est pas un processus d'optimisation, mais plutôt un processus d'imitation. Son principe est le suivant : un opérateur pilote le robot pour réaliser une tâche. Pendant ce temps, le robot enregistre son état, ses ordres moteurs et éventuellement ses perceptions. Il enregistre donc un échantillonnage de la fonction de comportement qu'on désire lui faire apprendre. Un algorithme spécifique est ensuite utilisé pour construire une fonction de comportement aussi proche que possible de la fonction enregistrée. Parmi les techniques existantes, on peut citer les méthodes à base de réseaux de neurones *[Hertz et al. 1991, Hérault & Jutten 1994]* ou encore les techniques d'apprentissage probabiliste telles que présentées dans *[Lebeltel et al. 2003]*.

3.4 Les trajets

Définition 10 – Trajet *Un* trajet T *est une séquence de points de* D*. Formellement, il s'agit d'une fonction allant d'un espace d'index* I *vers l'espace* D*.*

$$T : I \longrightarrow D$$

3.4.1 Exemples de trajets selon les valeurs de I et D

Définition 11 – Route, chemin, trajectoire *On distingue différents types de trajet selon la nature de* I *:*

- $I \subset \mathbb{N}$ *: le trajet est une* route*, c'est à dire, une liste de points dans* D*.*
- $I = [0, 1]$ *à une isotopie près : le trajet est juste une courbe de* D*, sans référence temporelle, appelée un* chemin*.*
- $I \subset \mathbb{T}$ *: le trajet est une séquence de points de* D *indexée par le temps* $\mathbb{T}$*. Nous utiliserons alors le terme* trajectoire*.*

Trajet sensorimoteur Si on choisit $D = \mathbb{O}$, alors on définit un trajet par une séquence de perception. On parlera alors de trajet sensoriel. De plus, si on associe à ces observations des ordres moteurs, $D = \mathbb{O} \times \mathbb{U}$, alors il s'agira d'un trajet *sensorimoteur*. Un aspect particulièrement séduisant des trajets sensorimoteurs est la possibilité de les considérer comme une hypothèse sur la manière dont les entités biologiques mémorisent et se représentent un trajet. Dans le chapitre 7, nous montrerons qu'une telle représentation est, par ailleurs, tout à fait utilisable pour la navigation d'un robot mobile.

Imbrication de tâches de navigation En observant cette définition d'un trajet, nous constatons qu'il s'agit d'une représentation du mouvement qui n'est pas directement en relation avec l'espace de contrôle $\mathbb{U}$. Or pour déplacer le robot le long du trajet, il faudra nécessairement appliquer – et donc générer – des commandes motrices. La fonction qui générera ces commandes fera se déplacer le robot avec pour objectif de rester sur le trajet, il s'agira donc d'une seconde tâche de navigation intentionnelle. Cette remarque met en évidence la nécessité d'utiliser plusieurs tâches de navigation imbriquées pour pouvoir réaliser une tâche complexe.

3.4.2 Principaux modes de construction

Comme pour les comportements, la définition des espaces de départ et d'arrivée n'est que la première étape de la définition d'un trajet. La seconde étape est la construction effective de la fonction T.

Dans la suite de ce document (chapitres 6 et 7), nous considérerons que la description du trajet à réaliser est disponible et que la question de sa construction est résolue.

a) Apprentissage par mémorisation d'un exemple

La façon la plus simple de mémoriser un trajet est sans doute d'en observer un exemple. Considérons tout d'abord le cas d'un trajet dans $\mathbb{W}$ ou $\mathbb{C}$. Dans ce cas, si le robot est équipé d'un système de localisation, il peut apprendre facilement une trajectoire. Il suffit de piloter le robot sur le trajet qu'on souhaite lui faire apprendre pendant que celui-ci enregistre sa position au cours du temps, ainsi que, éventuellement, les commandes appliquées. La représentation interne peut ensuite être retravaillée en construisant une représentation synthétique de la trajectoire : segments de droite, arcs de cercle, splines.

Dans le cas d'un trajet sensoriel, une stratégie similaire est applicable : on pilote le robot sur la trajectoire à apprendre pendant qu'il enregistre ses perceptions sensorimotrices. On obtient une représentation interne sous forme de film sensoriel dont le volume peut rapidement devenir très important (cas d'un capteur vidéo, par exemple). On peut alors envisager de construire une représentation synthétique qui résume le film par quelques lieux sensoriels représentatifs. Ceci est alors équivalent à la construction d'une représentation topologique du trajet dans l'espace sensorimoteur *[Chatila & Laumond 1985, Tomatis et al. 2003]*.

Avec un apprentissage par l'exemple, on a l'avantage de n'avoir besoin d'aucune représentation initiale de l'environnement (aussi bien dans le cas géométrique que dans le cas perceptif). Lorsqu'on envisage de plus une mémorisation sensorimotrice, on n'a même plus besoin d'un système de localisation. On a alors un système minimal en termes de compétences géométriques requises.

Cependant, nous n'oublions pas le défaut majeur de ces approches : elles ne représentent qu'un trajet et sont incapables de construire un trajet entre deux points quelconques.

b) Planification géométrique

On se place ici dans le cas où D est soit l'espace de tâche $\mathbb{W}$, soit l'espace des configurations $\mathbb{C}$. La construction autonome d'un trajet dans D est une tâche de planification *[Latombe 1995, P. et al. 1995, Scheuer & Fraichard 1997, ...]*. Pour tous ces algorithmes, les entrées sont les suivantes : d'une part une représentation interne aussi complète que possible de D, en général sous forme de polygones ou de grille d'occupation, et d'autre part de deux points particuliers dans D, la "position" initiale I et le but B. L'objectif d'un algorithme de planification est alors de construire une séquence de points de D reliant I à B. Pour construire cette séquence, il existe deux grandes familles d'algorithmes : les déterministes, qui explorent D de manière méthodique et prévisible *[Latombe 1995, Scheuer & Fraichard 1997]*, et ceux qui utilisent un échantillonnage aléatoire de l'espace *[P. et al. 1995]*. La clef de ces algorithmes est en général une procédure, appelée "planificateur local", permettant de déterminer s'il existe une trajectoire simple entre deux points de D : ligne droite, application de commandes constantes...

La tâche de planification devient en général plus complexe lorsqu'on considère des systèmes non-holonomes. Sans entrer dans les détails, un système holonome est un système qui, à partir d'un point particulier P de D peut atteindre tout point du voisinage de P. Au contraire, un système non-holonome possède des contraintes additionnelles (roulement sans glissement, par exemple) qui rendent une partie du voisinage inatteignable directement. Pour illustrer cette notion, il suffit d'imaginer une voiture : ce véhicule ne peut réaliser de mouvements latéraux, ce qui amène chaque apprenti conducteur à devoir maîtriser la manoeuvre difficile de garage en créneau. Parmi les méthodes permettant de planifier des trajectoires réalisables pour un système non-holonome, on peut citer celle utilisant la notion de platitude différentielle *[Hermosillo 2003, Sekhavat & Laumond 1998]* car elles sont utilisées dans nos expérimentations (chapitre 7).

c) Planification sensorimotrice

On pourrait s'attendre à ce que les approches de planification présentées ci-dessus ne puissent définir que des trajets géométriques. Cependant, avec un modèle des capteurs embarqués sur le robot, il est tout à fait envisageable de déduire une trajectoire sensorimotrice à partir de la trajectoire géométrique.

Formellement, ce type de planification peut être mis en oeuvre de la façon suivante : supposons que l'on soit capable de planifier un chemin $\mathcal{T}_g : I \rightarrow \mathbb{C}$ dans l'espace des configurations, que l'on dispose d'un modèle de l'environnement et d'un modèle de capteur $H : \mathbb{C} \rightarrow \mathbb{O}$, capable de prédire les perceptions étant donné un point de vue (un tel modèle est courant en robotique mobile : on peut citer *[Chatila & Laumond 1985, Crowley 1989]* pour des exemples précurseurs).

On constate alors que, par composition, on a immédiatement un trajet sensoriel : $\mathcal{T}_s : \mathbb{T} \rightarrow \mathbb{O} = H \circ \mathcal{T}_g$. En général, le résultat utilisable d'une planification contient des informations motrices de référence à appliquer le long de la trajectoire (vitesse ou angle de braquage dans notre cas). Il s'agit donc d'une fonction $\mathcal{T}_{gm} : \mathbb{T} \rightarrow \mathbb{C} \times \mathbb{U}$. En composant cette fonction avec $(H \times Id) : \mathbb{C} \times \mathbb{U} \rightarrow \mathbb{O} \times \mathbb{U}$, on obtient simplement une trajectoire sensorimotrice :

$$\mathcal{T}_{sm} : \mathbb{T} \rightarrow \mathbb{O} \times \mathbb{U} = (H \times Id) \circ \mathcal{T}_{gm}$$

La figure 3.2 illustre la construction d'une trajectoire sensorimotrice à partir d'une trajectoire géométrique. La partie gauche de la figure donne un exemple de trajectoire géométrico-motrice sous sa forme fonctionnelle : $T : \mathbb{T} \rightarrow \mathbb{W} \times \mathbb{U}$. Le graphe du milieu présente l'espace de travail dans lequel va évoluer le robot (symbolisé par un triangle inclus dans un rectangle), les amers qui y sont présents (représentés par les cercles jaunes) et le chemin déduit de la trajectoire géométrico-motrice. Finalement, la partie droite de la figure présente la trajectoire sensorimotrice, fonction du temps dans l'espace des perceptions et des commandes motrices, pouvant être déduite de la connaissance conjointe du modèle de l'environnement et de la trajectoire géometrico-motrice.

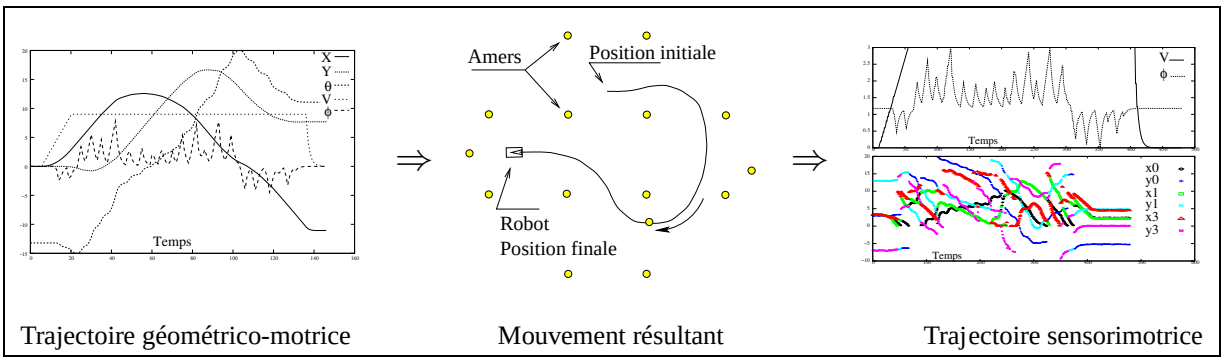


FIG. 3.2: Planification sensorimotrice.

3.5 Conclusion

Ce chapitre nous a permis d'expliciter notre compréhension de la notion de tâche de navigation. Il s'agit donc d'un objet dual, pouvant être défini soit sous la forme d'un comportement (fonction génératrice de commandes), soit sous la forme d'un trajet (séquence de points dans un espace particulier).

Toutefois, en définissant ainsi une tâche de navigation, on masque un aspect important d'une application robotique à la fois complète et complexe : le besoin de combiner plusieurs types de navigation pour atteindre l'objectif. Par exemple, la réalisation d'une trajectoire planifiée dans l'espace des configurations, telle que celle présentée au chapitre 7, demande de connaître non seulement la trajectoire ($\mathbb{T} \rightarrow \mathbb{C}$), mais aussi un comportement de suivi ($\mathbb{C} \rightarrow \mathbb{U}$) pour générer effectivement le déplacement et un comportement d'évitement d'obstacles ($\mathbb{O} \rightarrow \mathbb{U}$) pour garantir la sécurité du robot et de son environnement.

Chapitre 4

Localisation

4.1 Introduction

Ce chapitre apporte une réponse partielle au problème de la réalisation d'une tâche de navigation intentionnelle. Pour accomplir une telle tâche, il est d'abord nécessaire que le système autonome soit capable d'analyser l'exécution de son mouvement et d'en vérifier le bon déroulement. C'est cette compétence que nous appellerons, de manière générique, *localisation*.

D'un point de vue plus abstrait, on peut définir la localisation comme la mise en relation de perceptions d'un environnement avec un modèle de cet environnement de façon à estimer une grandeur décrivant l'état du système.

La première question qui se pose alors est : "Que va-t-on percevoir dans l'environnement ?" Comme en témoignent de nombreuses publications(*[Madhavan et al. 2000, Schmidt 1996]…*), la question du choix et de l'extraction de primitives pertinentes à partir de données brutes est un problème difficile dans le cas général. Afin de nous focaliser sur la problématique de la localisation, nous avons choisi, d'une part, de ne traiter que le cas de la localisation par rapport à des amers ponctuels, et d'autre part, d'installer des amers artificiels aisément détectables dans notre environnement.

Ce problème étant résolu, se pose alors le problème du choix entre deux types de localisation :

- D'une part, une localisation par rapport à une carte d'amers que nous traiterons dans les sections 4.2 et 4.3 en mettant l'accent sur l'importance du problème d'identification des observations.
- Et d'autre part une localisation sans carte (section 4.4) qui nous permet de n'utiliser qu'un modèle sensorimoteur de l'environnement.

Quel que soit le type de localisation choisi, il nous semble essentiel que, pendant la réalisation du mouvement, le système prenne du recul par rapport à son exécution et estime sa confiance en lui. Nous proposons donc un algorithme (section 4.5) permettant d'estimer cette grandeur. Intuitivement, nous définissons la confiance en soi comme la conséquence de la cohérence entre l'environnement observé et les perception attendues, ces dernières étant construites à partir de la représentation interne de l'espace et des hypothèses de localisation.

4.1.1 Contexte applicatif et contraintes associées

Sauf mention contraire explicite, les résultats que nous présenterons dans ce chapitre ont été développés dans le contexte applicatif suivant :

*1.***Robot et environnement** Le robot que nous considérons est un véhicule autonome de type voiture évoluant dans un environnement extérieur de type urbain. Cet environnement sera supposé contenir des amers en nombre suffisant pour que le capteur embarqué sur le robot puisse en détecter plusieurs en une mesure (cf. *[Pradalier & Sekhavat 2002b]* pour une réflexion sur la conception d'un environnement de travail adéquat).

*2.***Amers** Nous avons choisi d'installer des amers artificiels dans notre environnement de travail, de façon à éliminer le problème de la détection d'amers naturels. Ces amers sont des cylindres réfléchissants, tous identiques, spécialement adaptés à notre capteur. De cette manière, nous obtenons des mesures précises et sûres de leurs positions dans le repère du capteur. Toutefois, ces mesures ne sont pas identifiées.

*3.***Capteurs** Le capteur choisi est un télémètre laser à balayage de type Sick (cf. 7.3.2) sur lequel un détecteur d'amers a été implanté.

*4.***Variables** Formellement, la position du robot sera décrite par la position (x, y) de son point de référence et son orientation θ, dans un repère fixe. Pour la simplicité de l'exposé, nous considérerons que la position du capteur peut être décrite par les mêmes variables (x, y, θ) dans ce repère. La position des amers sera elle aussi décrite dans ce repère sous la forme d'un ensemble $L = \{l_i = (x_i, y_i), i = 1..p\}$.

Par ailleurs, les observations données par le capteur seront exprimées dans un repère mobile lié au capteur (i.e. centré sur le capteur et orienté selon l'orientation du robot) sous la forme d'un ensemble de points en coordonnées polaires : $Z = \{z_j = (\rho_j, \alpha_j), j = 1..n\}$. La figure 4.1 illustre ces variables et leurs relations.

4.2 Localisation par triangulation

> **amer** *n. m. MAR* Tout point des côtes très visible (clocher, balise, etc.), porté sur une carte, servant de repère pour la navigation.
>
> Dictionnaire Hachette

L'approche la plus courante (*[Betke & Gurvits 1994, Greiner & Isukapalli 1994, ...]*) au problème de localisation d'un robot mobile est issue de l'héritage de la navigation à la voile. Quiconque a eu l'occasion de naviguer sur un voilier de plaisance avant la percée du GPS[1] a sans doute pu observer ou pratiquer le "calcul du point". Pour ce faire, le marin relève le site de

[1] Global Positioning System.

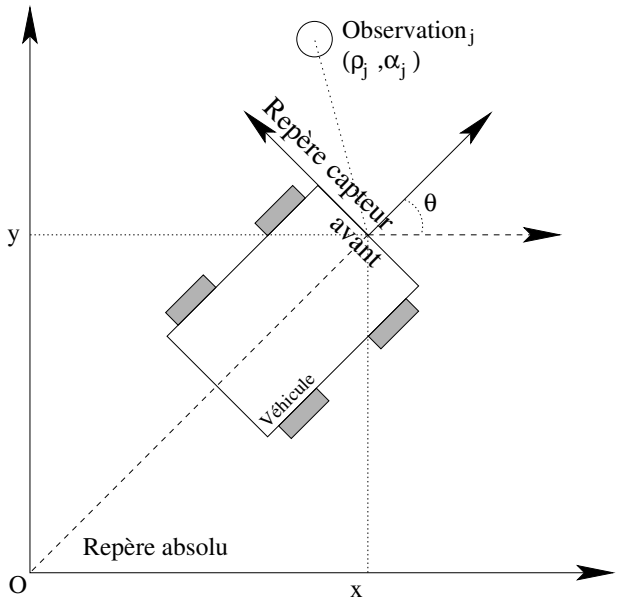


FIG. 4.1: Repères pour la localisation.

quelques amers connus à l'aide d'un compas de relèvement et, par une construction géométrique qui se résume à l'intersection d'autant de droites que d'amers, détermine sa position sur la carte. La partie gauche de la figure 4.2 illustre cette construction.

Pour réaliser un système de **localisation par triangulation** sur un système robotique, nous avons besoin des trois étapes suivantes :

1. Identification des amers observés.
2. Mesures de grandeurs pertinentes sur des amers dont la position est connue. De telles grandeurs peuvent être, par exemple, la distance, le site ou la différence de relevé angulaire.
3. Calcul géométrique de la position à partir de données mesurées.

Chacune de ces étapes présente des difficultés qui lui sont propres. Nous allons maintenant expliciter ces difficultés. Nous verrons ensuite quelles solutions peuvent y être apportées.

4.2.1 Difficultés générales

a) Identification des amers

Le pêcheur local connaît son port d'attache et ses environs et identifie donc facilement telle église ou tel château d'eau qui se détache de la côte. Cependant, le marin qui arrive pour la première fois sur la côte anglaise après avoir traversé la Manche doit, lui, trouver la correspondance entre sa carte marine et ce qu'il peut percevoir de la côte. Cette identification est rendue particulièrement difficile par la standardisation des architectures des églises et des châteaux d'eau, ou par le brouillard…

Dans le contexte de la robotique mobile le problème d'identification des observations est aussi présent et génère deux types d'approches : soit l'expérimentateur installe dans son environ-

nement des amers identifiables simplement *[Briggs et al. 2000, . . .]*, soit il utilise des ressources logicielles et matérielles pour réaliser cette identification malgré l'ambiguïté des observations.

b) Mesures à partir des données capteur

Dans le cas général, un capteur fournit un ensemble de données brutes dans lesquelles il est nécessaire de distinguer les amers. Cette détection de points d'intérêt, ou focalisation de l'attention, est naturelle pour le navigateur mais extrêmement difficile pour un robot mobile, comme en témoignent de nombreuses publications *[Lamon et al. 2000, Thrun 1998a, Ulrich & Nourbakhsh 2000, . . .]*.

Pour l'expérimentateur roboticien, deux approches sont possibles : soit il choisit un capteur et des amers faciles à distinguer (quitte éventuellement à ajouter des amers artificiels dans son environnement), soit il consacre une partie importante de ses ressources logicielles à l'extraction de points d'intérêt naturels. On peut mentionner en particulier le coût calculatoire très important des méthodes issues de la vision par ordinateur.

Pour notre plate-forme expérimentale, nous avons choisi d'utiliser des amers artificiels, faciles à détecter mais non identifiables par leur seul retour perceptif. Ce choix est justifié par deux éléments : d'une part, rendre tous les amers distincts est en général assez difficile techniquement ; d'autre part, l'identification directe et robuste des amers à partir des données perceptives est un problème complexe de traitement du signal que nous ne voulions pas aborder dans le cadre de ce travail.

c) Calcul géométrique de la position

Dans un système de localisation par triangulation, le calcul de position résulte généralement de l'intersection de primitives géométriques (droites ou arcs). Revenons à l'exemple du marin qui fait "le point" : dès qu'il a relevé le site de deux amers, il peut tracer deux droites sur sa carte et accepter leur intersection comme estimation de sa position. Cependant, par mesure de sécurité, le marin relève généralement un troisième amer pour vérifier que la droite correspondante passe bien par l'intersection des deux premières.

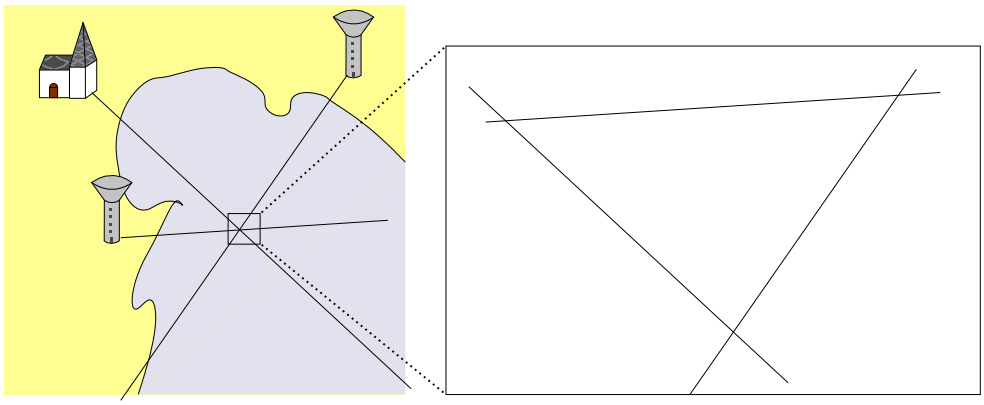


FIG. 4.2: Intersection de trois droites sur une carte marine et incertitude de la localisation.

Dans le cas général, de par les incertitudes et l'imprécision des capteurs, les trois droites s'intersectent en formant un triangle (cf. fig. 4.2). Pour un système informatique qui cherche une position unique, il sera donc impossible d'utiliser directement la construction géométrique et il faudra calculer une position qui traduise au mieux les données capteurs. Cette étape est souvent réalisée au moyen d'une optimisation aux moindres carrés.

D'autre part, si la troisième droite passe *trop* loin de l'intersection des deux autres, c'est un diagnostic d'échec pour le marin et il doit recommencer le calcul complet du point. Dans le cadre des systèmes robotiques, ce type de diagnostic est rarement utilisé car il est difficile de déterminer précisément ce qui est diagnostiqué. Est-ce l'extraction ou l'identification des amers qui a échoué, est-ce la mesure des primitives qui a donné des valeurs incohérentes ? Nous verrons à la fin de ce chapitre comment estimer de façon plus fine la confiance du système en lui-même.

4.2.2 Triangulation et télémétrie laser

Dans cette section, nous avons choisi de décrire la réalisation pratique d'un système de localisation par triangulation.

a) Définition du problème de localisation

De façon informelle le problème auquel nous sommes confrontés est le suivant : estimer aussi précisément que possible la position du véhicule à partir d'un ensemble d'observations sur un ensemble d'amers de référence.

Définition 12 – Localisation *A partir des variables de notre contexte applicatif, le problème de localisation se définit comme la construction d'une fonction "Localisation" qui associe une position x, y, θ à une carte des amers L et un ensemble d'observations Z.*

$$\textit{Localisation} : (L, Z) \longrightarrow (x, y, \theta)$$

b) L'identification des observations

La définition de la localisation présentée ci-dessus masque le problème difficile de l'identification des observations, aussi appelé problème de mise en correspondance. Ce problème est dû à trois phénomènes : d'abord, tous nos amers sont identiques et génèrent donc tous la même perception ; ensuite, il peut exister dans l'environnement des objets générant une perception identique à un amer ; enfin il est possible qu'un amer décrit dans la carte L ne soit pas visible, par exemple à cause d'une défaillance du capteur ou d'un problème d'occultation.

Intuitivement, le problème de l'identification des observations demande de déterminer pour chaque observation, d'une part s'il s'agit de l'observation d'un amer de L et d'autre part quel est l'amer qui a été observé.

La figure 4.3 illustre ce problème : la partie gauche représente la carte des amers connus L dans un repère absolu et la partie droite représente une observation particulière dans le repère capteur. Parmi ces observations, certaines (1, 2 et 3) correspondent à des amers connus (A, B et C) et une ne correspond à rien (4). De plus, seuls trois amers de L apparaissent dans les observations.

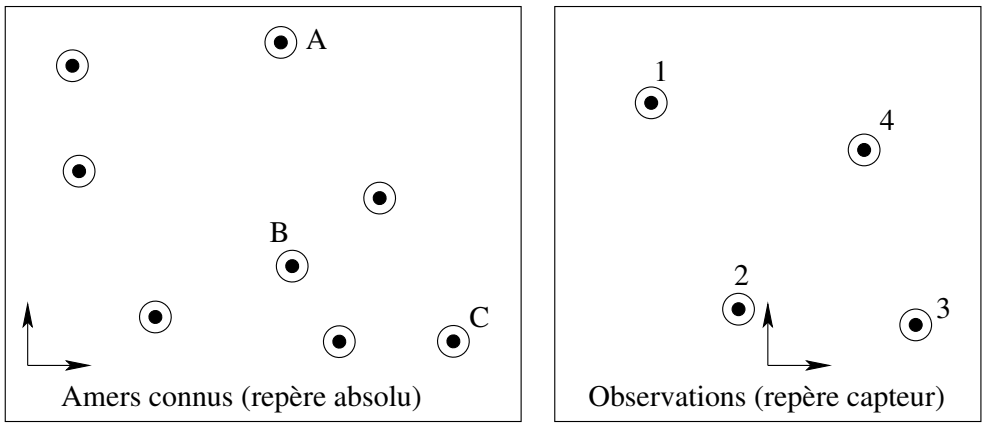


FIG. 4.3: Difficultés de la mise en correspondance.

Définition 13 – Mise en correspondance *Formellement, la mise en correspondance est la construction d'une fonction $\mathcal{I}$ de $[1..n]$ dans $[1..p] \cup \{\emptyset\}$ telle que, pour $j \in [1..n]$,*

- *$\mathcal{I}(j) = i$ si et seulement si z_j est une observation de l_i ,*
- *$\mathcal{I}(j) = \emptyset$ si et seulement si z_j ne peut être identifiée.*

La fonction $\mathcal{I}$ sera nommée fonction identificatrice.

Dans l'exemple de la figure 4.3 la fonction identificatrice est définie par les valeurs suivantes : $\mathcal{I}(1) = A,\ \mathcal{I}(2) = B,\ \mathcal{I}(3) = C$ et $\mathcal{I}(4) = \emptyset$.

c) Localisation à partir d'observation identifiée

Une fois que l'identification des observations a été faite, il est assez simple d'en déduire la position du robot. Il suffit de calculer la position qui correspond le mieux aux observations identifiées.

Idéalement, les formules de changement de repère entre le repère absolu et le repère capteur nous donnent l'expression suivante, pour une observation identifiée :

$$l_{\mathcal{I}(j)} = \begin{pmatrix} x \\ y \end{pmatrix} + R_\theta . z_i$$

où R_θ représente la rotation d'angle θ.

En pratique, diverses imprécisions rendent l'égalité précédente inexacte, et déterminer la position courante correspond à trouver un triplé (x, y, θ) qui minimise :

$$F(x, y, \theta) = \sum_{j, \mathcal{I}(j) \neq \emptyset} \| R_\theta . z_j + {}^t[x, y] - l_{\mathcal{I}(j)} \|^2 \tag{4.1}$$

avec $z_j = (x'_j, y'_j) = (\rho_j \cos(\alpha_j), \rho_j \sin(\alpha_j)$.

Heureusement, cette minimisation aux moindres carrés possède une solution analytique. La position résultante peut donc être calculée par la formule ci-dessous, en un temps qui dépend linéairement du nombre d'observations identifiées (on note $(x_j, y_j) = l_{\mathcal{I}(j)}$).

$$\begin{aligned} \theta &= \arctan\left(\frac{S_{x'y} - S_{y'x}}{S_{x'x} - S_{y'y}}\right) \\ x &= \bar{x} - (\bar{x}' \cos(\theta) - \bar{y}' \sin(\theta)) \\ y &= \bar{y} - (\bar{x}' \sin(\theta) + \bar{y}' \cos(\theta)) \end{aligned} \tag{4.2}$$

$$\forall a, \ \ \bar{a} = \tfrac{1}{n} \textstyle\sum_i a_i \qquad \forall (a, b), \ \ S_{ab} = \textstyle\sum_i (a_i - \bar{a})(b_i - \bar{b})$$

En général, cette estimation de position est utilisée dans le cadre d'une estimation temporelle de la position. On utilise donc un filtre de Kalman *[Kalman 1960, Welch & Bishop 97]* pour fusionner l'estimation calculée à partir des observations avec celle prédite à partir de l'estimation précédente et des caractéristiques du mouvement.

De plus, dans le cadre d'une estimation temporelle, une prédiction de la position est, en général, disponible au moment de la mise en correspondance. Cette connaissance peut être utilisée pour faciliter la mise en correspondance, ne serait-ce qu'en éliminant les amers non observables de par les caractéristiques du capteur.

4.2.3 Mise en correspondance par identification d'invariants

Comme on vient de le voir, dans notre contexte applicatif, la principale difficulté de la localisation par triangulation est l'identification des observations. Pour réaliser cette mise en correspondance, nous allons maintenant présenter deux méthodes issues de la littérature : les techniques d'exploration d'arbres d'interprétations par programmation dynamique de type *JCDA[Neira & Tardos 2001]* et les graphes de correspondances*[Bailey et al. 2000]*.

Nous verrons que ces méthodes sont toutes deux fondées sur la notion de primitive invariante et nous montrerons comment, par un choix judicieux de primitive, nous avons pu réduire la complexité de la méthode des graphes de correspondances.

a) Primitives invariantes

Définition 14 – Propriété invariante *Étant donné un ensemble S, un sous-ensemble E et un groupe G de transformations de S dans S, une propriété $P(E)$ de E sera dite invariante pour G si et seulement si,*

$$\forall g \in G, P(g(E)) = P(E) \tag{4.3}$$

Par exemple, avec un capteur tel que notre télémètre laser, la transformation qui permet de prévoir une observation dans le repère capteur est la composition d'une rotation et d'une translation. Pour ce groupe de transformations, la distance entre deux amers ou l'angle formé par trois amers sont des propriétés invariantes.

Définition 15 – Primitive invariante *L'évaluation d'une propriété invariante pour un sous-ensemble E sera nommée* primitive invariante associée à E.

Intérêt des propriétés invariantes Les propriétés invariantes sont utiles lorsque l'on essaye de mettre en correspondance des données observées avec des données de référence. De par leur définition, leur évaluation ne dépend pas de la manière dont l'observation a été faite, et en particulier elle ne dépend pas de la position du robot.

De cette façon, on peut montrer que l'incertitude associée aux primitives invariantes n'est pas corrélée avec celle de la position du robot. On peut ainsi réussir la mise en correspondance, même si l'estimation actuelle de la position du robot est loin de la réalité.

Démonstration 1 *La corrélation entre la position du robot X et une primitive invariante P se mesure en calculant la covariance $cov(X, P) = E[X\ P] - E[X]E[P]$.*

Or comme P ne dépend pas de X par définition, $E[X\ P] = E[X]E[P]$.

On a donc $cov(X, P) = 0$.

Reprenons l'exemple de notre contexte applicatif : puisque nous savons que ni les distances inter amers, ni les angles ne sont modifiés par l'observation, ces grandeurs pourront être utilisées dans la phase d'identification des amers observés.

b) Graphes de correspondances *[Bailey et al. 2000]*

Définition 16 – Graphe de correspondances *Un graphe de correspondances est un graphe construit pour mettre en correspondance un ensemble A avec un ensemble B. Dans un tel graphe :*

- *les noeuds représentent les correspondances possibles sous la forme d'éléments du produit cartésien $A \times B$,*
- *les arêtes représentent une relation de compatibilité entre ces correspondances.*

Replaçons-nous dans le cas d'un ensemble d'observations $Z = \{z_j = (\rho_j, \alpha_j), j = 1 \dots n\}$ à associer avec un ensemble de référence $L = \{l_i = (x_i, y_i), i = 1 \dots p\}$. Un noeud (l_i, z_j) représente l'hypothèse "z_j est une observation de l_i". Il existe une arête entre (l_i, z_j) et (l_m, z_n) si et seulement si la mise en correspondance de z_j avec l_i est cohérente avec celle de z_n avec l_m.

Comment évaluer cette cohérence ? C'est le point où l'utilisation de propriétés invariantes est indispensable : si P est une propriété invariante binaire, la cohérence sera vérifiée par la conservation de la propriété invariante.

Formellement, (l_i, z_j) est cohérent avec (l_m, z_n) si et seulement si $P(l_i, l_m) \approx P(z_j, z_n)$. Ainsi, une propriété d'arité 2 permet de définir une arête dans le graphe. De la même manière, une propriété d'arité supérieure définit une clique.

Définition 17 – Clique *Étant donné un graphe G, une clique sur G est un sous-ensemble S de noeud de G tel qu'il existe une arête de G entre tout couple de noeuds de S.*

Finalement la mise en correspondance se fait selon les étapes suivantes :

1. A partir de L, on construit une base donnée de primitives invariantes. Par exemple, en utilisant la distance entre deux amers comme invariant, pour toute paire d'amers (l_i, l_j), on stocke la distance $d(l_i, l_j)$.
2. Ensuite, pour toute propriété observée sur Z, on recherche dans L les sous-ensembles d'amers dont la primitive invariante correspond. On ajoute ensuite les arêtes correspondantes dans le graphe.
3. Enfin, on recherche dans le graphe une clique de taille maximum. Comme c'est une clique, ces noeuds forment un ensemble de correspondances qui sont toutes compatibles les unes avec les autres. De plus, puisqu'il s'agit d'une clique maximum, il n'existe pas d'ensemble de correspondances plus grand et aussi cohérent.

   Pour toute association (l_i, z_j) présente dans la clique maximum, l'image de la fonction indicatrice est donnée par $\mathcal{I}(j) = i$. Toute observation n'apparaissant pas dans la clique maximum est considérée non identifiable.

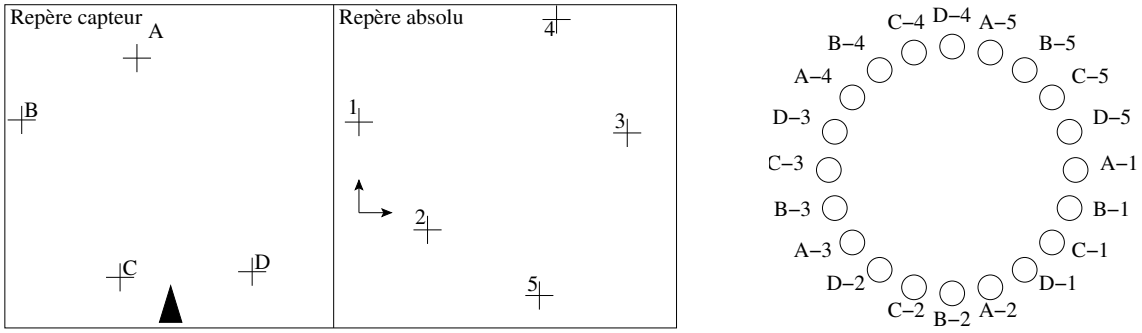


FIG. 4.4: Graphe de correspondances, situation initiale.

*1.***Processus de construction du graphe** Ce processus est illustré par les figures 4.4 à 4.11. La figure 4.4 montre l'observation Z, l'ensemble d'amers de référence L et l'état initial du graphe de correspondances : autant de noeuds que de couples (z_i, l_j) mais aucune arête. Ensuite (fig. 4.5), on considère un premier segment $L1 = (A, B)$. Ce segment peut être identifié à trois segments de référence : $(1, 2)$, $(2, 5)$ et $(3, 4)$. Comme les segments ne sont caractérisés que par leur longueur, ils ne sont pas orientés et $L1$ peut aussi être identifié à $(2, 1)$, $(5, 2)$ et $(4, 3)$. On ajoute donc les six

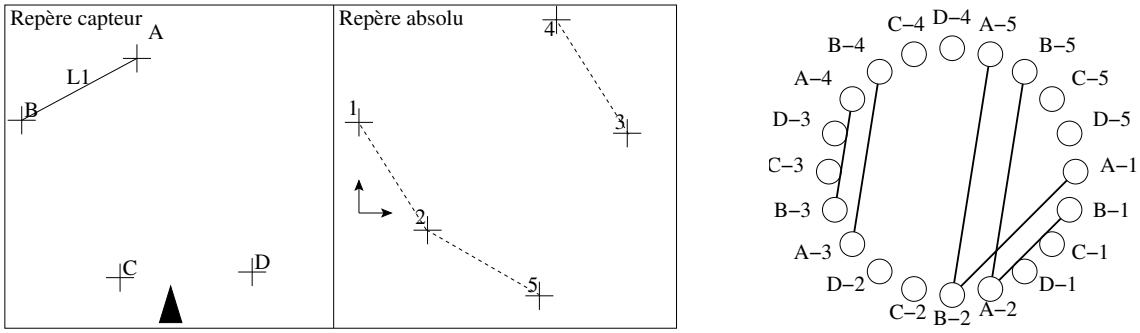


FIG. 4.5: Graphe de correspondances, identification du segment L1.

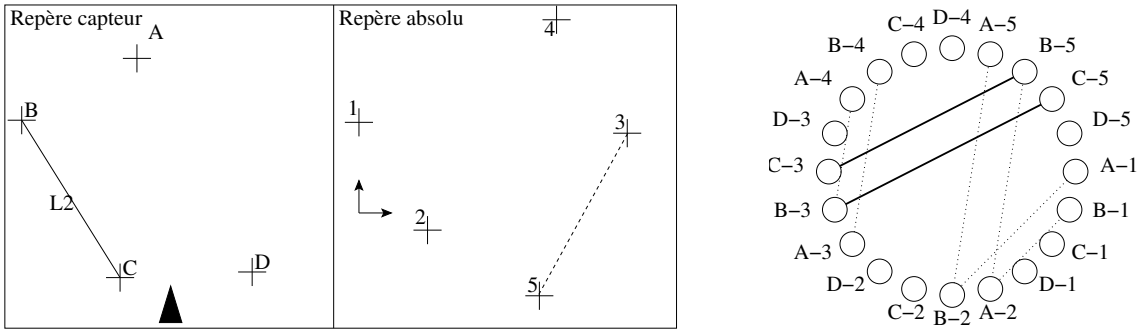


FIG. 4.6: Graphe de correspondances, identification du segment L2.

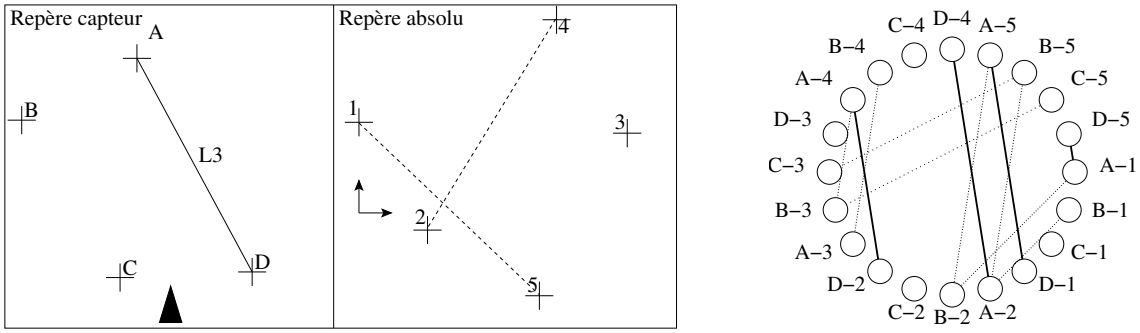


FIG. 4.7: Graphe de correspondances, identification du segment L3.

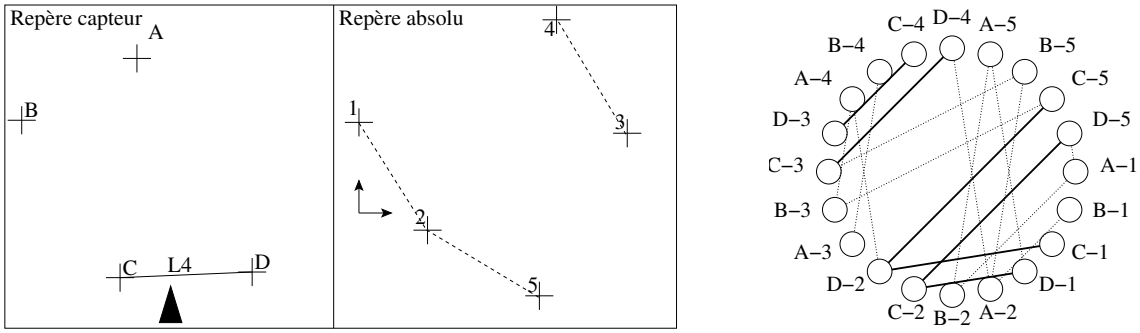


FIG. 4.8: Graphe de correspondances, identification du segment L4.

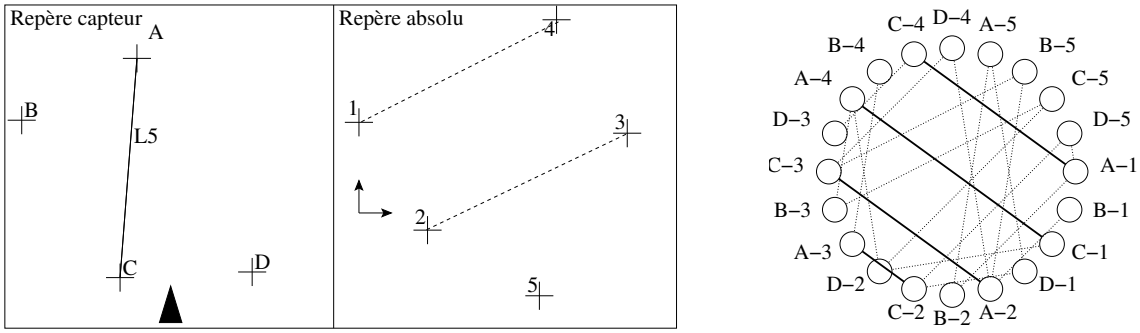


FIG. 4.9: Graphe de correspondances, identification du segment L5.

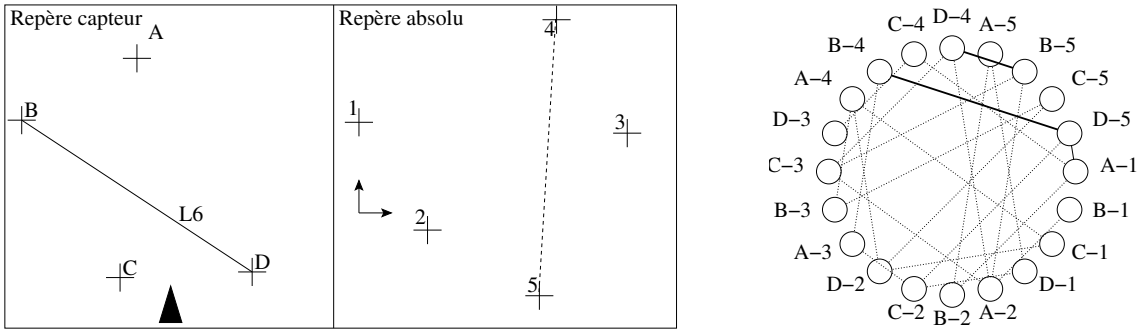


FIG. 4.10: Graphe de correspondances, identification du segment L6.

arêtes suivantes dans le graphe : $(A1, B2), (A2, B5), (A3, B4), (A2, B1), (A5, B2), (A4, B3)$. On considère ensuite les segments $L2$ à $L6$ (fig. 4.6 à 4.10). A chaque étape, les nouvelles arêtes sont représentées en traits pleins alors que celles héritées des étapes précédentes sont tracées en pointillés.

*2.***Résultat** On obtient finalement le graphe présenté dans la figure 4.11 dans lequel on recherche une clique (unique ici) de taille maximale[2]. Les noeuds présents dans cette clique ($A2$, $B5$, $C3$ et $D4$) donnent le résultat de la mise en correspondance : $\mathcal{I}(A) = 2, \mathcal{I}(B) = 5, \mathcal{I}(C) = 3$ et $\mathcal{I}(D) = 4$.

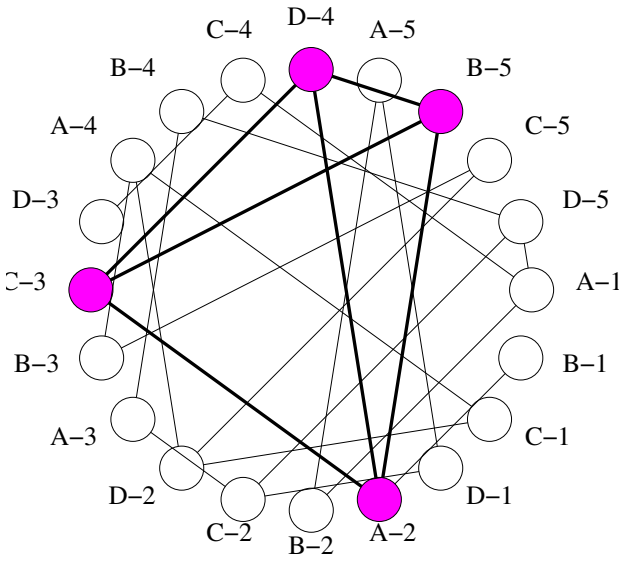


FIG. 4.11: Mise en correspondance par graphe de correspondances, en utilisant les segments liant deux amers : clique finale et résultat.

*3.***Bilan** On peut distinguer deux défauts dans le principe des graphes de correspondances : tout d'abord, le but est de trouver **une** clique maximum, et non **la** clique maximale. Dans le cas où il existe plusieurs cliques maximum de même taille, il faut donc soit considérer que la mise en correspondance a échoué, soit en choisir une arbitrairement. Nous verrons en 4.3 comment utiliser une approche probabiliste pour convertir l'incertitude sur la mise en correspondance en incertitude sur la localisation.

Ensuite, la recherche d'une clique maximum dans un graphe est, en général un problème NP-complet. Cependant, nous montrerons en section 4.2.4 qu'un choix judicieux d'invariant permet de traiter ce problème complexe en temps réel.

[2]Un ensemble de noeuds d'un graphe est appelé clique maximale s'il s'agit d'une clique (cf. définition 17) et si tout ajout de noeud à l'ensemble lui fait perdre cette propriété.

c) Arbres d'interprétation :

Un arbre d'interprétation permet de mettre en correspondance un ensemble d'observations $Z = \{z_j = (\rho_j, \alpha_j), j = 1 \dots n\}$ avec un ensemble de référence $L = \{l_i = (x_i, y_i), i = 1 \dots p\}$. C'est un arbre de hauteur n dont chaque étage représente les différentes valeurs possibles pour $\mathcal{I}(z_j)$ (cf. fig. 4.12). Ainsi, en considérant un chemin de la racine à une feuille, on considère une interprétation possible de Z. Dans l'optique d'une mise en correspondance, il est donc nécessaire de rechercher dans cet ensemble d'interprétation celle qui semble la meilleure. Comme pour les graphes de correspondances, nous considérons qu'une bonne interprétation doit contenir des identifications toutes cohérentes les unes avec les autres, c'est à dire conservant les propriétés invariantes. La meilleure interprétation est donc celle qui identifie le plus grand nombre d'observations, tout en ne proposant que des identifications cohérentes entre elles.

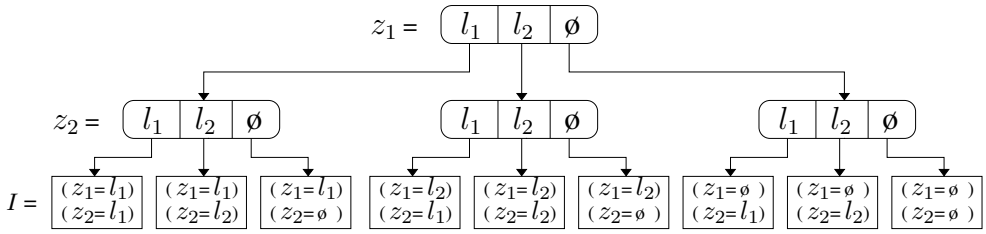


FIG. 4.12: Arbre d'interprétation pour $Z = \{z_1, z_2\}$ et $L = \{l_1, l_2\}$.

Pour réaliser cette exploration des interprétations efficacement, certains auteurs *[Neira & Tardos 2001]* proposent d'utiliser des techniques de programmation dynamique. C'est le principe d'algorithmes tels que la mise en correspondance par compatibilité conjointe *JCDA*[3] décrit dans l'algorithme 1. De même que les graphes de correspondances, ces algorithmes ne gèrent pas le cas où plusieurs interprétations sont maximales. Cependant ils ont l'avantage de fournir directement une interprétation contrairement aux graphes de correspondances qui laissent en suspend la recherche de clique.

d) Similarité entre les deux approches

Finalement ces deux approches sont très similaires. Chacune essaye d'énumérer l'ensemble des fonctions $\mathcal{I}$ de $[1 \dots n]$ dans $[1 \dots p] \cup \{\emptyset\}$ en choisissant celles qui associent le plus d'observations de manière complètement cohérente. En fait, la seule différence est la manière de représenter ces interprétations : les graphes de correspondances les représentent comme des cliques dans un graphe à $n \times p$ noeuds alors que le *JCDA* les représente grâce à la pile d'une fonction récursive.

Terminons par une limitation des deux approches présentées ci-dessus. Lorsque l'environnement ou les perceptions sont suffisamment ambiguës, il peut arriver que la clique maximale ne soit pas unique. De même, il peut arriver qu'il existe deux branches également valides dans l'arbre d'interprétation. Néanmoins, ces algorithmes doivent faire un choix et indiquer une des solutions comme LA solution. Les autres hypothèses sont alors oubliées. Une façon astucieuse

[3]Joint Compatibility Data Association.

Algorithme 1: "Joint Compatibility Data Association (JCDA)", d'après *[Neira & Tardos 2001]*.

Données : $Best = \emptyset$
Identifications(H) représente le nombre d'observations identifiées dans H
Résultat : $Best$
Fonction $JCDA(H, j)$
début
  si *il reste des observations à identifier* **alors**
    pour chaque *amer* L_i **faire**
      si *considérer* Z_j *comme une observation de* L_i *vérifie les contraintes de cohérence* **alors**
        $JCDA([H, i], j + 1)$
    fin
    si *Identifications(H)* $+p - j \geq$ *Identifications(Best)* **alors**
      $JCDA([H, \emptyset], j + 1)$
  sinon
    si *Identifications(H)* $>$ *Identifications(Best)* **alors** $Best \leftarrow \{H\}$
    si *Identifications(H)* $=$ *Identifications(Best)* **alors** $Best \leftarrow Best \cup \{H\}$
  fin
fin

de résoudre ce problème est de ne garder que les identifications qui appartiennent à toutes les hypothèses. Cependant, il faut alors que l'exploration de l'arbre d'interprétation ou la recherche de clique maximale trouve toutes les identifications possibles, ce qui peut se révéler relativement coûteux.

Dans la section 4.3 nous montrerons comment les probabilités permettent de mieux représenter ces cas d'ambiguïté. Remarquons toutefois que les situations où ni l'estimation de position courante ni la cohérence globale de la mise en correspondance ne sont suffisantes pour lever l'ambiguïté sur l'identification sont très rares en pratique.

4.2.4 Le triangle : une primitive invariante optimale

a) Critères désirables pour une primitive invariante

Comme nous l'avons vu, les invariants géométriques tels que la distance entre deux amers, ou les angles formés par trois amers sont des propriétés fondamentales pour une procédure de mise en correspondance. Cependant, pour obtenir robustesse et efficacité, l'invariant que nous allons choisir doit avoir des propriétés supplémentaires :

Être discriminant : La principale source de difficulté de la méthode des graphes de correspondances est la recherche d'une clique maximale. La complexité de cette recherche dépend de la densité du graphe. Il est donc souhaitable de choisir un critère de compatibilité aussi discriminant que possible. Avec un critère idéal, les seules arêtes du graphe de correspondance seraient celles de la clique maximale. Dans ce cas, l'extraction de la clique maximale est quasi immédiate.

Être facile à comparer : Lorsqu'on utilise des invariants pour la mise en correspondance, le principe sous-jacent est toujours de stocker les primitives invariantes dans quelque arbre de recherche et d'y rechercher les primitives observées. Pour cette raison, la comparaison entre deux primitives doit être rapide et simple (l'idéal étant de ne coûter qu'une opération processeur). Lorsqu'on compare des angles, il faut d'abord les normaliser, puis normaliser la différence. Nous éviterons donc les angles.

Être peu sensible aux fausses observations : Supposons qu'une première observation z_1 ait été identifiée à un amer l_i et qu'on sache qu'un second amer l_j est visible. Supposons de plus que nous voulons utiliser la distance entre deux amers comme critère invariant. Dans ce cas, toute observation située sur un cercle de centre l_i et de rayon $d(l_i, l_j)$ pourra être considérée comme une observation cohérente de l_j. Ce résultat est problématique car lorsqu'on installe des amers artificiels dans un environnement, il est très difficile, voire impossible, de garantir que les distances entre deux amers seront suffisamment différentes les unes des autres. Pour éviter ce type d'association incorrecte, il nous faut choisir une propriété invariante aussi robuste que possible. En pratique, la primitive associée à un ensemble d'amers ne doit pas être associable avec celle d'un ensemble d'observations dont l'une résulte d'une erreur de détection.

b) Choix d'un critère de comparaison optimal

Avec les critères ci-dessus, ni les distances, ni les angles ne semblent satisfaisant. C'est pourquoi nous avons choisi de mettre en correspondance les triangles formés par trois amers.

Définition 18 – Correspondance des triangles *Nous dirons que deux triangles correspondent si leurs aires sont proches et s'il existe une rotation qui les rend superposables.*

*1.***Analyse** L'aire est un critère très intéressant puisque, d'une part, c'est une propriété invariante par translation/rotation, et d'autre part, elle reflète à la fois les longueurs des arêtes du triangle et les angles de ses sommets. Cependant, comme deux triangles non superposables peuvent avoir la même aire, nous devons ajouter la recherche de rotations pour les différencier (notons que seules trois rotations doivent être testées).

A première vue, ce critère n'est pas plus simple que les comparaisons d'angle. Cependant, la comparaison des aires étant déjà très discriminante, on peut considérer une mise en correspondance en deux étapes : on compare d'abord les aires, puis, on utilise la recherche de rotations pour sélectionner les bonnes correspondances parmi celles qui ont passé la première étape.

*2.***Utiliser plus de trois amers ?** Comme un triangle est une structure rigide du plan, l'utilisation de triangles fournit des contraintes très fortes sur la mise en correspondance et satisfait donc pleinement notre besoin de robustesse.

Toutefois, on pourrait se demander s'il est possible d'améliorer encore la robustesse en considérant un invariant d'arité supérieure. Puisque le triangle est le simplexe de dimension deux, tout invariant d'arité supérieure pourra s'exprimer en fonction d'une triangulation de ses arguments. On peut donc considérer que le gain en robustesse ne compensera pas l'augmentation de complexité due à une arité supérieure.

*3.***Limitations** Finalement, il faut noter l'inconvénient lié à l'utilisation d'un invariant ternaire : il n'est pas possible d'identifier moins de trois observations. Pour identifier deux observations, on utilisera donc uniquement la distance et on ne pourra utiliser cette méthode pour identifier une seule observation.

c) Résultats

Les figures 4.13 à 4.18 illustrent l'intérêt de l'utilisation de triangles dans le processus de mise en correspondance. Comme pour les segments, à partir de la situation initiale (fig. 4.13), on cherche à identifier chaque triangle observé à un triangle de la carte de référence (fig. 4.14 à 4.17).

A la différence du graphe obtenu à partir des segments (fig. 4.11), le graphe final (fig. 4.18) construit ici est très simple et la recherche d'une clique maximale y est extrêmement simplifiée.

Cette simplification, conséquence des très fortes contraintes introduites par l'identification de triangles, est ce qui rentabilise l'augmentation de complexité due à l'utilisation d'un invariant ternaire.

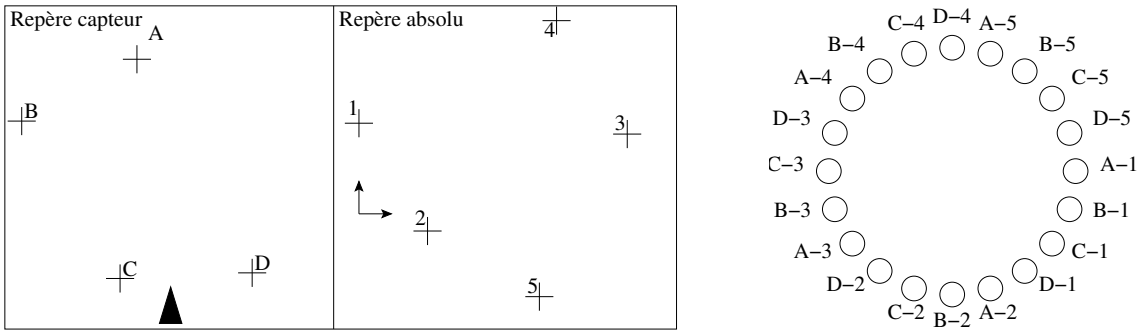


FIG. 4.13: Graphe de correspondances, situation initiale.

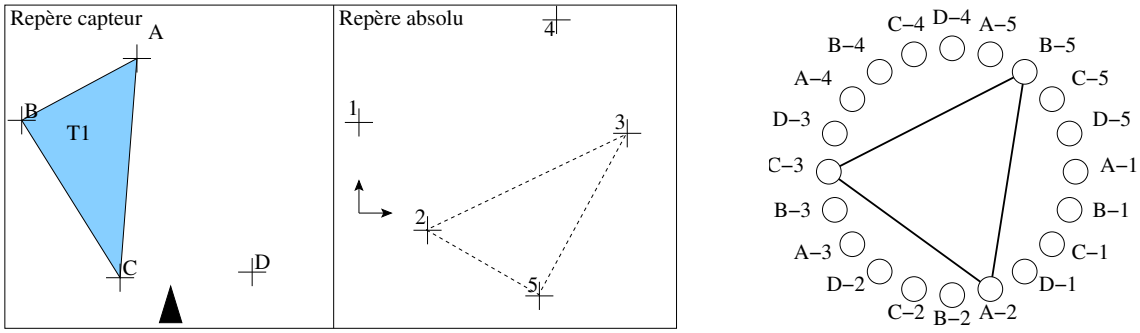


FIG. 4.14: Graphe de correspondances, identification du triangle T1.

4.2.5 Conclusion

Nous pouvons résumer les thèmes abordés dans cette section en trois points :

- Tout d'abord, étant donné un ensemble d'observations identifiées, la construction d'une estimation de la localisation est un problème résolu.
- Ensuite, la réelle difficulté de la localisation par triangulation est l'identification des observations. De plus, sur un système robotique cette identification doit être robuste, fiable et rapide car la sécurité globale du système dépend, en général, d'une bonne localisation.
- Enfin, pour optimiser le processus de mise en correspondance par graphes de correspondances, nous proposons de réaliser l'identification de triplés d'amers. L'utilisation de ces primitives invariantes permet d'obtenir des graphes de correspondances moins denses, et ainsi de construire plus rapidement la fonction indicatrice $\mathcal{I}$.

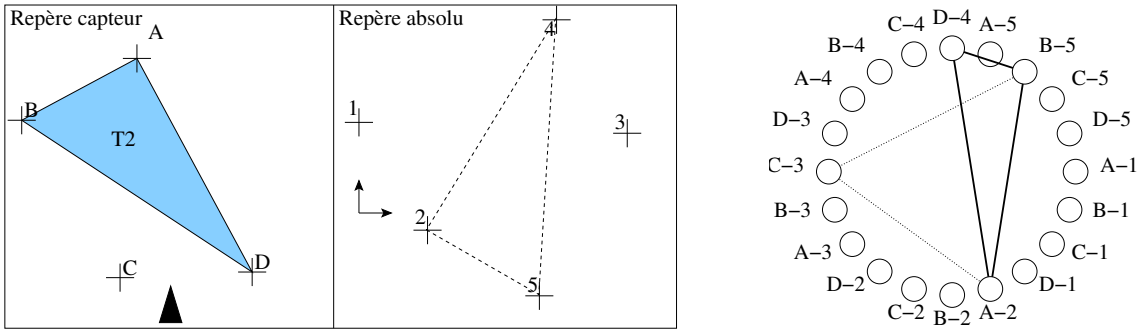


FIG. 4.15: Graphe de correspondances, identification du triangle T2.

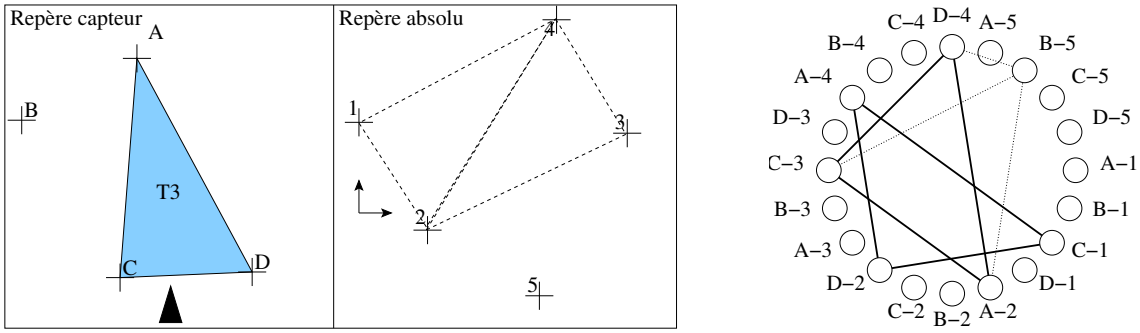


FIG. 4.16: Graphe de correspondances, identification du triangle T3.

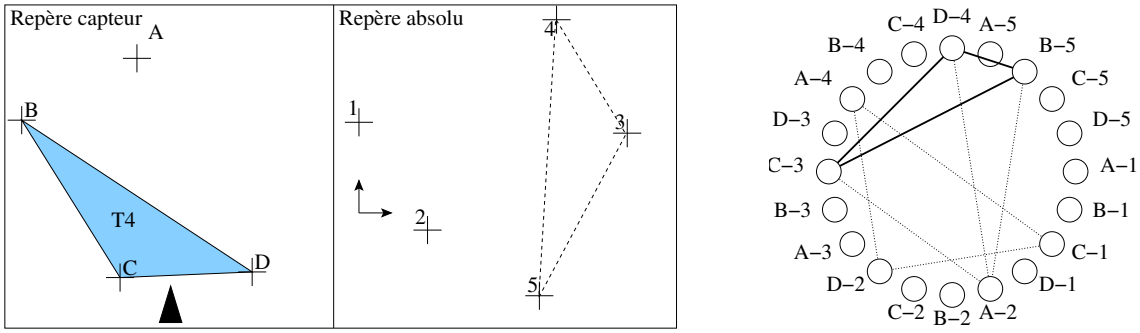


FIG. 4.17: Graphe de correspondances, identification du triangle T4.

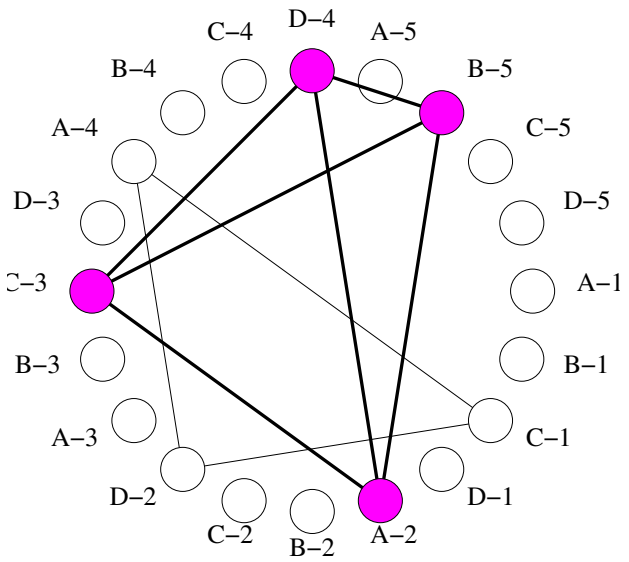


FIG. 4.18: Mise en correspondance par graphe de correspondances, en utilisant les triangles formés par trois amers : clique finale et résultat.

La nécessité de connaître la carte des amers *a priori* peut être vu comme un aspect limitant des mécanismes de localisation présentés dans cette section. Dans le chapitre 5, nous montrerons comment intégrer ces algorithmes de mises en correspondances dans un processus de localisation et construction de carte simultanées (SLAM).

4.3 Localisation sans mise en correspondance

Le point clef de la localisation par triangulation est la mise en correspondance. Telle qu'elle a été décrite dans la section précédente, la robustesse de cette procédure provient de sa capacité à construire une identification des observations complètement auto-cohérente : toutes les associations entre observations et références sont cohérentes les unes avec les autres.

Cependant, il existe un type de situation où le fonctionnement de cet algorithme n'est pas satisfaisant. Lorsque l'environnement contient des ambiguïtés perceptives, et lorsqu'il existe plusieurs identifications possibles des observations, toutes aussi cohérentes les unes que les autres, trois résultats sont possibles (selon l'implantation de l'algorithme) : l'échec de la mise en correspondance, l'énumération de toutes les identifications possibles ou le choix arbitraire d'une des identifications. **Aucune des ces réponses n'est satisfaisante car elles ne fournissent aucun moyen de transformer l'incertitude sur la mise en correspondance en incertitude sur la localisation.**

Dans cette section, nous allons montrer qu'en utilisant la méthodologie de la programmation bayésienne, il est possible de construire un processus de localisation par rapport à une carte d'amers qui réalise la mise en correspondance de façon implicite, transformant ainsi une identification ambiguë des amers en localisation ambiguë. Cette ambiguïté est plus intéressante car plus facile à manipuler. En particulier, elle peut, simplement, être réduite par l'accumulation

d'observations dans le cadre d'un filtre bayésien.

4.3.1 Localisation bayésienne par fusion de données perceptives

Pour pouvoir faire de la localisation bayésienne, il est nécessaire de définir un modèle probabiliste mettant en relation la configuration C du robot et son observation Z. On pourra ensuite utiliser une approche de fusion de données bayésienne pour construire une distribution de probabilité sur la configuration du robot. Pour illustrer la construction d'un tel modèle capteur, nous utiliserons le cas particulier suivant : un robot ponctuel décrit par sa position x, y dans le plan, un ensemble d'amers ponctuels fixes dans le plan et un capteur qui mesure la distance entre un amer et le robot. Le modèle capteur sera alors utilisé pour évaluer $P(C \mid Z)$. Pour faciliter la comparaison, une même observation sera utilisée dans tous les exemples.

a) Modèle capteur de base

Lorsqu'on définit un modèle de capteur, la solution la plus simple (et la plus souvent utilisée dans la littérature) est de considérer que l'observation ne dépend que de la configuration, et de définir ainsi une distribution conjointe :

$$P(C\ Z) = P(C)P(Z \mid C) \tag{4.4}$$

où $P(C)$ décrit notre *a priori* sur la configuration, en général uniforme. $P(Z \mid C)$ est, dans le contexte de la programmation bayésienne, une expression d'un modèle du processus d'observation et des incertitudes associées. Exprimé de cette façon, ce modèle est prédictif : connaissant la configuration du robot C, il prédit, avec une certaine confiance, l'observation qui devrait être réalisé en C. Plusieurs types de spécification sont possibles :

- Une loi de Dirac : il s'agit d'un modèle parfait, connaissant une configuration C, il n'y a qu'une seule observation possible. Tout autre observation n'est même pas envisageable. Avec un capteur réel, il est très rare, voire impossible, d'utiliser un tel modèle.
- Une loi uniforme : connaissant C, toutes les observations sont équiprobables. Cela signifie que C n'apporte aucune information sur la prédiction de Z. Même si un tel modèle peut paraître inutile, nous verrons, dans la suite de ce chapitre, qu'il existe des situations où il est indispensable d'exprimer l'ignorance avec une telle distribution. Nous verrons aussi, dans quelques paragraphes, que l'expression de l'ignorance n'est pas toujours une chose facile.
- Une loi gaussienne ou une loi uniforme sur un segment : connaissant C, on peut prédire l'observation Z avec une certaine précision. Cette précision est exprimée par l'écart type de la distribution gaussienne ou par la largeur du segment de support de la loi uniforme.
- Dans l'optique de l'approche subjectiviste des probabilités, on peut combiner à l'infini les formes de distributions pour exprimer de façon plus précise nos connaissances sur les phénomènes que nous modélisons. Cependant, les trois lois précédentes sont des échantillons assez représentatifs du type de connaissance que l'on peut vouloir exprimer avec un modèle capteur.

L'équation 4.4 masque un ensemble de variables qui représentent la modélisation de l'environnement. Dans le cas d'une localisation à partir de l'observation d'un amer, nous appellerons L la variable probabiliste décrivant la position de cet amer. La distribution conjointe devient :

$$P(C\ Z\ L) = P(C)P(L)P(Z \mid C\ L) \tag{4.5}$$

où $P(L)$ donne l'*a priori* sur la position des amers, en général uniforme, voire non spécifié. La figure 4.19 représente la distribution de probabilité sur les configurations, obtenue à partir d'une mesure de distance sur un amer. On observe un segment d'arc de cercle correspondant au lieu des configurations situées à distance constante Z de l'amer L.

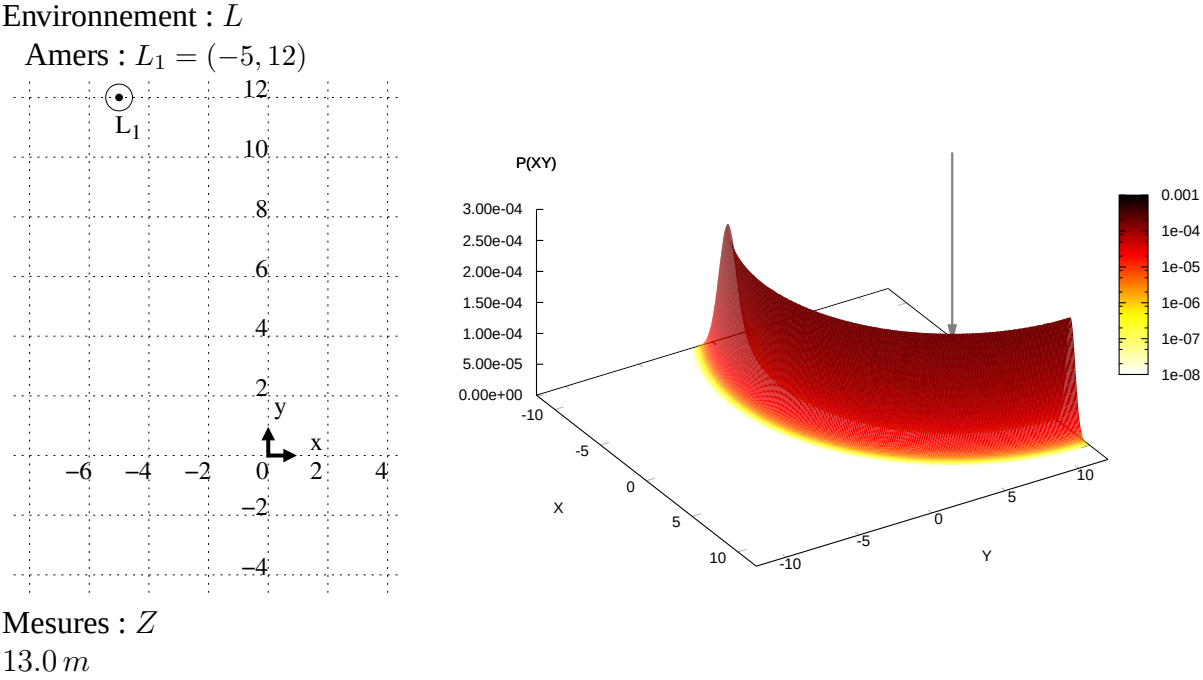


FIG. 4.19: Localisation avec un modèle capteur élémentaire (eq. 4.5) : A gauche, vue cartésienne de l'environnement. A droite, sur la distribution de probabilité $P(C \mid Z\ L)$, la flèche indique la position de l'amer L_1.

b) Modèle capteur intégrant la notion de fausse observation

Pour obtenir un modèle complet, il faut tenir compte du fait qu'avec un capteur réel, il peut arriver que Z résulte d'un phénomène non modélisé[4], et donc ne soit l'observation d'aucun amer connu. Nommons F cet évènement : F est vrai si Z ne résulte pas de l'observation d'un amer connu. Avec un seul amer dans l'environnement, on obtient :

$$P(C\ Z\ F\ L) = P(C)P(L)P(F \mid C\ L)P(Z \mid C\ F\ L) \tag{4.6}$$

[4]souvent appelé "bruit" ou "bruit de mesure" dans la littérature.

$P(F \mid C\ L)$ permet d'exprimer les chances de succès de la mesure connaissant la configuration et les amers. Toutefois, en général, cette dépendance est assez difficile à établir. On préférera alors définir $P(F)$ indépendamment et l'exprimer comme la probabilité d'échec du capteur indépendamment de son environnement. Considérons maintenant $P(Z \mid C\ F\ L)$.

- Si F est faux, on se trouve de nouveau dans le cas implicitement utilisé pour définir l'équation 4.5.
- Si F est vrai, il faut que cette distribution soit le reflet de l'observation attendue sachant que cette observation résulte d'un phénomène non modélisé. On choisit donc une distribution qui traduit un minimum de connaissance *a priori*, en général uniforme.

La figure 4.20 illustre l'effet de la variable F sur la localisation. Sans cette variable (fig. 4.19), les positions du robot incohérentes avec la mesure actuelle sont déclarées très improbables, voire impossibles (en blanc sur la figure 4.19). En tenant compte de la possibilité d'échec du capteur, les positions incohérentes sont peu probables mais restent envisageables (en orange[5] sur la figure 4.20). Ceci est particulièrement intéressant dans les cas où l'on souhaite fusionner cette distribution avec une distribution prédite, par exemple dans un filtre bayésien.

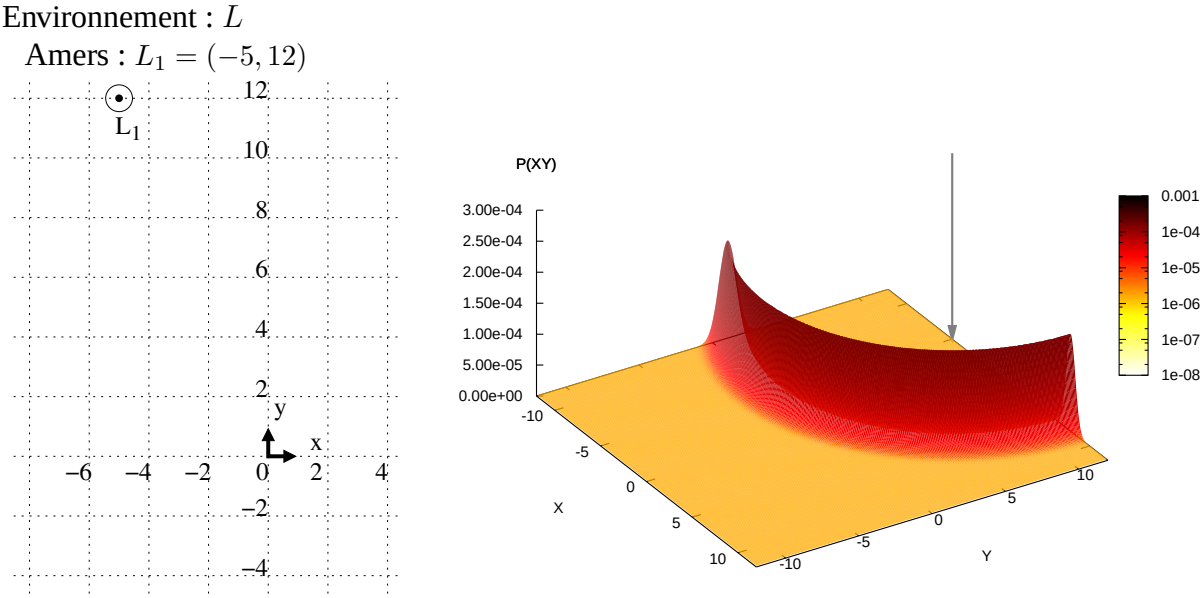


FIG. 4.20: Localisation avec un modèle de capteur élémentaire intégrant la possibilité d'échec du capteur (eq. 4.6) : A gauche, vue cartésienne de l'environnement. A droite, sur la distribution de probabilité $P(C \mid Z\ L)$, la flèche indique la position de l'amer L_1.

[5] gris clair sur la version monochrome de ce document.

c) Introduction de la mise en correspondance dans le modèle capteur

En général, il n'y a pas qu'un seul amer dans l'environnement. Nommons $\mathcal{L} = L_i, i = 1..p$ l'ensemble des variables correspondant aux positions des amers. En posant $P(\mathcal{L}) = \prod_i P(L_i)$, on obtient

$$P(C\ Z\ \mathcal{L}) = P(C)P(\mathcal{L})P(Z \mid C\ \mathcal{L}) \tag{4.7}$$

Z est l'observation d'un amer mais l'équation ci-dessus ne permet pas de savoir lequel a été observé. Autant on devine assez facilement le type de modèle prédictif utilisé dans l'équation 4.5, autant l'écriture d'un modèle d'observation non identifiée d'un ensemble d'amers n'est pas immédiate. Pour faciliter cette définition, nous introduisons donc une variable d'identification W (pour **W**hich) qui indique l'amer ayant produit l'observation Z. On obtient donc une nouvelle distribution conjointe :

$$P(C\ Z\ W\ \mathcal{L}) = P(C)P(\mathcal{L})P(W \mid C\ \mathcal{L})P(Z \mid C\ W\ \mathcal{L}) \tag{4.8}$$

$P(W \mid C\mathcal{L})$ permet de donner un *a priori* sur l'identification. Le plus naturel étant d'utiliser cette distribution pour indiquer un sous-ensemble d'amers potentiellement visibles depuis la configuration C. On peut aussi envisager des modèles plus complexes favorisant certaines identifications par rapport à d'autres, mais leur justification et leur implantation deviennent plus difficiles. La figure 4.21 montre l'ensemble des localisations possibles avec une mesure de distance et deux amers, sans tenir compte des possibilités d'échec du capteur. On y observe deux arcs de cercle correspondant aux deux hypothèses d'identification de la mesure. La probabilité d'être à l'intersection des deux cercles est plus importante car dans ce cas, on fait l'observation Z quelle que soit la valeur de W.

En tenant compte à la fois du problème d'identification de l'observation et de la probabilité d'échec du capteur, le modèle complet s'exprime par :

$$P(C\ Z\ W\ F\ \mathcal{L}) = P(C)P(\mathcal{L})P(F \mid C\ \mathcal{L})P(W \mid F\ C\ \mathcal{L})P(Z \mid C\ W\ F\ \mathcal{L}) \tag{4.9}$$

Si $F = 1$, W n'a plus de sens. La définition de $P(W \mid F\,C\,\mathcal{L})$ devient donc problématique. Afin de ne pas exprimer de connaissance dans ce cas, on choisit de rendre uniforme cette distribution pour $F = 1$. Comme on le verra par la suite, cette définition est de toute façon superflue pour le problème de la localisation. La figure 4.22 présente le résultat de l'influence conjointe de F et W sur la localisation à partir d'une observation et de deux amers. Comme pour la figure 4.21, on observe deux arcs de cercles correspondant aux positions compatibles avec un fonctionnement correct du capteur. Contrairement à la figure précédente, les positions en dehors de ces arcs, compatibles avec un dysfonctionnement du capteur, sont peu probables mais envisageables.

d) Modèle complet

Finalement, on peut aussi se placer dans le cas d'un ensemble d'observations d'un ensemble d'amers, et fusionner les informations apportées par chaque observation sur la localisation. On pose $\vec{Z} = \{Z_j\}_{j=1\cdots n}$, $\vec{F} = \{F_j\}_{j=1\cdots n}$ et $\vec{W} = \{W_j\}_{j=1\cdots n}$ et on obtient la distribution conjointe

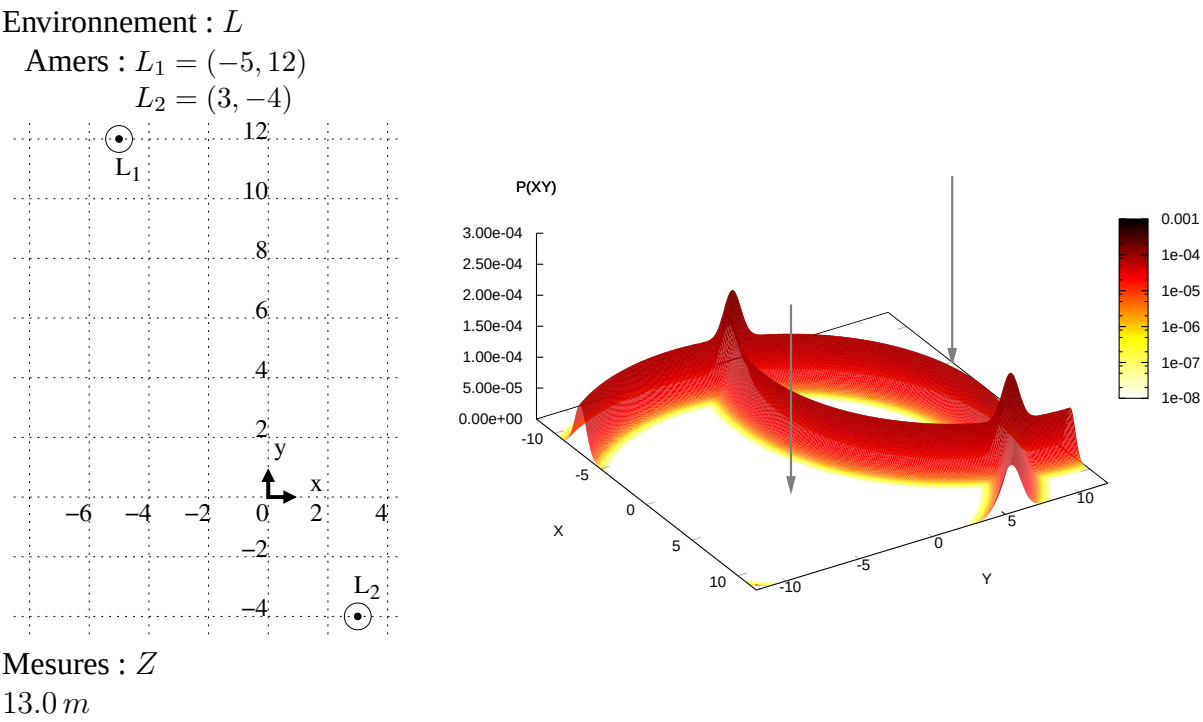


FIG. 4.21: Localisation avec deux amers (eq. 4.8) : A gauche, vue cartésienne de l'environnement. A droite, sur la distribution de probabilité $P(C \mid Z\ \mathcal{L})$, les flèches indiquent les positions des amers L_1 et L_2.

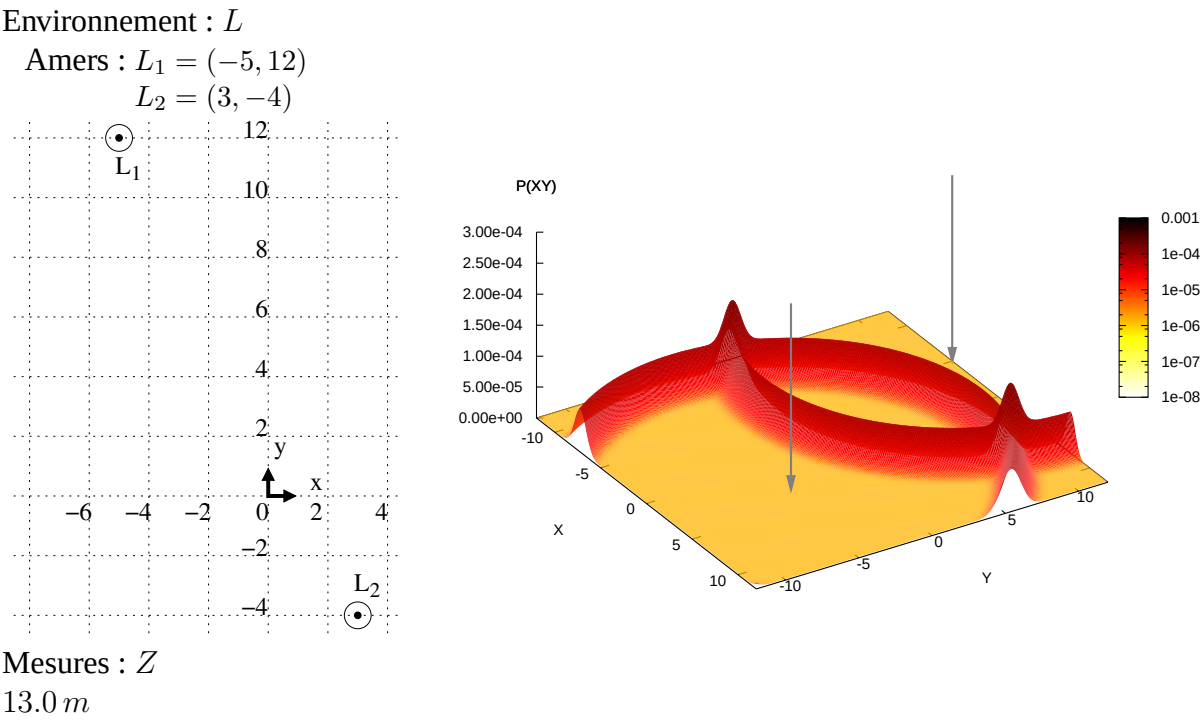


FIG. 4.22: Localisation multi-amers avec un modèle d'échec : à gauche, vue cartésienne de l'environnement. À droite, sur la distribution de probabilité $P(C \mid Z\,\mathcal{L})$, les flèches indiquent les positions des amers L_1 et L_2.

suivante :

$$P(C\ \vec{Z}\ \vec{W}\ \vec{F}\ \mathcal{L}) = P(C)P(\mathcal{L})\prod_j P(F_j \mid C\ \mathcal{L})P(W_j \mid F_j\ C\ \mathcal{L})P(Z_j \mid C\ W_j\ F_j\ \mathcal{L}) \quad (4.10)$$

Le modèle complet, exprimé sous forme de programme bayésien, est présenté dans la figure 4.23.

Programme

Description

Spécification

Variables Pertinentes

C	:	Configuration du robot
$\mathcal{L} = L_1 \ldots L_p$	:	Amers observables
$\vec{Z} = Z_1 \ldots Z_n$	:	Observation
$\vec{F} = F_1 \ldots F_n$	:	Booléen, l'observation i résulte-t-elle d'un phénomène non modélisé ?
$\vec{W} = W_1 \ldots W_n$	:	Identification de l'observation i

Décomposition

$P(C\ \vec{Z}\ \vec{W}\ \vec{F}\ \mathcal{L}) =$
$P(C)P(\mathcal{L})$
$\prod_j P(F_j \mid C\ \mathcal{L})P(W_j \mid F_j\ C\ \mathcal{L})P(Z_j \mid C\ W_j\ F_j\ \mathcal{L})$

Formes Paramétriques

cf. texte

Identification :

cf. texte

Question :

$P(C \mid \vec{Z}\ \mathcal{L}) = P(C)P(\mathcal{L})\times$
$\prod_j \left[\sum_{F_j} \left\{P(F_j \mid C\ \mathcal{L}) \sum_{W_j} P(W_j \mid F_j\ C\ \mathcal{L})P(Z_j \mid C\ W_j\ F_j\ \mathcal{L})\right\}\right]$

FIG. 4.23: Modèle de localisation par fusion, avec mise en correspondance implicite

La question qui nous intéresse au final est :

$$P(C \mid \vec{Z}\ \mathcal{L}) = P(C)P(\mathcal{L})\prod_j \sum_{F_j} P(F_j \mid C\ \mathcal{L}) \sum_{W_j} P(W_j \mid F_j\ C\ \mathcal{L})P(Z_j \mid C\ W_j\ F_j\ \mathcal{L}) \quad (4.11)$$

Cette équation permet de comprendre pourquoi il n'est pas nécessaire de définir $P(W_j \mid [F_j = 1]\ C\ \mathcal{L})$. En effet, lorsque F_j est vrai, l'observation résulte d'un phénomène non modélisé et donc l'observation Z_j est indépendante des autres variables, en particulier de W_j. L'expression

$$P([F_j = 1] \mid C\ \mathcal{L}) \sum_{W_j} P(W_j \mid F_j\ C\ \mathcal{L})P(Z_j \mid C\ W_j\ [F_j = 1]\ \mathcal{L})$$

se simplifie donc, par normalisation, de la façon suivante :

$$P([F_j = 1] \mid C\ \mathcal{L}) \sum_{W_j} P(W_j \mid F_j\ C\ \mathcal{L})P(Z_j \mid C\ W_j\ [F_j = 1]\ \mathcal{L})$$

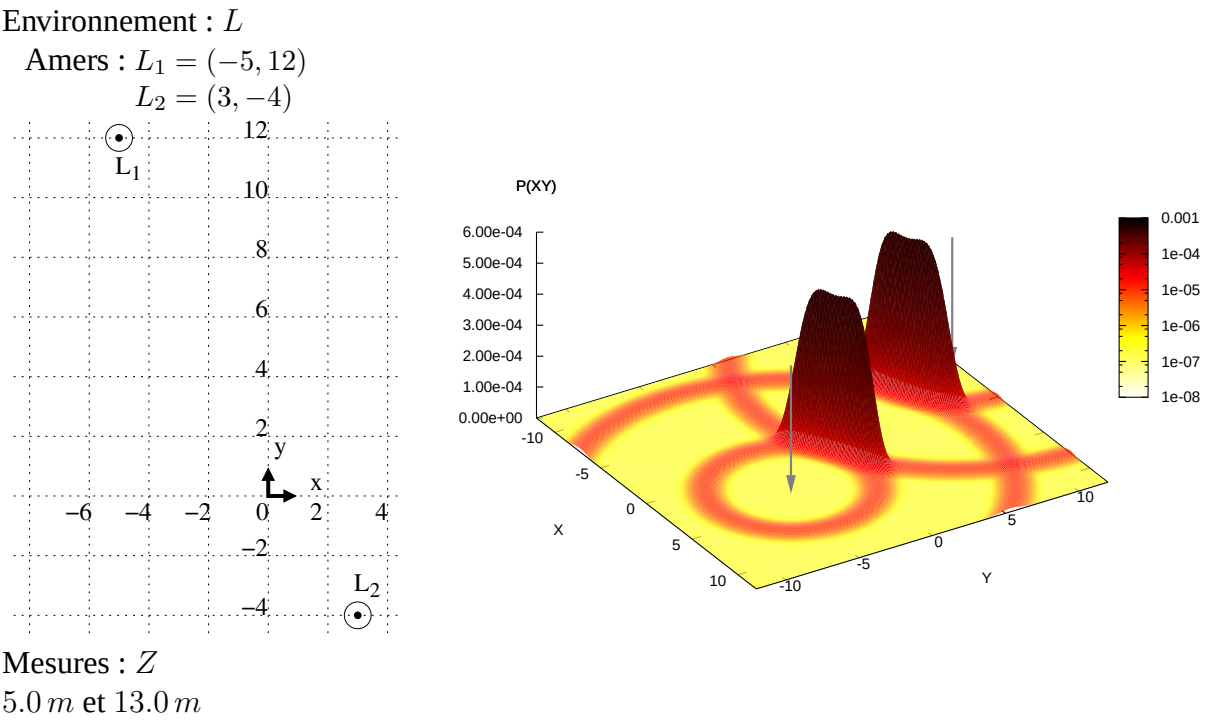


FIG. 4.24: Localisation avec un modèle complet (eq. 4.11) : A gauche, vue cartésienne de l'environnement. A droite, sur la distribution de probabilité $P(C \mid \vec{Z}\,\mathcal{L})$, les flèches indiquent les positions des amers L_1 et L_2.

$$
\begin{aligned}
&= P([F_j = 1] \mid C\ \mathcal{L}) P(Z_j \mid [F_j = 1]) \quad \times \quad \sum_{W_j} P(W_j \mid F_j\ C\ \mathcal{L}) \\
&= P([F_j = 1] \mid C\ \mathcal{L}) P(Z_j \mid [F_j = 1]) \quad \times \quad 1 \\
&= P([F_j = 1] \mid C\ \mathcal{L}) P(Z_j \mid [F_j = 1])
\end{aligned}
$$

Avec deux mesures de distance sur les deux amers de l'environnement, l'évaluation de l'équation 4.11 produit la distribution de probabilité illustrée par la figure 4.24. Sur cette figure on peut distinguer trois ensembles de positions selon la valeur des probabilités :

- Les deux pics, centrés sur $(0, 0)$ et $(-7, 9)$, correspondent aux localisations découlant des deux hypothèses de mise en correspondance équiprobables : soit Z_0 correspond à L_0 et Z_1 à L_1, soit Z_0 correspond à L_1 et Z_1 à L_1. Avec deux mesures de distances, les deux hypothèses permettent une interprétation aussi cohérente des observations.
- Les cercles correspondant à des valeurs de probabilité de l'ordre de 10^{-5} (en orange[6]) sont les localisations possibles si une des deux mesures est une fausse observation. Dans ce cas quatre possibilité équiprobables sont à considérer selon que la fausse observation est Z_1 ou Z_2 et qu'il s'agit d'une observation de L_1 ou L_2. Pour chacune de ces possibilités, comme pour la figure 4.19, on connaît uniquement la distance à un amer donc les positions possibles sont sur un cercle centré sur l'amer observé.
- Le reste de l'espace (en jaune[7]) correspond à l'ensemble des positions possibles si les deux mesures sont des échecs du capteur. On constate que cette hypothèse est très peu probable car les positions correspondantes ont des probabilités de l'ordre de 10^{-7}.

Intérêt de l'expression bayésienne de la localisation : En considérant la figure 4.24, il est important de remarquer que toutes les hypothèses d'identification des observations et d'échec des capteurs sont présentes dans cette figure. Cependant, grâce au mécanisme de fusion bayésienne, seules celles qui traduisent le plus de cohérence entre les données et le modèle sont gratifiées d'une haute valeur de probabilité. Contrairement aux approches de localisation par graphe de correspondances ou JCDA (cf. section 4.2), il n'est pas nécessaire de mettre en place un algorithme pour l'identification explicite des observations. Toutes les identifications sont envisagées en un seul calcul.

Limitation de l'approche Comme on peut le deviner en considérant l'équation 4.11, une des limitations de l'approche est sa complexité calculatoire.

Par ailleurs, le fait de réaliser une identification implicite des observations rend cette approche de localisation inutilisable pour construire une carte de l'environnement. Comme nous le verrons dans le chapitre 5, il est indispensable de calculer une identification explicite des observations pour estimer avec précision l'état de la carte des amers.

[6] gris sombre en monochrome.
[7] gris clair en monochrome.

4.3.2 Localisation bayésienne par diagnostic

a) Difficulté de l'expression de l'ignorance

Pendant l'écriture d'un modèle probabiliste tel que celui présenté ci-dessus, il arrive souvent que l'on ait besoin d'exprimer l'ignorance ou l'absence d'*a priori*. Dans ce cas, on a recours à la théorie de l'information de Shannon qui nous dit que la distribution contenant le moins d'information est la distribution uniforme. C'est donc cette distribution qui est choisie pour représenter l'ignorance. Cela n'est pas sans conséquences, la plus importante étant que pour pouvoir dire d'une variable qu'elle suit une loi uniforme, il faut qu'elle soit définie sur un sous ensemble mesurable d'un espace isomorphe à $\mathbb{N}$ ou à $\mathbb{R}^n$. En pratique, cela signifie qu'on ne peut exprimer son ignorance de cette façon que sur une variable dont le domaine est borné.

Une autre conséquence de cette utilisation de l'uniforme apparaît lorsque l'on veut comparer ou ajouter des probabilités. Pour peu que l'espace de définition de la variable considérée soit grand, la valeur de probabilité de la distribution uniforme devient vite très petite. Dans le modèle précédent, cela se produit quand on marginalise sur F_j (fig. 4.21 et 4.22). Dans ce cas, la contribution apportée par l'hypothèse d'échec du capteur est minime, mais cette faiblesse est principalement due à la taille de l'espace de définition de Z_j plutôt qu'à la faible proportion d'erreur du capteur. Mathématiquement, cela ne pose aucun problème. Toutefois, en pratique, les bornes de l'espace de définition sont souvent définies *a priori* de façon à limiter la taille de l'espace ou à pouvoir définir une distribution uniforme. Il faut donc être conscient du fait que ces choix de bornes ne sont pas sans conséquences sur les résultats.

Un des mérites de l'approche alternative de localisation probabiliste que nous allons évoquer dans la section suivante est de permettre une expression de l'absence d'*a priori* qui est moins influencée par les contraintes précédentes.

b) Modèle capteur de base

Par rapport à l'approche précédente, on introduit une différence majeure : le modèle d'observation unitaire. Le modèle de localisation globale est ensuite adapté à cette modification.

Lorsqu'on exprime $P(Z \mid C\ L)$, on exprime l'observation attendue sachant que le système est dans une configuration donnée, il s'agit donc d'une expression fonctionnelle. En mettant l'accent sur l'aspect fonctionnel, on introduit une dissymétrie dans la relation entre observation et configuration : l'observation devient une conséquence de la configuration.

Cependant, lorsqu'on cherche à localiser un robot à partir d'un ensemble d'observations, on cherche les configurations les plus cohérentes avec les observations. Pour traduire cette relation, nous allons utiliser les variables de diagnostic présentées en 2.3.5. Nous introduisons donc la variable booléenne de diagnostic I. Cette variable exprime l'événement "Z est cohérente avec C et L" et on s'intéressera à $P(I \mid Z\ C\ L)$.

En pratique, si on dispose d'un modèle h permettant de prédire une observation $\hat{Z}$ à partir de la configuration C et de l'environnement L,

$$\hat{Z} = h(C, L)$$

alors la probabilité de I sera proche de 1 si $Z \approx \hat{Z}$ et tendra vers 0 quand Z s'éloignera de $\hat{Z}$.

Pour obtenir ce comportement, on définit $P(I \mid Z\,C\,L)$ par un distribution en cloche du type :

$$P([I=1] \mid Z\ C\ L) = e^{(Z-h(C,L))^T P^{-1}(Z-h(C,L))} \tag{4.12}$$

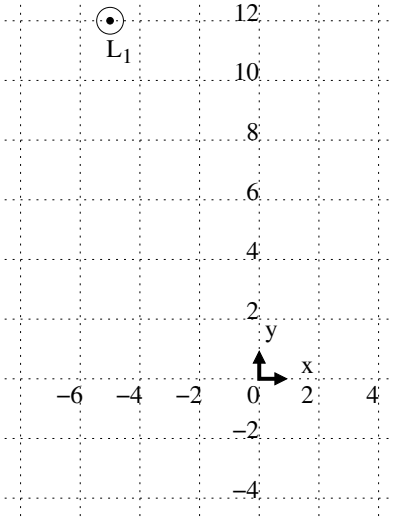
On définit ensuite le modèle conjoint suivant :

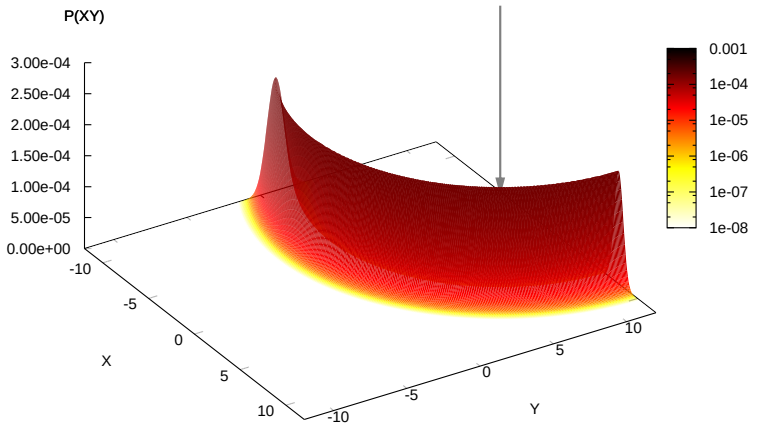
$$\begin{aligned} P(C\ Z\ L\ I) &= P(C)P(L)P(Z)P(I \mid Z\ C\ L) \\ \text{Question} &: P(C \mid Z\ L\ [I=1]) \end{aligned} \tag{4.13}$$

Avec C, Z et L indépendants et $P(C)$, $P(L)$ et $P(Z)$ uniforme. Il peut paraître surprenant, de prime abord, que C et Z soient indépendants. Cependant, il faut noter que ce n'est le cas que si l'on ignore s'ils sont exprimés dans le cadre du même modèle, c'est à dire si on ne connaît pas I. Ce modèle conjoint permet de répondre à des questions de localisation telles que $P(C \mid Z\,L\,[I=1])$, ce qui signifie que l'on veut déterminer les configurations C cohérentes ($[I=1]$) avec l'observation Z. La figure 4.25 illustre le résultat de cette question dans les mêmes conditions que pour le modèle de fusion présenté précédemment. On constate que bien que l'équation soit différente, la distribution résultante sur la localisation ne change pas.

Environnement : L

Amers : $L_1 = (-5, 12)$





Mesures : Z

$13.0\,m$

FIG. 4.25: Localisation par diagnostic (eq. 4.13) : A gauche, vue cartésienne de l'environnement. A droite, sur la distribution de probabilité $P(C \mid Z\ L\ [I=1])$, la flèche indique la position de l'amer L_1.

c) Modèle capteur intégrant la notion de fausse observation

En fait, c'est avec la variable F que la fusion par diagnostic prend tout son sens. Comme dans le modèle de fusion classique, cette variable booléenne indique si l'observation est le résultat d'une vrai mesure, ou d'un dysfonctionnement du capteur.

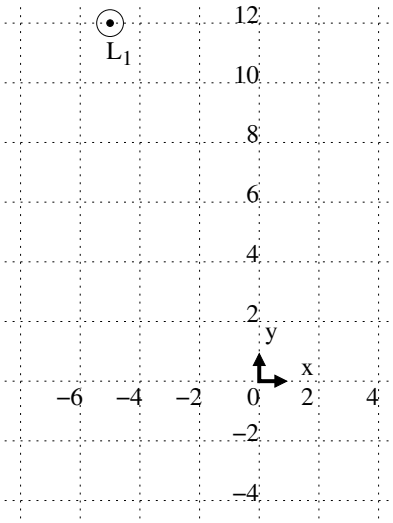
Pour exprimer $P(I \mid Z\ C\ F)$, deux cas doivent être envisager :
- Si $F = 1$: Z ne résulte pas d'un fonctionnement correct du capteur et n'apporte donc aucune information. On choisit donc $P(I_j \mid Z\ C\ F) = 0.5$.
- Si $F = 0$: Cette observation résulte d'un fonctionnement correct du capteur. On peut donc utiliser le modèle décrit par l'équation 4.13

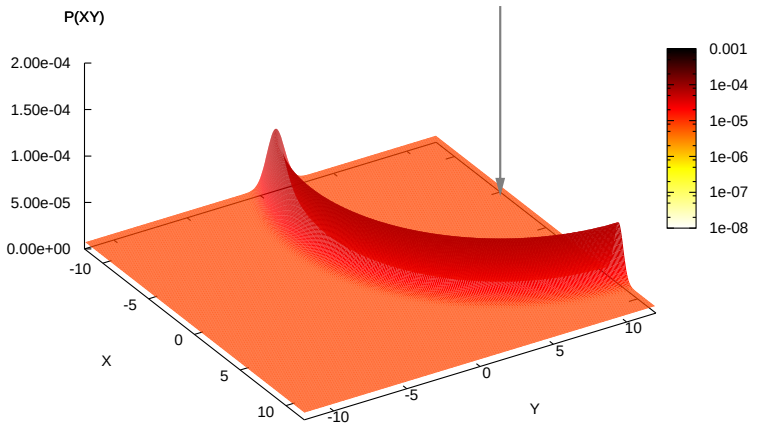
On obtient donc le modèle conjoint suivant :

$$P(C\ I\ Z\ L) = P(L)P(C)P(Z)P(F \mid C\ Z)P(I \mid C\ Z\ F) \tag{4.14}$$

Comme pour la fusion classique, on peut simplifier le modèle en exprimant $P(F)$ indépendemment de C et Z.

Environnement : L
Amers : $L_1 = (-5, 12)$





Mesures : Z
$13.0\,m$

FIG. 4.26: Localisation avec modèle d'échec par diagnostic (equ. 4.14) : A gauche, vue cartésienne de l'environnement. A droite, sur la distribution de probabilité $P(C \mid Z\ L)$, la flêche indique la position de l'amer L_1.

La figure 4.26 illustre l'influence de F sur la localisation. Par rapport à la figure 4.20, le changement principal est la variation de l'amplitude de la courbe (différence entre les probabilités maximales et minimales). Avec un modèle de diagnostic, cette différence est beaucoup moins importante qu'avec un mécanisme de fusion tel que celui décrit plus haut. Ceci vient du choix d'utiliser une distribution uniforme pour exprimer une connaissance minimale. Puisque la valeur d'une distribution uniforme est l'inverse de la mesure du domaine de la variable qu'elle décrit, dans une fusion classique, la valeur du quotient

$$\frac{\max_C P(C \mid Z[I = 1])}{\min_C P(C \mid Z[I = 1])}$$

dépend surtout du choix des domaines Z et C.

Au contraire, avec un modèle de diagnostic, l'expression de l'indétermination de la relation entre Z et C pour $F = 1$ s'exprime indépendamment des domaines de Z et C. Ainsi, le quotient précédent est lui aussi indépendant des domaines de Z et C. De plus, grâce à l'utilisation d'une variable de diagnostic, ces domaines peuvent éventuellement n'être ni bornés ni mesurables[8], on pourra tout de même décrire une relation indéterminée entre Z et C.

Comparaison des variables F et I : Une question se pose immédiatement en comparant F et I : quelle est la différence sémantique entre ces deux variables ? Rappelons tout d'abord leur définition : I décrit l'appartenance des variables diagnostiquées à l'espace de validité d'un modèle alors que F sépare les mesures capteur en deux catégories : les mesures normales et celles provenant d'un dysfonctionnement du capteur.

Sans F, toute mesure aberrante du capteur serait considérée comme hors modèle et la probabilité

$$P([C = \text{position réelle}] \mid [Z = \text{mesure aberrante}]\,[I = 1])$$

serait approximativement nulle. Au contraire, F permet d'intégrer les mesures aberrantes dans le modèle en ayant à la fois $[I = 1]$ et $[F = 1]$.

Ainsi, on laisse une place aux configurations incohérentes au vu d'une certaine mesure, au cas où cette mesure résulterait d'un dysfonctionnement du capteur ou d'un phénomène non modélisé dans l'environnement.

d) Modèle complet

Finalement, le système multi-observation qui résulte de l'équation 4.14 est le suivant :

$$\begin{aligned} P(C\,\vec{I}\,\vec{Z}\,\vec{W}\,\vec{F}\,\mathcal{L}) \;=\; & P(C)\,P(\mathcal{L}) \times \\ & \prod_j P(Z_j)P(F_j \mid C\,\mathcal{L})P(W_j \mid F_j\,C\,\mathcal{L})P(I_j \mid W_j\,F_j\,C\,\mathcal{L}) \end{aligned} \tag{4.15}$$

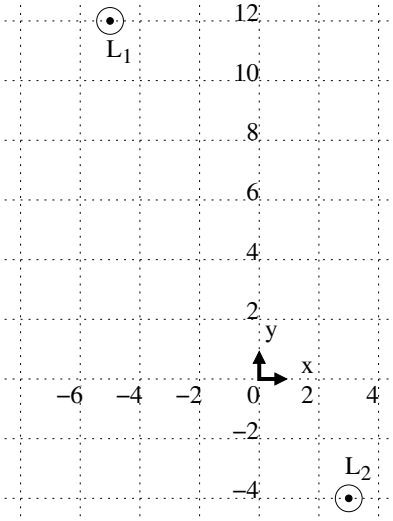
Intérêt d'une fusion par diagnostic La figure 4.27 illustre le résultat d'une localisation bayésienne en utilisant une fusion par diagnostic. En comparant avec le résultat obtenu avec une fusion classique (fig. 4.24), on constate d'abord que l'on retrouve les trois ensembles de positions correspondant respectivement aux cas où zéro, une ou deux mesures sont correctes.

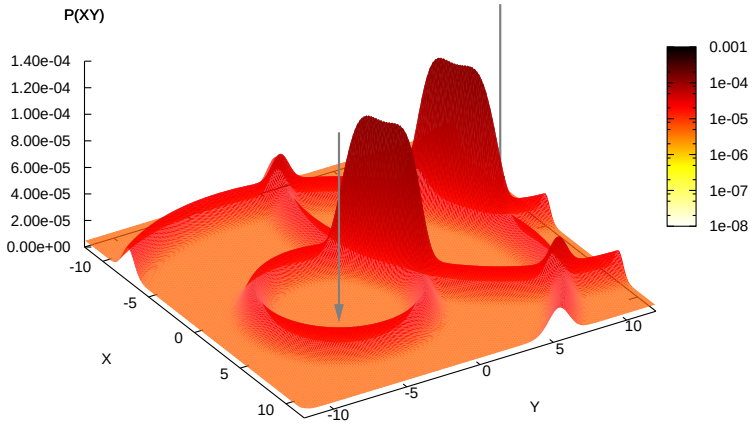
Cependant, les probabilités des hypothèses faibles (cercles correspondant à un échecs du capteur ou surface correspondant à deux échecs) sont bien moins faibles que dans la figure 4.24. Ainsi, le processus de fusion évalue ces hypothèse comme peu vraisemblables mais leur probabilité restent du même ordre de grandeur, Sur la figure 4.27, cette observation se traduit par une figure n'utilisant que des tons de rouge.

L'accumulation de cohérence entre observations met donc en évidence deux ensembles de configurations plus probables, mais sans écraser le reste du domaine de localisation.

[8] Pour répondre à la question : $P(C \mid Z\,L)$.

Environnement : L
Amers : $L_1 = (-5, 12)$
$L_2 = (3, -4)$





Mesures : Z
$5.0\,m$ et $13.0\,m$

FIG. 4.27: Localisation complète par diagnostic (eq. 4.15) : A gauche, vue cartésienne de l'environnement. A droite, sur la distribution de probabilité $P(C \mid \vec{Z}\,\mathcal{L}\,[\vec{I} = \vec{1}])$, les flèches indiquent les positions des amers L_1 et L_2.

L'intérêt de cette propriété apparaît clairement lorsque la distribution de probabilité sur la localisation est intégrée dans un filtre bayésien. Dans ce cas l'utilisation d'une fusion sous forme de diagnostic permet d'améliorer la robustesse aux fausses observations et de rendre la localisation insensible à la taille de l'espace d'observation. Ces résultats sont présentés avec plus de détails dans la section 4.3.3.

4.3.3 Intégration dans un filtre bayésien

a) Rappel sur les filtres bayésiens

En général, un filtre bayésien (cf. section 2.3.2) est utilisé pour estimer une variable d'état au cours du temps, dans le cadre d'un modèle de Markov d'ordre 1, non observable. Formellement, l'expression générique d'un tel filtre est la suivante :

$$P(C_t \mid Z_t \dots Z_0) \propto P(Z_t \mid C_t) \int_{C_{t-1}} P(C_t \mid C_{t-1}) P(C_{t-1} \mid Z_{t-1} \dots Z_0) \tag{4.16}$$

où $P(Z_t \mid C_t)$ est un modèle de la relation entre observation et configuration, tel que ceux définit plus haut ($P(Z \mid C)$ ou $P(Z \mid C[I = 1])$), et $P(C_t \mid C_{t-1})$ est le modèle de transition du système, résultant de l'hypothèse de Markov d'ordre 1.

Dans le cas particulier d'un processus de localisation pour un robot mobile, le modèle de transition est lié au modèle cinématique du système, c'est à dire le modèle qui estime le mouvement à partir des commandes appliquées.

b) Localisation par fusion ou par diagnostic

Pour illustrer l'influence du choix de description du modèle de fusion, nous avons choisi le système trivial suivant : un système unidimensionnel avançant d'une unité par unité de temps et capable d'observer directement son état. Ce système est illustré par la figure 4.28. Malgré

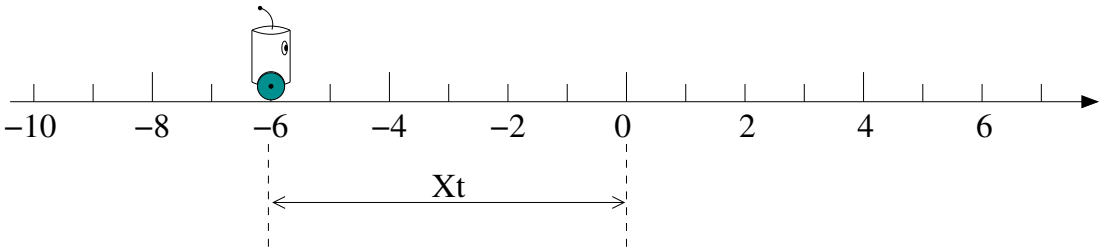


FIG. 4.28: Système unidimensionnel trivial.

la simplicité de ce système, les conclusions que nous allons en tirer illustrent l'essence de la différence entre un modèle de diagnostic et un modèle de fusion classique.

Les figures 4.29 et 4.30 présentent deux programmes bayésiens modélisant ce système suivant un schéma de fusion, soit classique soit par diagnostic. **Ces deux programmes sont paramétrés par la taille $2N$ des supports de X et Z.**

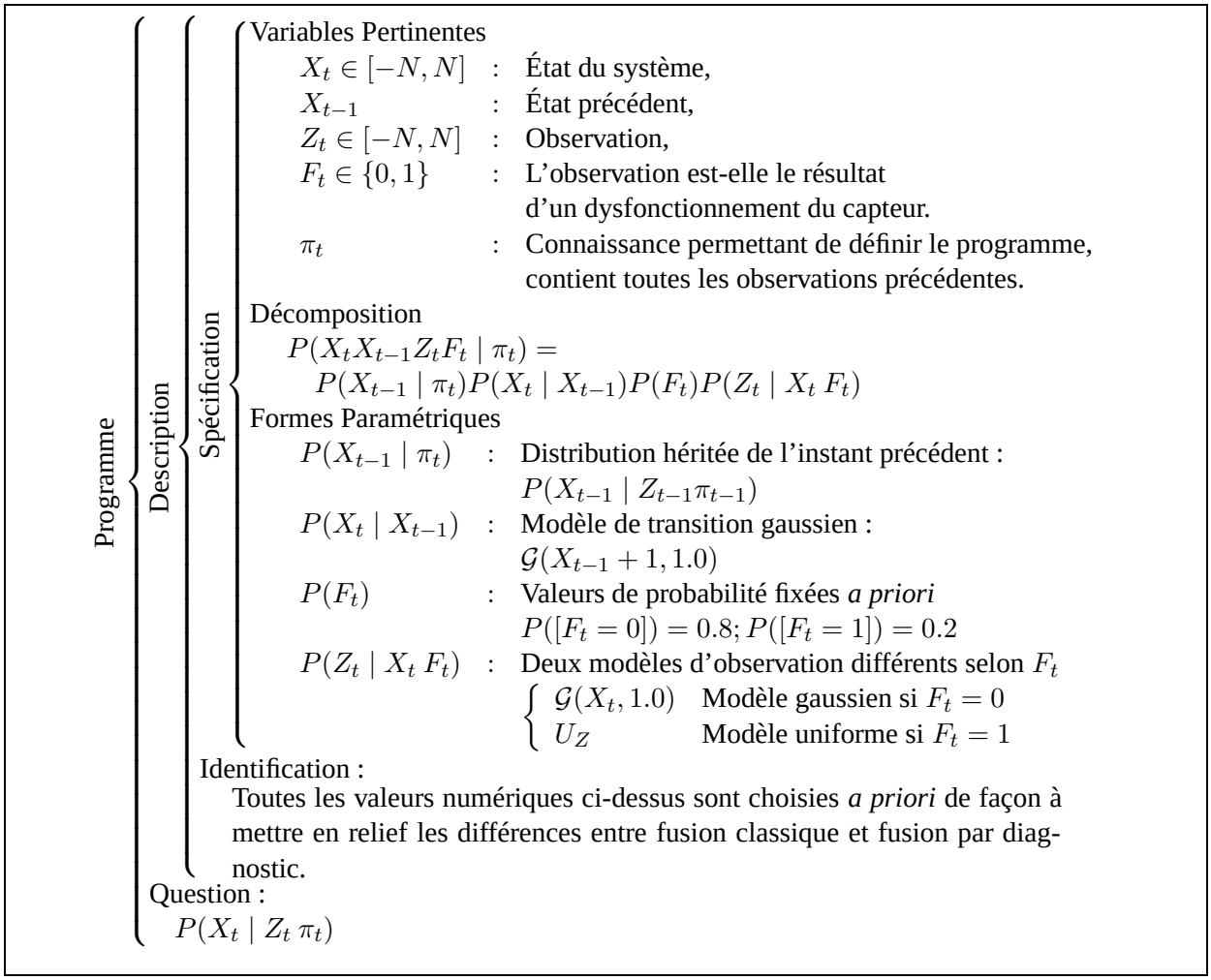


FIG. 4.29: Filtre bayésien par fusion classique

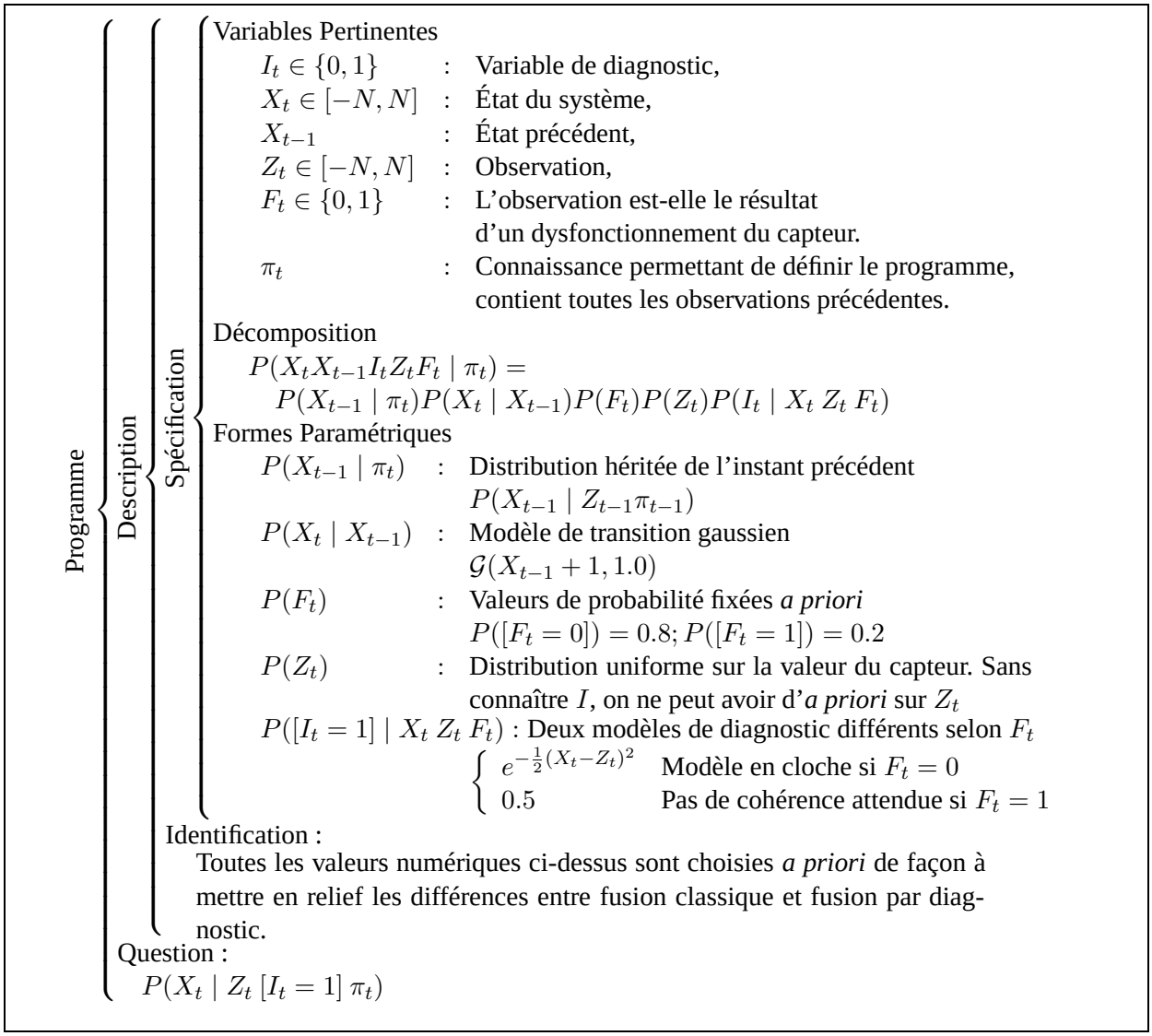


FIG. 4.30: Filtre bayésien par diagnostic

c) Robustesse de la localisation par diagnostic

Les figures 4.31 à 4.34 montrent l'évolution des deux filtres pour les mêmes observations pour deux valeurs de N : $N = 20$ et $N = 200$.

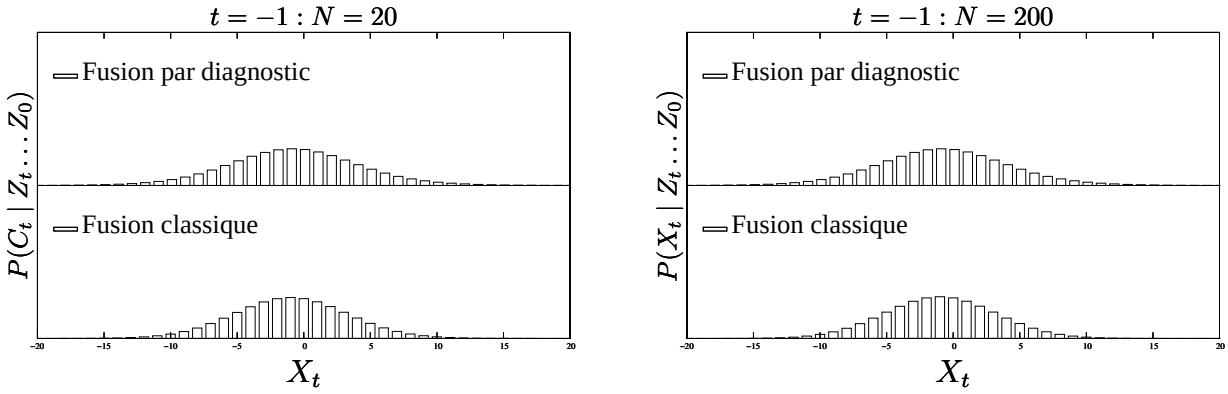


FIG. 4.31: État initial des filtres bayésiens à $t = -1$.

1. Au départ ($t = -1$), tous les filtres sont dans le même état.

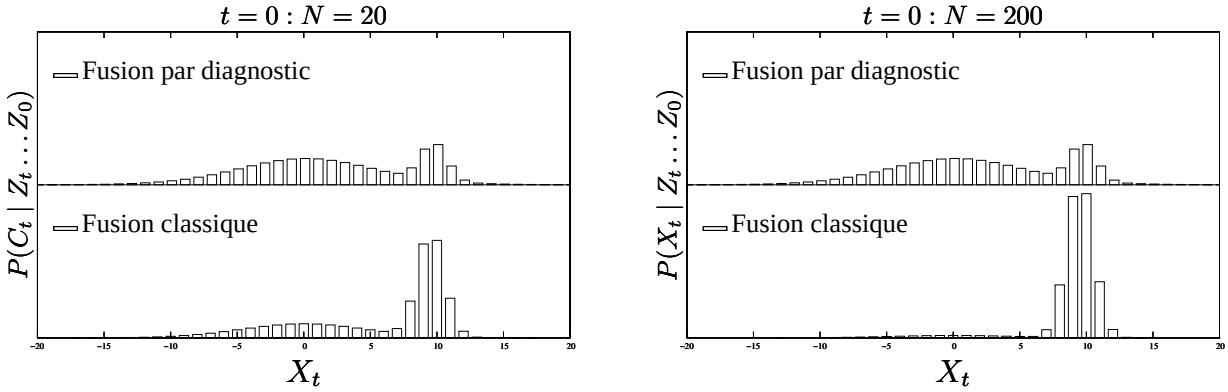


FIG. 4.32: État des filtre après observation d'un faux-positif à $t = 0$.

2. A $t = 0$, une observation incorrecte est réalisée. On peut observer que la réaction du modèle de fusion par diagnostic est indépendante du choix de N alors qu'avec un modèle de fusion classique, l'hypothèse d'échec du capteur est complètement sous-évaluée pour $N = 200$.

   Cette différence d'évaluation est due à la définition de la distribution $P(Z_t \mid X_t\,[F_t = vrai])$ comme une distribution uniforme. La valeur de probabilité de cette distribution est donc divisée par 10 lorsque N passe de 20 à 200, ce qui divise par autant le poids de l'hypothèse d'échec.

3. Après avoir propagé (sans nouvelle observation) l'état du système jusqu'à $t = 4$, on constate que le filtre par diagnostic est centré autour de $X_t = 4$ (position correcte) avec une très large incertitude, quel que soit N. Cette forme de distribution provient de l'étalement des deux pics à $t = 0$ par l'application du modèle de transition gaussien.

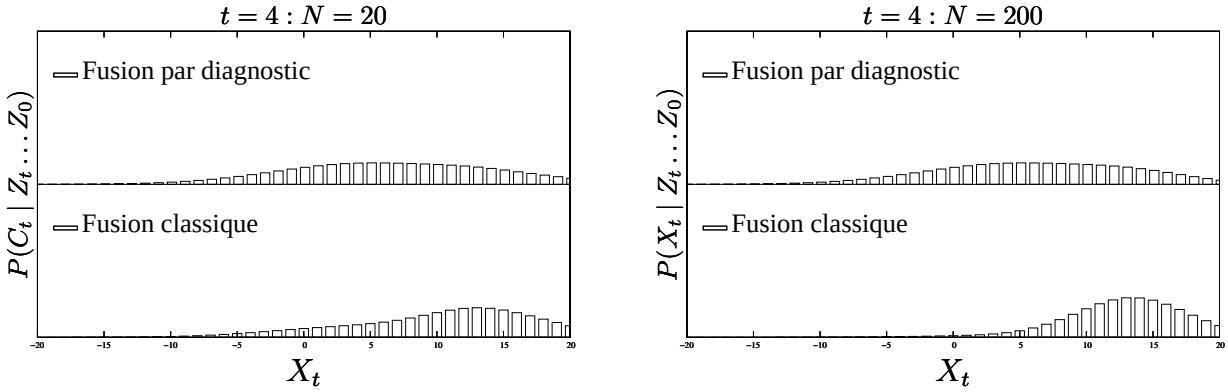


FIG. 4.33: État des filtres après propagation de l'observation d'un faux-positif à $t = 4$.

Au contraire, pour $N = 200$, le filtre classique est centré complètement sur la localisation résultant de la mesure incorrecte. Ce résultat est la conséquence de la faiblesse de l'hypothèse d'échec du capteur à $t = 0$.

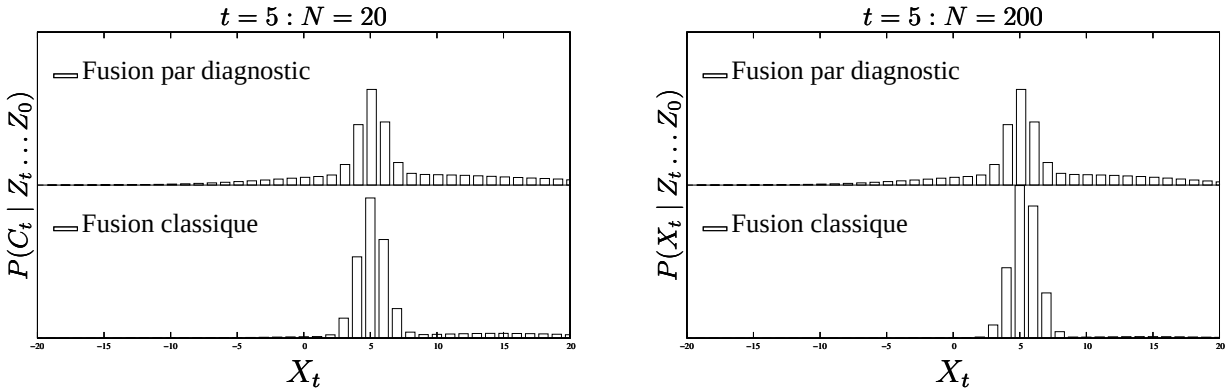


FIG. 4.34: État des filtres après une observation correcte à $t = 5$.

4. Après une nouvelle observation correcte, les deux filtres se réconcilient mais la fusion classique élimine de nouveau toutes les hypothèses alternatives pour $N = 200$.

Le but de cet ensemble de figures n'est pas de mettre l'accent sur les différences de comportement des deux modèles mais surtout sur leur différence de sensibilité à la taille de l'espace d'observation. La principale raison de cette différence de sensibilité provient de la différence d'expression de l'ignorance. Dans le cas de la fusion classique, celle-ci est exprimée par une distribution uniforme dont la **valeur** dépend de N alors qu'avec un modèle de diagnostic, l'ignorance est exprimée par rapport à I indépendamment de N.

La relative insensibilité de la fusion par diagnostic lui permet de travailler sur des espaces très grands quitte à ne réaliser l'inférence que sur une partie de l'espace. C'est ce que nous aurions pu faire dans ces figures, en ne calculant la distribution de localisation que sur la partie de l'espace représentée $X_t \in [-20, 20]$, quel que soit N.

4.3.4 Conclusions

L'expression du problème de localisation, à l'aide du formalisme de la programmation bayésienne, nous a menés à deux résultats : d'une part le problème de l'identification des observations s'est dissout dans l'inférence bayésienne, et d'autre part, nous avons converti l'incertitude de cette identification en incertitude sur la localisation. Ces résultats ont été obtenus en conservant des temps d'exécution compatibles avec le temps réel, ce qui nous a permis d'utiliser cet algorithme dans l'application de suivi de trajet sensorimoteur, qui sera détaillée au chapitre 7.

Par ailleurs, ce chapitre nous a permis de mettre en évidence l'influence considérable de la modélisation de l'ignorance dans un système de localisation robuste, ainsi que l'influence éventuelle de la taille de l'espace d'observation dans la prise en compte de cette ignorance. Selon le type de fusion utilisé pour décrire le processus de localisation, la prise en compte de l'ignorance sera plus ou moins sensible à la dimension de l'espace. Pour une application de localisation où la prise en compte des dysfonctionnements potentiels du capteur est particulièrement importante, il nous paraît donc plus judicieux d'utiliser un modèle de fusion par diagnostic.

Cependant, contrairement à la localisation par triangulation présentée en 4.2, un algorithme de localisation probabiliste avec mise en correspondance implicite ne permet pas de construire une représentation de l'environnement sous forme de carte des amers. En effet, l'identification des amers est un élément indispensable à la construction d'une telle carte. La localisation probabiliste, présentée dans cette section, ne pourra donc être utilisée que dans un contexte où la carte des amers est connue, ou bien lorsque l'on veut estimer une différence de point de vue entre deux observations. Nous verrons dans la prochaine section et dans les applications du chapitre 7 que la localisation sur un trajet sensorimoteur correspond tout à fait à ce dernier cas.

4.4 Localisation sur une trajectoire sensorimotrice

Dans le contexte de la réalisation d'un trajet, le rôle de la localisation est de fournir un diagnostic de la bonne exécution du trajet anticipé. Connaissant l'erreur entre ses états réels et observés, le système peut alors prendre des mesures correctives.

Dans cette section, nous envisageons une application illustrée par la figure 4.35, où

- le système connaît un trajet sensorimoteur d'un point A à un point B,
- désire rejoindre le point B en suivant autant que possible le trajet connu,
- **mais** part d'un point C dans le voisinage du trajet $A \rightarrow B$,
- **et** ne connaît pas de modèle allocentré de l'environnement.

Si on considère un trajet géométrique (séquence de points de l'espace de travail ou de l'espace des configurations) et une carte de l'environnement, cette application se réalise simplement à partir d'algorithmes de localisations tels que ceux présentés en sections 4.2 et 4.3.

La difficulté réelle apparaît lorsque l'on considère un trajet sensorimoteur sans carte, et donc sans pouvoir mettre en place une localisation absolue.

Définition 19 – Indice de référence et position temporelle *Soient un trajet sensoriel $T : I \rightarrow \mathbb{O}$ et une observation Z. L'* indice de référence $I^T_{ref}(Z)$ *associé à Z est l'indice de la position de*

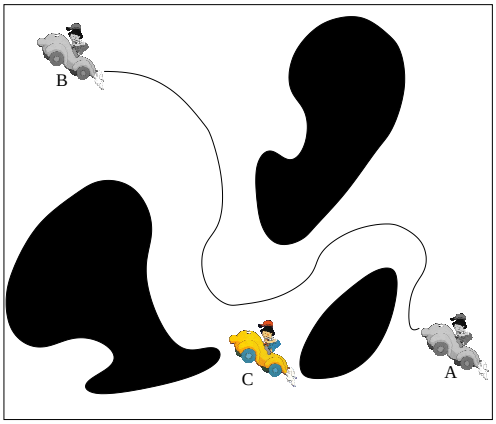


FIG. 4.35: Application considérée dans la section 4.4 : le trajet sensorimoteur $A \rightarrow B$ est connu mais le véhicule part d'une position C dans un voisinage du trajet.

T considérée comme la plus proche de Z. Formellement :

$$I_{ref}^{T}(Z) = \arg\min_{i} \|T(i) - Z\|$$

On nomme position temporelle $\tau(t)$ *au temps t l'indice de référence associé à la perception du robot au temps t.*

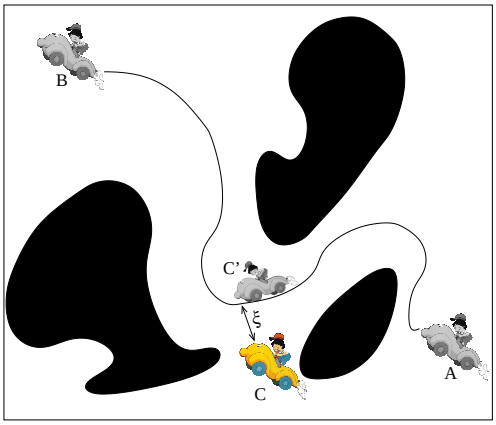


FIG. 4.36: Indice de référence C' et erreur de suivi ξ associée à la position C.

A titre d'exemple, dans la figure 4.36, l'indice de référence associé à la position initiale C est l'indice de la position C'.

Définition 20 – Différence de point de vue *Soit Z_1 l'observation réalisée par le robot dans une configuration C_1, et Z_2 l'observation d'une même scène réalisée depuis une configuration C_2. On nomme* différence de point de vue *entre Z_1 et Z_2 la différence de configuration $C_2 - C_1$.*

Définition 21 – Erreur de suivi *Soient un trajet $T : I \rightarrow \mathbb{O}$ et une observation $Z(t)$ au temps t. L'erreur de suivi $\xi(t)$ au temps t est la différence de point de vue entre $T(\tau(t))$ et $Z(t)$.*

Dans la figure 4.36, l'erreur de suivi en C' est $C' - C$.

Problème considéré : Formellement, le problème considéré dans cette section se décompose en deux parties :

- Le premier problème est la localisation temporelle initiale : étant donné un trajet $A \rightarrow B$ et une observation initiale $Z(0)$, déterminer la position temporelle initiale $\tau(0)$.
- Le second problème est l'estimation simultanée de l'erreur de suivi $\xi(t)$ et de la position temporelle $\tau(t)$ au cours du mouvement, en fonction des observations $Z(t)$.

Rappel de définition On définit une *trajectoire sensorimotrice* comme une fonction du temps dans le produit cartésien de l'espace de commande du robot $\mathbb{U}$ et de son espace d'observation $\mathbb{O}$. Formellement :

$$\mathcal{T}_{sm} : [0, t_1] \longrightarrow \mathbb{O} \times \mathbb{U}$$

Par la suite, nous noterons $\mathcal{T}_{sm}(t) = [\mathcal{T}_{sm}(t).Z, \mathcal{T}_{sm}(t).U)]$.

4.4.1 Localisation temporelle initiale

Le problème de la localisation temporelle initiale est de comparer l'observation initiale $Z(0)$ aux observations prévisibles dans un voisinage de la trajectoire $\mathcal{T}_{sm}$. En réalisant cette comparaison, on veut déterminer quel est l'indice de la perception de la trajectoire qui ressemble le plus à la perception courante.

En utilisant les principes de la programmation bayésienne, on peut calculer une distribution de probabilité sur la position temporelle initiale. La figure 4.37 formalise le programme bayésien qui réalise cette tâche.

a) Conception des modèles bayésiens

*1.***Modèle capteur** Le modèle de prévision des observations doit exprimer quelles observations sont attendues autour de $\mathcal{T}_{sm}$. Pour atteindre ce but, il est tout d'abord nécessaire de définir un modèle de capteur probabiliste.

Étant donné une observation de référence Z_{ref} et une différence de point de vue ξ, on suppose connaître un modèle déterministe H capable de prévoir l'observation Z_{prev} correspondant à ξ :

$$Z_{prev} = H(Z_{ref}, \xi)$$

Par exemple, dans notre contexte applicatif, H est la composition d'une rotation et d'une translation.

Le modèle probabiliste est défini par une distribution gaussienne centrée sur le modèle déterministe. Il exprime ainsi la distribution de probabilité sur les observations attendues :

$$P_m(Z_{prev} \mid Z_{ref}\ \xi) = \mathcal{G}(H(Z_{ref}, \xi), \Sigma_m) \tag{4.17}$$

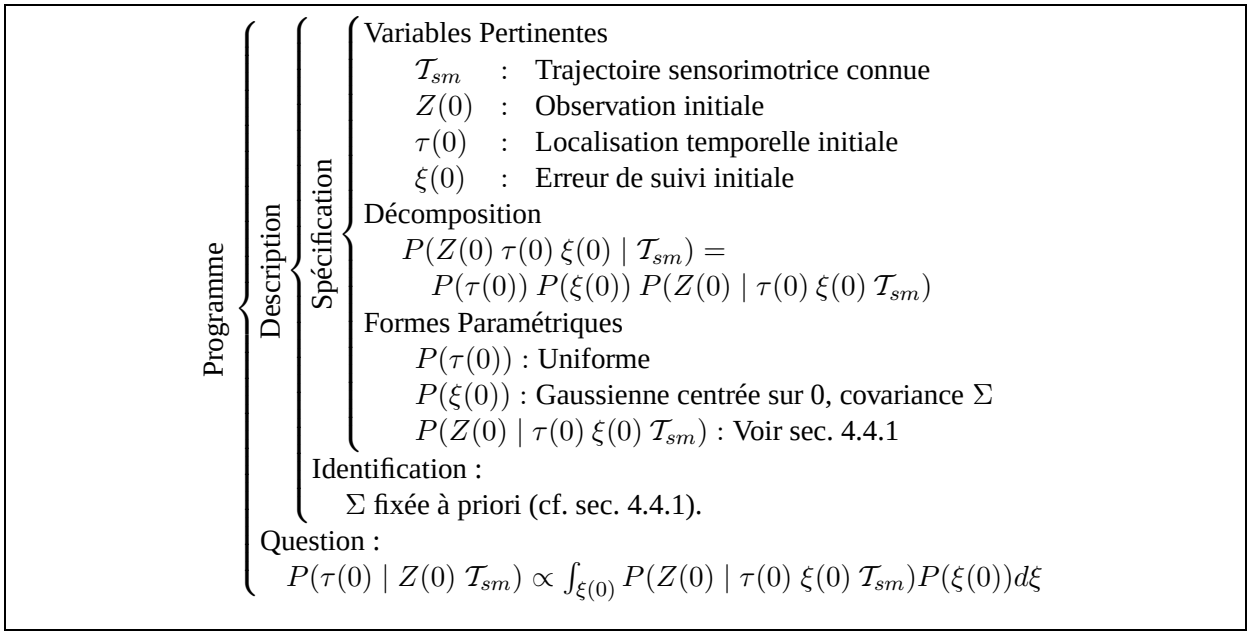


FIG. 4.37: Programme bayésien pour la localisation temporelle initiale

Dans cette équation, Σ_m représente la matrice de covariance de la distribution gaussienne. Cette covariance est le reflet de la précision du capteur réel et de la confiance que l'on a dans une mesure.

Si l'on souhaite utiliser un modèle plus complexe tenant compte des possibilités d'échec du capteur ou des problèmes d'identification des observations, on peut aussi utiliser les modèles capteurs présentés dans la section 4.3 sur la localisation sans mise en correspondance.

A partir de ce modèle, on définit :

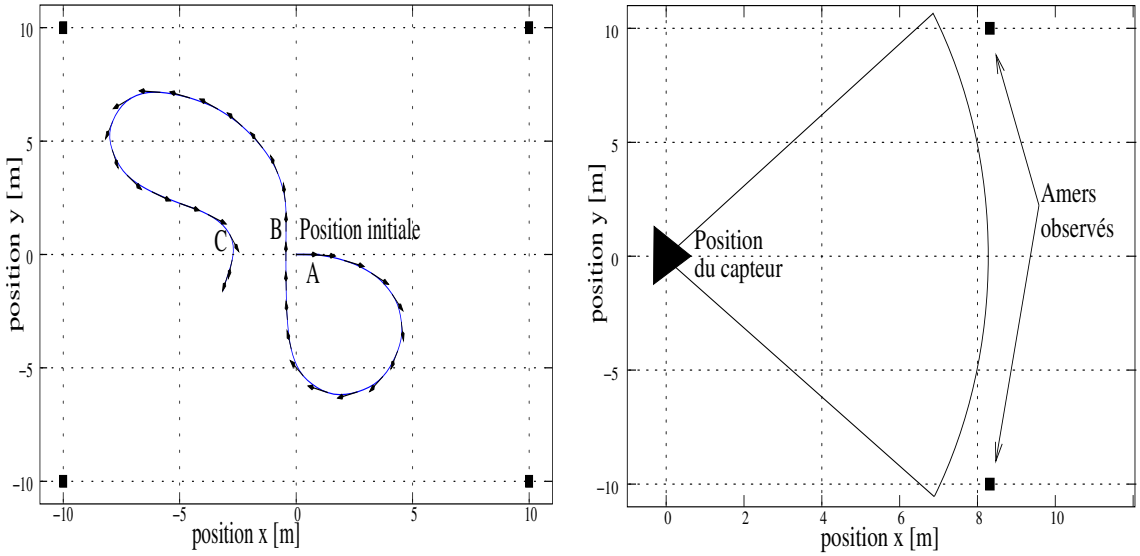
$$P(Z(0) \mid \tau(0)\,\xi(0)\,\mathcal{T}_{sm}) = P_m(Z(0) \mid [Z_{ref} = \mathcal{T}_{sm}(\tau(0)).Z]\,\xi(0)) \tag{4.18}$$

*2.**A-priori* sur l'erreur de suivi initiale** Pour exprimer le fait que nous nous intéressons à ce qui est attendu *autour* de la trajectoire nominale, $P(\xi)$ est définie comme une gaussienne centrée en zéro dont la covariance formalise l'expression "autour de $\mathcal{T}_{sm}$". En effet, plus la covariance est petite, plus $P(\xi)$ est piquée et moins les grandes différences de point de vue seront considérées pour identifier la localisation temporelle initiale.

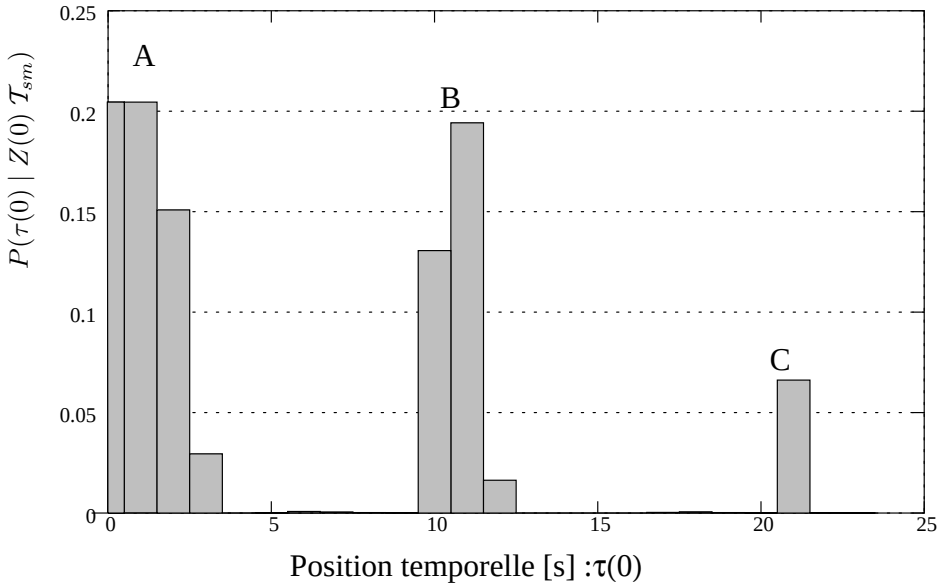
Toutefois, il est nécessaire de ne pas prendre une gaussienne trop large car cela correspondrait à accepter de très grandes différences de point de vue et rendrait peu précise l'estimation de la localisation temporelle initiale.

b) Initialisation à partir de $P(\tau(0) \mid Z(0)\,\mathcal{T}_{sm})$

En utilisant le programme bayésien présenté dans la figure 4.37, on peut calculer une approximation numérique de $P(\tau(0) \mid Z(0)\,\mathcal{T}_{sm})$, nommée $\hat{P}$ par la suite. La figure 4.38 donne un exemple typique d'une distribution ainsi obtenue.



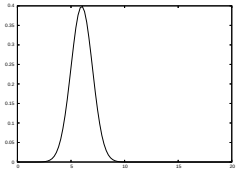
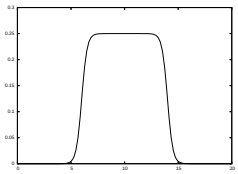
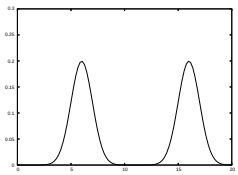
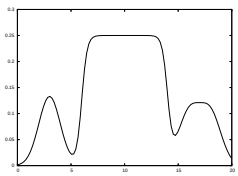
A gauche : Trajet initial dans un environnement avec quatre amers sur un carré (carrés noirs). Pour plus de clarté, le trajet géométrique est représenté ici, mais c'est le trajet sensorimoteur qui est utilisé par la localisation. **A droite** : Perception initiale dans le repère capteur.



Estimation de la localisation temporelle initiale $P(\tau(0) \mid Z(0)\ \mathcal{T}_{sm})$. Les trois hypothèses exprimées par les trois pics de la distribution correspondent aux points de vue A, B et C sur le tracé du trajet. La position réelle est la position A.

FIG. 4.38: Exemple de localisation temporelle initiale $P(\tau(0) \mid Z(0)\ \mathcal{T}_{sm})$: cas d'un robot planaire observant des amers (carrés noirs) avec un télémètre laser.

A partir de cette distribution, il nous faut extraire une valeur unique pour t qui sera notre estimation initiale $\tau(0)$. Plusieurs cas sont possibles :

	1- $\hat{P}$ est unimodale et fortement piquée. Dans ce cas, il n'y a pas d'ambiguïté et $\tau(0)$ est l'espérance de la distribution.
	2- $\hat{P}$ est unimodale, mais largement étalée (par exemple, une gaussienne avec un écart type important, ou un créneau large). Dans ce cas, on peut utiliser l'espérance de la distribution comme valeur $\tau(0)$. Cette situation est souvent due à une pause dans la trajectoire de référence, se traduisant par une série d'observations similaires.
	3- $\hat{P}$ a plusieurs pics bien marqués. Cela révèle des ambiguïtés perceptives dans la trajectoire nominale. Dans ce cas, il faut choisir un pic ou signaler un échec. Le choix peut se faire selon différents critères (largeur ou hauteur des pics par exemple). Pour lever l'ambiguïté, il peut aussi être nécessaire de lancer quelques manoeuvres de perception active.
	4- $\hat{P}$ est composée de plusieurs dômes ou est proche d'une distribution uniforme. Ceci est souvent dû au fait que le système ne peut trouver d'observations de $\mathcal{T}_{sm}$ proches de l'observation courante. On peut supposer que le robot n'est pas dans un voisinage de $\mathcal{T}_{sm}$ ou que l'environnement n'est pas reconnaissable. Le plus sage est sans doute de générer un rapport d'échec.

Perception active Lorsque la première perception amène à une localisation temporelle initiale ambiguë, on suppose qu'en se déplaçant le robot accumulera de nouvelles observations qui lui permettront de désambiguïser sa situation. Une question se pose alors : quel est le mouvement ou la séquence de mouvements qui permettra de maximiser le gain d'information ? Ce problème complexe est connu sous le nom de perception active. Nous n'avons pas eu le temps de traiter ce sujet passionnant en détails, mais nous pouvons citer quelques références auxquelles le lecteur intéressé pourra se reporter : *[Baccon et al. 2002, Bourgault et al. 2004, Mitsunaga & Asada 2002, Soyer et al. 2002]*.

4.4.2 Localisation au cours du temps

Une fois que l'on connaît la position initiale dans le trajet *sensorimoteur*, on peut commencer à suivre ce dernier. Deux variables doivent alors être estimées : la position temporelle $\tau(t)$ et l'erreur de suivi $\xi(t)$. A partir de ces variables, on pourra calculer les contrôles qui permettent de suivre la trajectoire.

a) Approche théorique : utilisation d'un filtre bayésien

L'objectif de cette section étant d'estimer une variable au cours du temps, on pense immédiatement à un filtre bayésien. Pour cela, on pose $X(t) = [\tau(t)\ \xi(t)]$. On doit alors définir un modèle de transition $P(X(t+dt) \mid X(t))$ et un modèle d'observation $P(Z(t) \mid X(t))$.

*1.***Modèle d'observation** Le modèle d'observation est celui que nous avons utilisé pour la localisation temporelle initiale (eq. 4.18) :

$$P(Z(t) \mid \tau(t)\ \xi(t)\ \mathcal{T}_{sm}) = P_m(Z(t) \mid [Z_{ref} = \mathcal{T}_{sm}(\tau(t)).Z]\ \xi(t)) \tag{4.19}$$

*2.***Modèle de transition** Pour prévoir $X(t+dt)$ à partir de $X(t)$, on doit prévoir $\tau(t+dt)$ et $\xi(t+dt)$:

- Le modèle de transition pour la position temporelle est construit en supposant que la réalisation du trajet se fait de manière synchrone avec sa représentation. Dans ce cas, on peut espérer $\tau(t+dt) = \tau(t) + dt$. Pour tenir compte des incertitudes de cette réalisation, on utilise un modèle gaussien :

$$P(\tau(t+dt) \mid \tau(t)) = \mathcal{G}(\tau(t) + dt, \sigma_\tau) \tag{4.20}$$

- Le modèle déterministe de transition de l'erreur de suivi est construit à partir du modèle cinématique K du robot et de la connaissance des commandes décrites par la trajectoire sensorimotrice $\mathcal{T}_{sm}(t).U$ et appliquées par le robot $U(t)$:

$$\hat{\xi}(t+dt, \tau(t)) = K(\xi(t), U(t), dt) - K(0, \mathcal{T}_{sm}(\tau(t)).U, dt)$$

  On utilise encore une fois un modèle gaussien pour construire le modèle probabiliste de transition :

$$P(\xi(t+dt) \mid \xi(t)\ \tau(t)\ \mathcal{T}_{sm}) = \mathcal{G}(\hat{\xi}(t+dt, \tau(t)), \sigma_\xi) \tag{4.21}$$

*3.***Filtre complet** Avec les modèles précédents, on obtient l'équation complète du filtre bayésien :

$$\begin{aligned} &P(\tau(t+dt)\ \xi(t+dt) \mid Z(t+dt) \dots Z(0)) \propto \\ &\quad P(Z(t+dt) \mid \tau(t+dt)\ \xi(t+dt)\ \mathcal{T}_{sm}) \\ &\quad \int_{\tau(t)} \left[P(\tau(t+dt) \mid \tau(t)) \int_{\xi(t)} P(\xi(t+dt) \mid \xi(t)\ \tau(t)) P(\tau(t)\xi(t) \mid Z(t) \dots Z(0)) d\xi \right] d\tau \end{aligned} \tag{4.22}$$

b) Implantation pratique : approximations

Étant donné la complexité de l'expression complète du filtre bayésien, nous ne pouvons utiliser directement cette formulation pour une application embarquée. Deux approximations sont possibles :

- On peut utiliser une approximation linéaire des modèles d'observation et de transition (on utilisera alors un filtre de Kalman étendu *[Lewis 1986]*). Cette approximation du modèle de transition ne serait pas abusive, cependant, l'approximation gaussienne n'est pas justifiable en utilisant les modèles de localisation bayésienne de la section 4.3.
- La seconde approximation envisagée est la séparation des estimations de position temporelle et d'erreur de suivi. La position temporelle est estimée en utilisant l'heuristique présentée ci-dessous. L'erreur de suivi est, quant à elle, estimée en supposant connue la position temporelle, par un filtre bayésien implanté sous la forme d'un filtre à particules (cf. section 2.3.4).

Approximation de l'estimation de position temporelle Pour s'affranchir du coût calculatoire de l'application directe du filtre bayésien, il est préférable d'utiliser une approximation, grossière mais efficace, pour estimer la position temporelle au cours du temps : tant que l'erreur de suivi est suffisamment petite ($|\xi(t)| \leq \xi_{\lim}$), nous supposons que la trajectoire est exécutée de façon synchrone par rapport à la trajectoire de référence, c'est à dire que $\tau(t+dt) = \tau(t) + dt$. Quand l'erreur devient trop grande, on garde τ constant, donc $\tau(t+dt) = \tau(t)$.

L'hypothèse d'exécution synchrone tant que l'erreur est petite est une hypothèse classique dans les algorithmes de suivi de trajectoire. C'est une hypothèse judicieuse car elle permet de conserver a peu près les profils de vitesse prévus sur la trajectoire. Lorsque l'erreur augmente de façon monotone, cela indique que quelque chose (limitations moteurs ou physiques) empêche le suivi de la trajectoire. Il est donc inutile de continuer à faire avancer l'index de la position de référence tant que le système n'a pas rattrapé son retard. En pratique, cela se traduit par le maintien de τ constant.

4.4.3 Conclusion

En utilisant le formalisme de la programmation bayésienne et en particulier la notion de filtre bayésien, nous avons donc décrit un processus de localisation spatio-temporelle autour d'une trajectoire sensorimotrice. Cette localisation, combinée avec les algorithmes de contrôle présentés dans le chapitre 6, nous permettra de construire une application complète de suivi de trajectoire sensorimotrice dans le chapitre 7.

Revenons maintenant sur un point laissé en suspend dans les paragraphes précédents. Même si la distribution de probabilité sur la localisation temporelle initiale est fortement piquée, il n'y a pas de garantie que l'observation courante corresponde effectivement à une partie de $\mathcal{T}_{sm}$. Cependant, sans informations supplémentaires, il nous semble indispensable de commencer le suivi comme si nous croyions à l'estimation initiale, tout en gardant à l'esprit que les observations futures peuvent remettre en cause ou rendre invalide cette hypothèse de localisation initiale.

Pour pouvoir concrétiser ces notions de "croire", "remettre en cause" ou "rendre invalide", nous allons maintenant voir comment décrire la notion de confiance en soi dans le formalisme bayésien et comment faire évoluer cette grandeur au cours du temps en fonction des observations réalisées par le robot.

4.5 Qualité de la localisation et confiance en soi

En consultant la littérature, on découvre un grand nombre d'expérimentations implémentant un cas particulier de navigation intentionnelle, que ce soit sous l'angle de l'automatique, de la localisation ou de la planification. La plupart d'entre elles démontrent le succès de la navigation intentionnelle par une analyse humaine des traces de la trajectoire. Il est très rare que le système soit capable de prendre du recul sur sa réalisation, et diagnostique seul son succès ou son échec. Dans cette section, nous allons décrire une approche probabiliste au diagnostic pour un robot mobile navigant vers un objectif.

Définition 22 – Certitude *Formellement, la* certitude *du système à l'instant t sera représentée par la variable booléenne Cert$(t) \in \{0, 1\}$ dont la sémantique est la suivante :*

- *lorsque Cert$(t) = 1$, le système est complètement sûr de sa localisation à l'instant t (on peut donc supposer qu'il a accumulé une série d'indices le confortant dans son estimation) ;*
- *au contraire, lorsque Cert$(t) = 0$, le système est presque sûr que ces hypothèses de localisation sont fausses. Plusieurs origines à cette situation sont envisageables : l'environnement n'est plus reconnaissable, la localisation initiale était incorrecte...*

Définition 23 – Confiance *Nous nommerons* Confiance *(ou* Confiance en soi*) la probabilité que la certitude soit vraie : $P($Cert$(t) = 1)$, ce qui est équivalent à l'espérance de la certitude $E[$Cert$(t)]$. On notera Confiance(t) la valeur de la confiance en soi à l'instant t.*

La confiance en soi est donc une variable réelle comprise entre 0 et 1.

Le problème considéré dans cette section est le suivant : étant donné un modèle de l'environnement et une séquence d'observations jusqu'à l'instant t, comment estimer la confiance de système en lui-même Confiance(t).

Dans de nombreux problèmes où une estimation d'état est nécessaire, la comparaison de modèles est utilisée pour *diagnostiquer* l'état de confiance du système (voir par exemple *[Murphy 1998]* ou *[Lerner et al. 2000]*). Après en avoir décrit le principe, nous verrons comment la comparaison de modèle peut être utiliser pour qu'un robot mobile estime sa confiance en lui au fur et à mesure de ses observations.

Notations : Dans la suite de cette section, au temps t, nous considérerons un robot dont la variable de localisation est X_t et dont la variable d'observation est notée Z_t. On notera de plus $\mathcal{Z}_t = Z_0 \dots Z_t$.

4.5.1 Comparaison de modèles

Supposons que l'on travaille avec une variable A qui peut être évaluée par deux modèles différents. Par exemple l'odométrie d'un robot à roue sera estimée différemment selon que ses pneus sont dégonflés ou non. Pour décrire cette situation, on peut construire la distribution conjointe suivante :

$$P(A \textit{ Modèle}) = P(\textit{Modèle})P(A \mid \textit{Modèle})$$

$P(\textit{Modèle})$ exprime notre *a priori* sur *le meilleur de ces modèles* (*a priori* souvent uniforme en pratique), et $P(A \mid \textit{Modèle})$ permet d'évaluer la distribution de probabilité sur A pour un modèle donné. Ainsi, en utilisant la règle de Bayes, on peut calculer $P(\textit{Modèle} \mid [A = a])$, c'est à dire, étant donnée une observation a de A, quel est le modèle qui explique le mieux $A = a$?

$$P(\textit{Modèle} \mid [A = a]) = \frac{P(\textit{Modèle})P([A = a] \mid \textit{Modèle})}{P([A = a])}$$

4.5.2 Diagnostic récursif avec comparaison de modèle

a) La certitude comme variable de sélection

La certitude, définie plus haut, peut être utilisée comme une variable de sélection de modèle : un modèle qui exprime l'observation à laquelle le système peut s'attendre étant donné qu'il est complètement sûr sa localisation, et un autre qui exprime l'observation attendue lorsqu'il est sûr d'être perdu.

Formellement, on définit $P(Z_t \mid \text{Cert}(t))$ de la manière suivante :

- Si $\text{Cert}(t) = 0$, puisque le système sait pertinemment qu'il ne sait rien sur sa localisation, $P(Z_t \mid [\text{Cert}(t) = 0])$ doit donc être une distribution qui exprime le minimum de connaissances *a priori* . En général, on choisit une distribution uniforme.
- Au contraire pour $\text{Cert}(t) = 1$, $P(Z_t \mid [\text{Cert}(t) = 1])$ est définie par marginalisation sur l'estimation courante de localisation $P(X)$:

$$P(Z_t \mid [\text{Cert}(t) = 1]) = \int_X P(Z_t \mid [X = x]\ [\text{Cert}(t) = 1])P([X = x])dx \qquad (4.23)$$

De cette manière, on peut utiliser la règle de Bayes pour calculer $P(\text{Cert}(t) \mid Z_t)$.

$$P(\text{Cert}(t) \mid Z_t) \propto P(\text{Cert}(t))P(Z_t \mid \text{Cert}(t))$$

b) Estimation récursive de confiance

L'estimation précédente de la certitude ne concerne qu'une observation à un instant donné et, de ce fait, ne tient pas compte de l'historique des observations. Elle ne peut donc pas être utilisée comme mécanisme d'estimation de la confiance en soi car celle-ci doit se construire au cours du temps en accumulant des succès ou des désillusions.

Si on souhaite considérer l'estimation de confiance comme une estimation d'état au cours du temps, l'outil qui vient naturellement à l'esprit est le filtre bayésien. Il nous faut donc un modèle d'observation et un modèle de transition :

– L'expression de $P(Z_t \mid \text{Cert}(t))$ donnée par l'équation 4.23 fournit un modèle d'observation.
– Sans perte de généralité, on peut définir le modèle de transition $P(\text{Cert}(t+dt) \mid \text{Cert}(t))$ par le tableau suivant :

$$P(\text{Cert}(t+dt) \mid \text{Cert}(t)) = \begin{array}{c|c|c|c} & & \text{Cert}(t+dt) & \\ & & 0 & 1 \\ \hline \text{Cert}(t) & 0 & 1-\lambda & \delta \\ \hline & 1 & \lambda & 1-\delta \end{array} \tag{4.24}$$

Les paramètres λ et δ seront nommés respectivement *taux de confiance* et *taux de doute* puisqu'ils expriment la quantité de probabilité passant respectivement de $P(\text{Cert} = 0)$ à $P(\text{Cert} = 1)$ et de $P(\text{Cert} = 1)$ à $P(\text{Cert} = 0)$, entre deux instants. En général il est tout à fait envisageable de choisir des valeurs différentes pour ces paramètres. Avec $\delta < \lambda$ on définit un système qui a du mal à prendre confiance en lui. Par rapport au cas $\delta = \lambda$, il faudra donc que les observations soient plus cohérentes avec le modèle – $P(Z_t \mid \text{Cert}(t))$ plus probable – pour que la confiance augmente.

Expression complète du filtre : En utilisant les modèles ci-dessus, on obtient un estimateur de certitude défini par l'expression :

$$P(\text{Cert}(t+dt) \mid \mathcal{Z}_{t+dt}) \propto P(Z_{t+dt} \mid \text{Cert}(t+dt)) \times \sum_{\text{Cert}(t)} P(\text{Cert}(t+dt) \mid \text{Cert}(t)) P(\text{Cert}(t) \mid \mathcal{Z}_t) \tag{4.25}$$

De l'expression ci-dessus, on déduit la confiance du système en lui-même : $\text{Confiance}(t) = P(\text{Cert}(t) = 1 \mid \mathcal{Z}_t)$.

c) Fonctionnement de l'estimateur de confiance

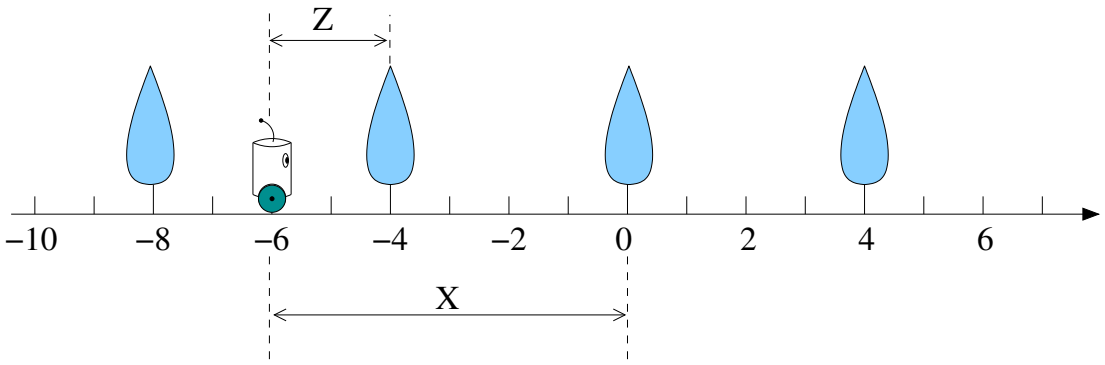


FIG. 4.39: Système unidimensionnel pour illustrer les mécanismes de prise de confiance.

*1.***Système considéré** Pour illustrer les mécanismes de prise de confiance que nous avons décrits plus haut, et surtout en illustrer les faiblesses, nous avons choisi d'utiliser de nouveau

un exemple trivial. Cependant, le système unidimensionnel et observable présenté dans la section 4.3.3 n'est pas suffisamment complexe pour servir d'illustration. Nous considérerons donc le système suivant : un robot unidimensionnel se déplaçant d'une unité par unité de temps, mais seulement capable d'observer la distance signée qui le sépare du multiple de 4 le plus proche. Ce système est illustré sur la figure 4.39.

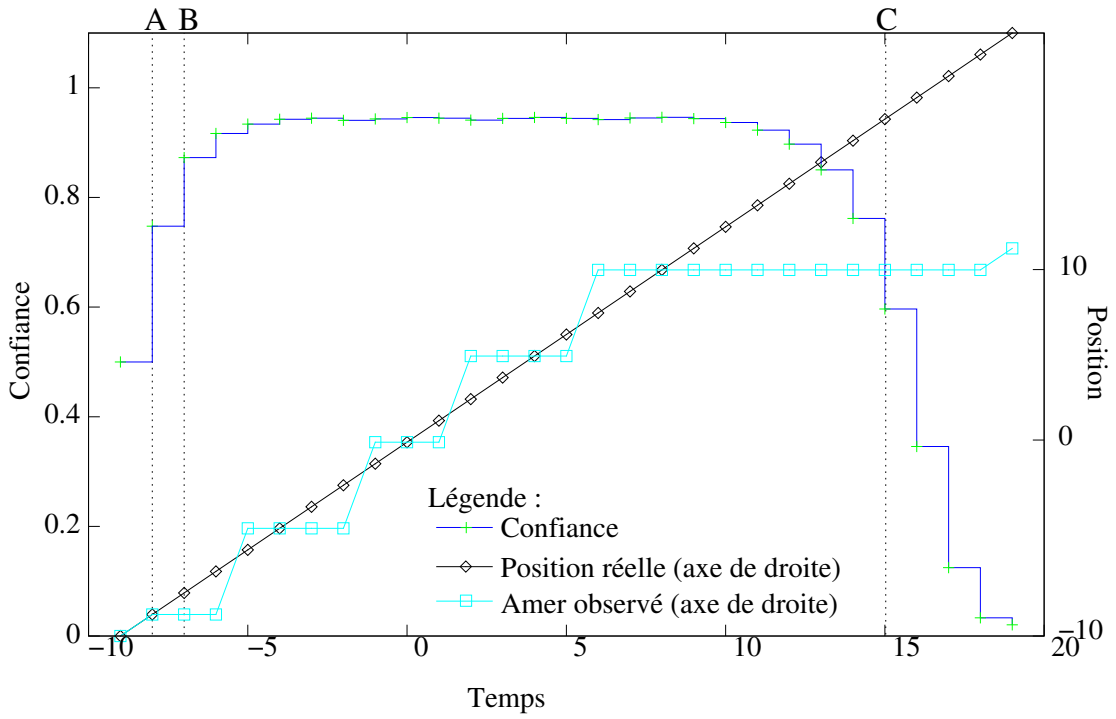


FIG. 4.40: Évolution de la confiance au cours du mouvement.

2.**Description de l'expérience** Le mouvement que nous allons considérer est lui aussi très simple : le robot commence en -9 avec une estimation de sa position gaussienne centrée sur -9. A chaque pas de temps il avance d'une unité, fait une observation et l'utilise pour mettre à jour son estimation de confiance et son estimation de position par l'intermédiaire d'un filtre de Kalman. Tout se passe bien jusqu'à la position 8 : les observations correspondent bien au multiple de 4 le plus proche de la position. A partir de l'observation 8, le capteur tombe en panne et retourne toujours la distance de la position courante à 8.

3.**Analyse des résultats** L'évolution du robot et de ses observations est illustré par la figure 4.40, en parallèle avec l'évolution de la confiance. On observe que tant que les données capteurs correspondent aux mesures la confiance augmente, puis celle-ci chute lorsque le capteur ne fonctionne plus.

Nous allons maintenant considérer trois instants particuliers de cette expérience, indiqués par les lettres A, B et C sur la figure 4.40.

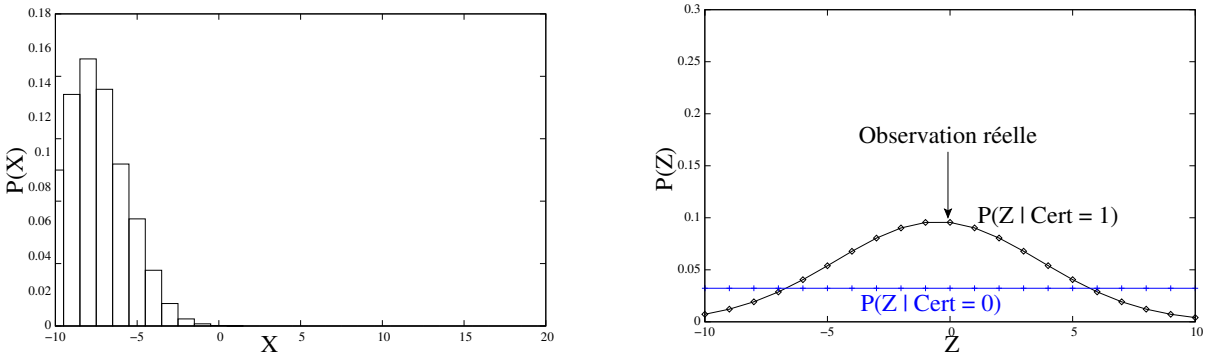


FIG. 4.41: Situation A : estimation de l'état (à gauche) et comparaison des mesures prédites avec la réalité.

Instant A : La figure 4.41 va nous permettre d'expliquer pourquoi la confiance augmente à l'instant A. Connaissant son estimation de position (partie gauche de la figure), le robot peut prévoir une distribution de probabilité sur les observations attendues $P(Z \mid [\text{Cert} = 1])$, sur la partie droite. Ensuite, au vu de l'observation réelle (flèche noire), il peut comparer la vraisemblance de l'observation réelle avec ou sans certitude. Dans ce cas $P([Z = z] \mid [\text{Cert} = 1])$ étant nettement supérieur à $P([Z = z] \mid [\text{Cert} = 0])$, le modèle de prise de confiance est favorisé et la confiance augmente.

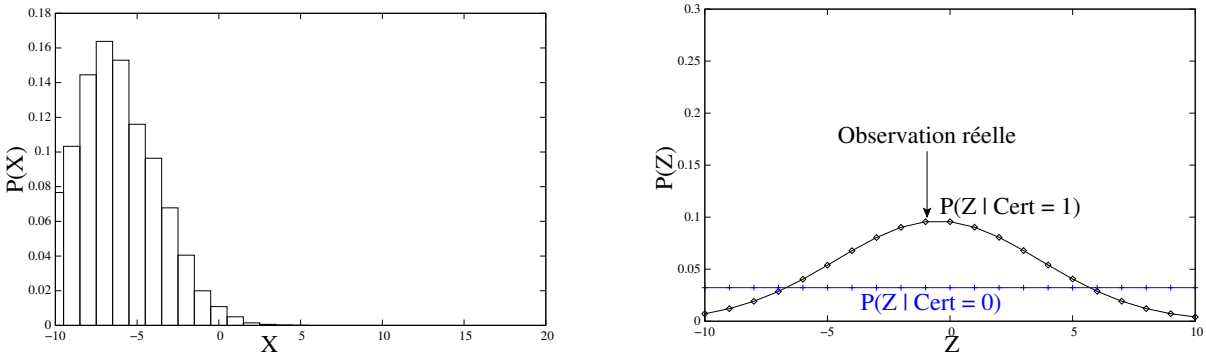


FIG. 4.42: Situation B : estimation de l'état (à gauche) et comparaison des mesures prédites avec la réalité.

Instant B : La situation B (fig. 4.42) met en évidence une faiblesse de la prise de confiance implantée avec seulement un mécanisme de comparaison de modèle. L'amer (le multiple de 4) observé est le même que dans la situation A. Ceci implique que le fait de prédire correctement cette observation n'apporte pas plus d'information sur la vraisemblance de la localisation que l'observation précédente, et donc que le robot n'a pas de raison d'avoir plus confiance en ses hypothèses de localisation après cette observation. Toutefois, avec un mécanisme de comparaison de modèle simple, l'évolution de la confiance n'est gouvernée que par la cohérence entre la prédiction et l'observation. Avec cette définition de la confiance, il est donc normal que le robot

prenne confiance en lui à cet instant, même si ce n'est ni justifié ni judicieux. Nous verrons par la suite un mécanisme capable de gérer la prise de confiance de manière plus judicieuse.

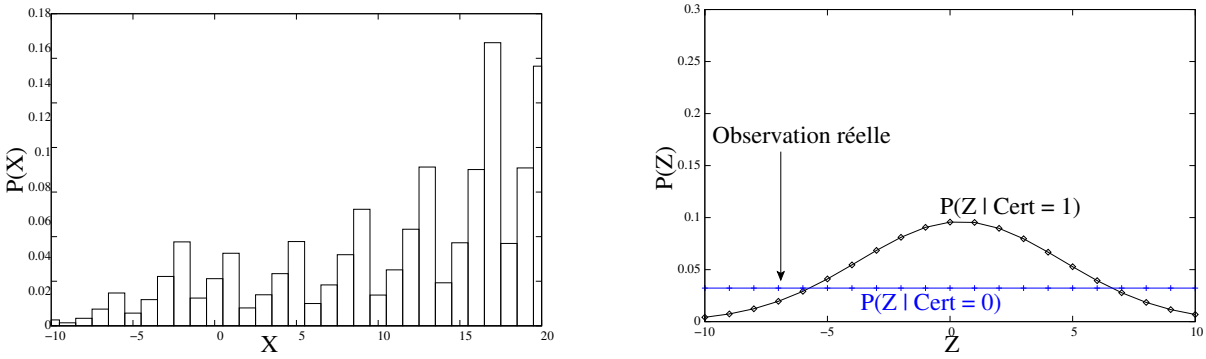


FIG. 4.43: Situation C : estimation de l'état (à gauche) et comparaison des mesures prédites avec la réalité.

Instant C : En réponse à la situation A, la situation C (fig. 4.43) permet de comprendre pourquoi la confiance décroît. A cet instant, l'observation réalisée est en désaccord complet avec l'observation prédite par le modèle de certitude. On a donc

$$P([Z = z] \mid [\text{Cert} = 1]) < P([Z = z] \mid [\text{Cert} = 0])$$

ce qui favorise le modèle de doute et, ainsi, fait diminuer la confiance.

4.5.3 Prise de confiance en fonction de l'innovation

L'estimation de confiance présentée dans la section précédente fonctionne de façon très satisfaisante dans l'ensemble. Cependant, on peut regretter un comportement qui apparaît lorsque le robot est immobile : si, à ce moment, l'observation est bien expliquée par le modèle "confiant", la confiance en soi converge rapidement vers 1. **Ce comportement n'est pas satisfaisant car la confiance ne devrait pas changer sans percevoir de nouvelle information**.

Comme nous sommes convaincus que la confiance en soi ne doit augmenter que lorsque le système est capable de prédire une observation qui ne résulte pas directement de l'observation précédente, nous avons décidé de modifier légèrement le modèle précédent de façon à implémenter le comportement désiré.

a) Modèle d'observation

Au lieu d'utiliser la certitude comme variable de sélection de modèle, on introduit une nouvelle variable de sélection : *Modèle* $\in \{0, 1, 2\}$. A partir de cette variable, on construit un nouveau modèle d'observation $P(Z(t) \mid \textit{Modèle})$ tel que :

- $P(Z(t) \mid [\textit{Modèle} = 0])$ est une distribution uniforme (équivalent à $P(Z \mid [\text{Cert}(t) = 0]$ dans le modèle précédent).

- $P(Z(t) \mid [\textit{Modèle} = 1]) = P(Z(t) \mid Z(t')\ U(t'))$ exprime l'observation attendue connaissant seulement l'observation précédente et le déplacement entre les deux observations.
- $P(Z(t) \mid [\textit{Modèle} = 2]) = P(Z(t) \mid L)$ exprime l'observation attendue avec toute l'information disponible, c'est à dire la localisation et un modèle de l'environnement (équivalent à $P(Z \mid [\text{Cert}(t) = 1]$ dans le modèle précédent).

A partir de ce modèle et d'un *a priori* uniforme $P(\textit{Modèle})$, on définit une distribution conjointe $P(Z(t)\ \textit{Modèle}) = P(\textit{Modèle})P(Z(t) \mid \textit{Modèle})$, dont on peut déduire $P(\textit{Modèle} \mid Z(t))$ en utilisant de nouveau la règle de Bayes.

b) Modèle d'évolution de la confiance

Pour tout $i \in \{0, 1, 2\}$, nommons

$$M_i = P([\textit{Modèle} = i] \mid Z(t)) \ \text{ et } \ Q_i = \log(\frac{M_i}{M_0})$$

Connaissant Q_1 et Q_2, nous pouvons prédire comment la confiance devrait évoluer, et en déduire un modèle probabiliste qui implémente ce comportement. On peut distinguer trois modes distincts dans ce dernier :

Mode 0 : Si $Q_2 < 0$, la prédiction d'observation connaissant la localisation et un modèle de l'environnement est pire qu'une prédiction sans connaissance *a priori*. De même que dans la section 4.5.1, la confiance du système en lui-même doit décroître. Plus Q_2 est éloigné de 0, plus la prédiction est mauvaise, et en conséquence, plus la perte de confiance doit être importante.

Mode 1 : Si $Q_1 \geq Q_2 \geq 0$, la prédiction d'observation connaissant uniquement l'observation précédente et le mouvement est meilleure qu'avec une connaissance complète. Cela signifie que l'observation courante ne reflète aucune innovation par rapport à l'observation précédente. Comme il n'y a pas de raison de changer la confiance sans nouvelles preuves, celle-ci doit rester constante.

Mode 2 : Si $Q_2 > Q_1$ et $Q_2 \geq 0$, l'observation courante est non seulement mieux prédite par le modèle résultant d'une connaissance complète, mais elle traduit aussi une certaine innovation par rapport à l'observation précédente. Cette capacité à prédire une observation inattendue doit augmenter la confiance en soi. De plus, une grande différence entre Q_2 et Q_1 implique que l'observation courante était vraiment difficile à prédire connaissant uniquement l'observation précédente. Donc la prise de confiance devrait être d'autant plus grande que la différence $Q_2 - Q_1$ est importante.

Pour faire le choix entre ces modes, surtout lorsque les valeurs sont proches, on introduit une variable $M \in \{0, 1, 2\}$ qui correspond au mode et on définit une relation $P(Q_1\ Q_2 \mid M)$ entre M et le couple $(Q1, Q2)$, de façon à obtenir les distributions présentées dans la figure 4.44.

Pour implanter ce comportement, on utilise le programme bayésien défini dans la figure 4.45. Ce programme est fondé sur une définition de $P(\text{Cert}(t) \mid \text{Cert}(t')MQ_1Q_2)$ similaire à l'équation 4.24, à la différence que le *taux de doute* δ et le *taux de confiance* λ sont maintenant des fonctions

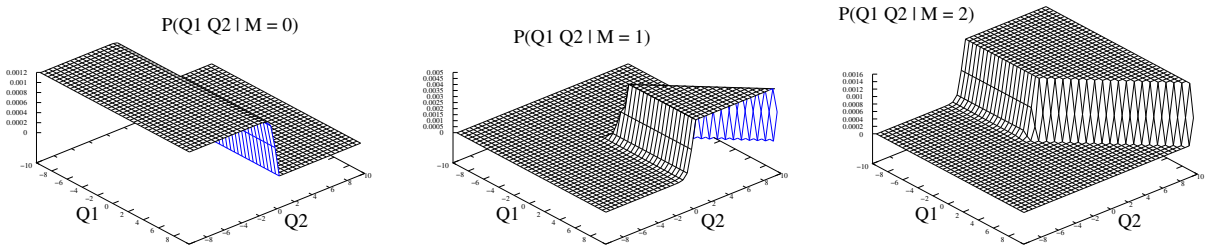


FIG. 4.44: Choix du mode M en fonction des valeurs de Q1 et Q2.

de M, Q_1 et Q_2.

$$\delta(Q_1, Q_2, M) = \begin{cases} 0 & \text{si : } M \neq 0 \\ -k_\delta.Q_2 & \text{sinon} \end{cases} \quad (4.26)$$

$$\lambda(Q_1, Q_2, M) = \begin{cases} 0 & \text{si : } M \neq 2 \\ k_\lambda.(Q_2 - Q_1) & \text{sinon} \end{cases} \quad (4.27)$$

(4.28)

Programme
- Description
  - Spécification
    - Variables Pertinentes
      - Cert(t) : Certitude à l'instant t
      - Cert(t') : Certitude à l'instant t'
      - M : Mode d'évolution de la confiance
      - Q_1, Q_2 : Observations, comme définies dans 4.5.3
    - Décomposition
      - $P(\text{Cert}(t)\ \text{Cert}(t')\ M\ Q_1\ Q_2) =$
        $P(Q_1\ Q_2)\ P(\text{Cert}(t'))P(M \mid Q_1\ Q_2)P(\text{Cert}(t) \mid \text{Cert}(t')M\ Q_1\ Q_2)$
    - Formes Paramétriques
      - $P(Q_1\ Q_2)$: Non définie
      - $P(\text{Cert}(t'))$: Résultat de l'itération précédente
      - $P(\text{Cert}(t) \mid \text{Cert}(t')\ M\ Q_1\ Q_2)$: Voir sec. 4.5.3
      - $P(M \mid Q_1\ Q_2)$: *idem*
  - Identification :
    - k_δ et k_λ sont ajustés de manière empirique.
- Question :
  - $P(\text{Cert}(t) \mid Q_1\ Q_2)$

FIG. 4.45: Programme bayésien pour l'estimation récursive de confiance en soi

c) Fonctionnement de l'estimateur de confiance

Nous allons maintenant reprendre le système unidimensionnel utilisé plus haut pour illustrer la prise de confiance par comparaison de modèle et appliquer notre estimateur de confiance sur la même expérience. La figure 4.46 rappelle le mouvement du robot et les amers observés en parallèle avec la nouvelle estimation de confiance. De manière qualitative, on voit immédiatement que

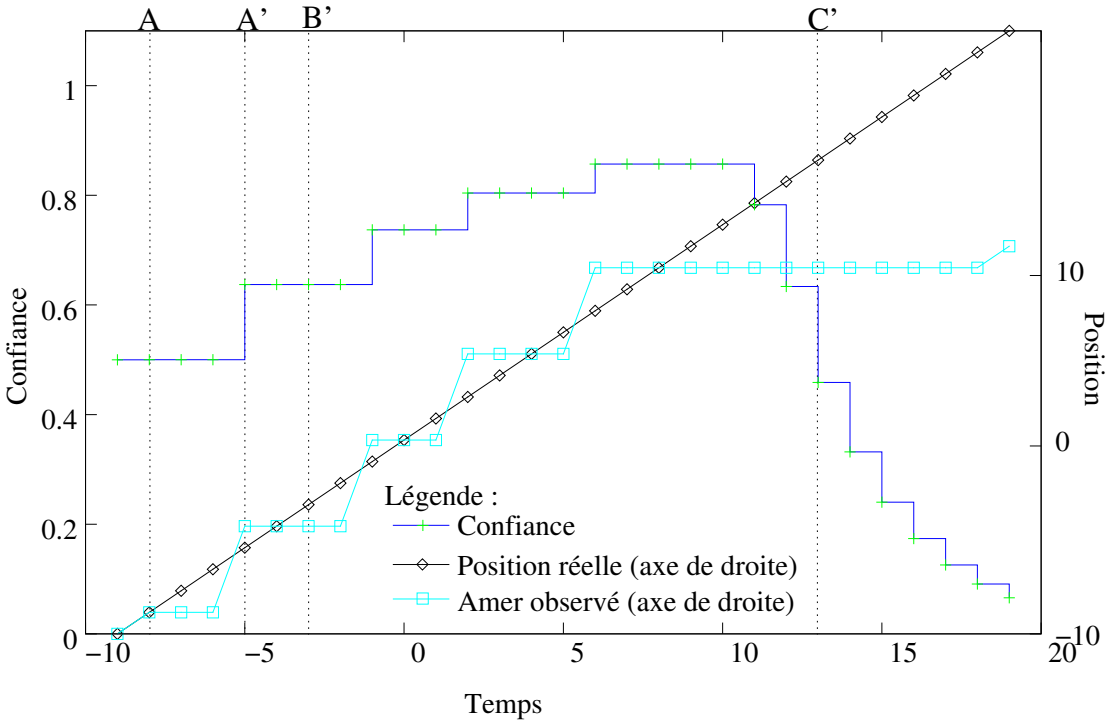


FIG. 4.46: Évolution de la confiance au cours du mouvement.

la confiance augmente beaucoup moins vite, et surtout qu'elle n'augmente que lorsqu'un nouvel amer est observé. On obtient donc le comportement désiré.

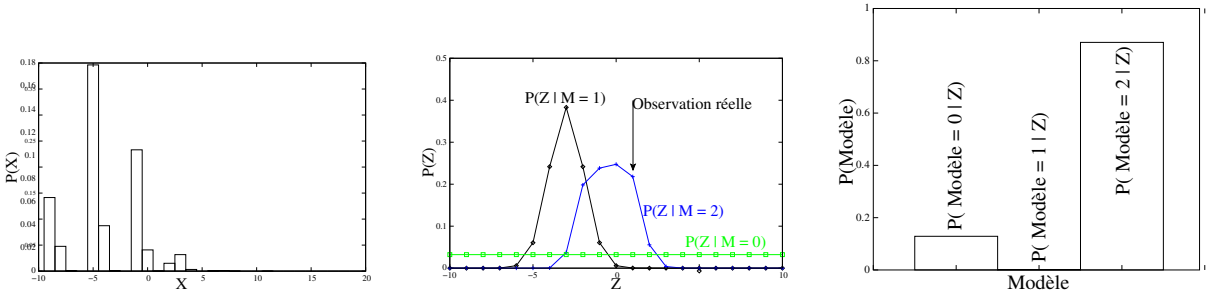


FIG. 4.47: Situation A' : estimation de l'état (à gauche), comparaison des mesures prédites avec la réalité (milieu) et probabilités des modèles (à droite).

Comme précédemment, nous allons nous intéresser à trois instants particuliers de cette trajectoire, indiqués par les lettres A', B' et C' sur la figure 4.46. L'instant A est ajouté pour comparaison. Il correspond à l'instant A de la figure 4.40.

Instant A' : Dans la situation A' (fig. 4.47), le robot possède une connaissance tri-modale de sa position. A partir de cette connaissance, il peut construire une distribution de probabilité sur les observations attendues. Comme pour la comparaison de modèles simples, cette distribution

est comparée à la distribution uniforme (graphe du milieu). Cependant, elle est aussi comparée à la prédiction qui peut être faite connaissant l'observation précédente et le déplacement du robot. Pour cette situation, l'observation correspond à un nouvel amer : la prévision à partir de l'amer précédent ne correspond donc pas du tout à cette observation. Ainsi, lorsqu'on calcule la distribution de probabilité sur le modèle le plus vraisemblable, le modèle 2 domine nettement les autres, ce qui justifie l'augmentation de la confiance.

Par rapport à la méthode précédente, nous pouvons remarquer que la première augmentation de la confiance se produit beaucoup plus tard (instant A, dans la figure 4.40). Il faut en effet attendre la première observation d'un nouvel amer, c'est à dire l'instant A'.

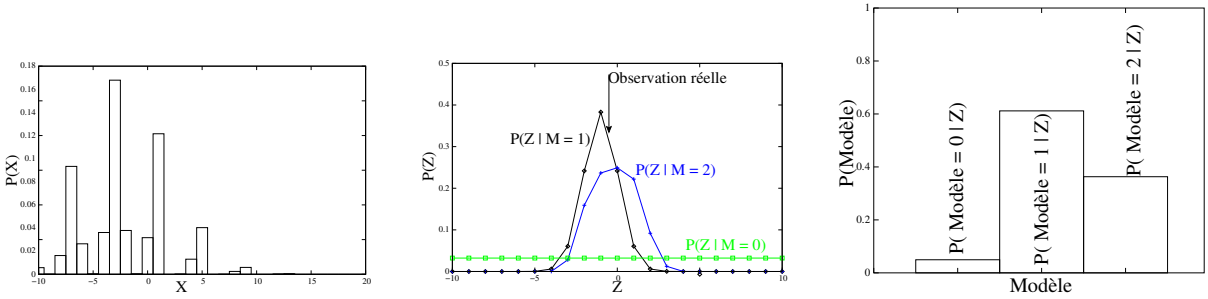


FIG. 4.48: Situation B' : estimation de l'état (à gauche), comparaison des mesures prédites avec la réalité (milieu) et probabilités des modèles (à droite).

Instant B' : Au contraire de la situation A', la situation B' (fig. 4.48) présente une situation où la confiance n'augmente pas. Ceci est dû principalement à la bonne prédiction de l'observation courante à partir de l'observation précédente. Le modèle 1 est donc le plus vraisemblable, ce qui provoque la stabilité de la confiance. Notons toutefois que cette stabilité n'est possible que parce que l'observation courante est tout de même bien prédite par le modèle 2.

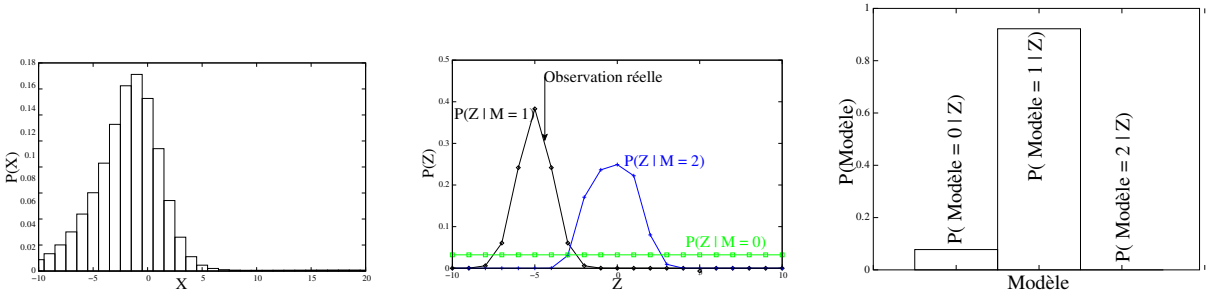


FIG. 4.49: Situation C' : estimation de l'état (à gauche), comparaison des mesures prédites avec la réalité (milieu) et probabilités des modèles (à droite).

Instant C' : Finalement, dans la situation C' (fig. 4.49), le robot fait une observation tout à fait prévisible à partir de l'observation précédente, mais complètement incompatible avec la prévision utilisant le modèle de capteur (modèle 2). Cette situation est caractéristique d'une

incapacité du système à prédire le monde qui l'entoure et il est normal qu'il perde alors confiance en lui.

4.5.4 Discussion

Dans cette section, nous avons présenté les principes d'un filtre bayésien pour l'estimation de confiance d'un système autonome. L'application de ces principes n'ont été illustrés ici que sur un cas d'école. L'implantation d'un mécanisme de prise de confiance sur notre robot réel, le CyCab, sera présentée dans le chapitre 7 sur l'intégration logicielle.

Le message fondamental de cette section est la définition de la confiance d'un système autonome comme sa capacité à utiliser ses connaissances sur sa localisation et sur son environnement pour prévoir ses observations.

Cependant, ce mécanisme d'estimation de la confiance pose encore quelques questions qui ne seront pas traitées dans cette thèse. Tout d'abord, dès que l'on met en place une procédure de diagnostic en ligne sur un robot mobile, un ensemble de questions se pose naturellement : doit-on ajouter un système de diagnostic du diagnostic ? Si oui, un diagnostic du diagnostic du diagnostic est-il lui aussi nécessaire ? Comment évaluer le gain en robustesse apporté par le diagnostic et comment évaluer la robustesse du diagnostic lui-même ?

On se pose ensuite la question de l'utilisation du diagnostic : comment utiliser intelligemment le diagnostic ? Faut-il replanifier une trajectoire ou juste ralentir ? Faut-il l'utiliser dans le cadre d'une interaction avec un opérateur humain ?

Relation entre confiance et diagnostic Les variables de diagnostic et la certitude étant des variables booléennes utilisées pour analyser des relations entre d'autres variables, on peut se demander si la similarité entre ces deux concepts n'est pas plus profonde. Tout d'abord, il est clair que si une relation existe, alors la variable de certitude est un cas particulier de variable de diagnostic.

Théoriquement, on peut considérer que la certitude est une variable de diagnostic qui évalue la cohérence entre une observation Z et un modèle *Modèle*. Cependant, pour spécifier le modèle de diagnostic correspondant, il faudrait exprimer $P(\text{Cert} \mid Z\,\textit{Modèle})$. Malheureusement, si nous avons dû définir les stratégies de comparaison de modèles présentées plus haut, c'est parce qu'il nous paraît impossible d'exprimer directement cette distribution.

Par ailleurs, l'essence de la notion de confiance en soi, telle que nous l'avons définie, est l'accumulation d'information dans le temps. Au contraire les modèles de diagnostic ne contiennent aucune référence au temps.

Au vu de ces quelques remarques, il nous semblerait abusif de vouloir rassembler les notions de confiance et de diagnostic. Ces notions utilisent effectivement des outils similaires, toutefois, elles sont conceptuellement et sémantiquement bien différentes.

4.6 Conclusion

Dans ce chapitre, nous avons cherché à répondre à la question de la localisation d'un robot mobile pour la réalisation d'une tâche de navigation. Notre réponse se décompose en trois volets : la localisation par rapport à une carte d'amers, la localisation par rapport à une trajectoire sensorimotrice et l'estimation de la qualité de la localisation par un indicateur de confiance.

La localisation par rapport à une carte d'amers est un problème dont la principale difficulté est l'identification d'un ensemble d'observations indistinguables. Pour optimiser le processus de mise en correspondance par graphes de correspondances, nous proposons d'identifier les triangles formés par des triplés d'amers. Cette identification résolue, il est alors aisé de calculer une estimation aux moindres carrés de la position du robot. Cependant, le processus déterministe d'identification par graphes de correspondances n'étant pas capable de gérer de façon satisfaisante les situations d'ambiguïté perceptive, nous avons montré comment décrire le problème de localisation sous forme d'inférence bayésienne sans réaliser une identification explicite des observations. Dans ce contexte, nous avons explicité l'intérêt d'utiliser des modèles capteurs exprimer sous forme de modèle de diagnostic pour améliorer la robustesse de la localisation aux fausses observations.

Nous avons vu ensuite comment exprimer et résoudre le problème de la localisation sur une trajectoire sensorimotrice. Cette section nous a permis de présenter une expression alternative de la localisation, sans carte, uniquement fondée sur la mémorisation d'une séquence sensorimotrice.

Finalement, nous avons terminé par un aspect essentiel, et pourtant peu abordé, de la localisation : l'analyse de la qualité de la localisation au cours du mouvement. L'approche choisie pour faire ce diagnostic mesure la cohérence entre les observations réalisées par les capteurs et la représentation interne de l'espace. Cette prise de recul par rapport aux algorithmes de localisation nous paraît indispensable et essentielle : aucune méthode de localisation ne peut être infaillible car tout algorithme est construit sur la base d'un modèle et donc d'une simplification de la réalité.

Les prochains chapitres vont nous permettre de traiter deux questions laissées en suspens dans ce chapitre : d'une part, la construction autonome d'une carte des amers présents dans l'environnement au cours du mouvement (chap. 5), et d'autre part, la génération effective de commande pour déplacer le robot vers l'objectif de sa navigation (chap. 6).

Chapitre 5

Construction d'une carte

5.1 Introduction

Dans le chapitre sur la localisation, nous avons supposé l'existence d'une carte, ou plus généralement d'un modèle de l'environnement. Ce modèle était alors utilisé comme référence par nos algorithmes de localisation.

Le problème qui va nous préoccuper dans ce chapitre est la construction autonome d'une carte de l'environnement par un robot mobile. La principale source de complexité de ce problème provient de la nécessité de construire la carte pendant l'exploration de l'environnement, en utilisant cette carte imparfaitement connue pour la localisation.

De nombreux travaux sur la construction de carte ont déjà été réalisés et publiés. Nous allons donc commencer ce chapitre par un tour d'horizon des principales méthodes existantes : grilles d'occupation, cartes stochastiques et cartes d'invariants.

Nous mettrons ensuite en évidence la forte adéquation entre les cartes d'invariants et l'algorithme de mise en correspondance par graphe de correspondances (cf. section 4.2), et nous montrerons comment implanter efficacement un algorithme combinant graphes de correspondances et construction d'une carte d'invariants.

5.2 Étude bibliographique

Avant de commencer cette étude, nous devons d'abord introduire deux notions antagonistes :

Une carte dense d'un espace E est une fonction qui associe à tout point de E une information sur ce point. Intuitivement une carte de randonnée, indiquant l'altitude et la végétation en tout point, est une carte dense. Ce type de carte est très riche en informations, mais en contrepartie, très coûteux en terme de volume de stockage. En général, en robotique, les cartes denses sont utilisées dans les algorithmes de planification de trajectoire, et plus rarement dans les algorithmes de localisation.

Une carte d'amers sur un espace E n'est une description que d'une sous-partie de E, généralement de mesure nulle et présentant un intérêt particulier : les amers. Intuitivement, une

carte routière est une carte d'amers, car elle ne décrit que les villes et les routes, mais ne dit rien sur l'espace situé en dehors de ces amers. Ce type de carte est très synthétique, peu volumineux, mais ne fournit qu'une vue partielle de l'environnement. Une carte d'amers est généralement utilisée et spécialisée pour la localisation.

Ces deux types de cartes étant complètement différents, les méthodes permettant leur estimation au cours du mouvement sont, comme nous allons le voir, complètement différentes.

Toutefois, il ne serait sans doute pas judicieux de considérer une des représentations comme meilleure que l'autre *a priori*. Dans le cadre d'une application complète, prévoyant à la fois de planifier des trajectoires et de se localiser avec précision, il peut être intéressant d'utiliser en parallèle ces deux représentations. C'est ce que nous avons choisi d'implanter sur notre plate-forme expérimentale. Les résultats correspondants seront présentés dans le chapitre 7.

5.2.1 Estimation d'une carte dense : les grilles d'occupation

Depuis une quinzaine d'année, la *grille d'occupation*[1] présentée par A. Elfes *[Elfes 1989]* est un outil très utilisé pour la construction de carte en robotique mobile.

L'objectif d'une grille d'occupation est l'estimation de l'état d'occupation de chaque point de l'espace de travail. Elle fournit une carte métrique, en général allocentrée, de l'environnement du robot.

*1.***Définition** La grille d'occupation est un découpage de l'espace dans lequel évolue le robot mobile (typiquement 2-D ou 3-D) en un ensemble de *cellules*. On cherche à estimer la probabilité qu'un *objet* quelconque (mur, humain ...) occupe chacune de ces cellules, à partir de mesures fournies par un capteur. Cette estimation se fait de manière incrémentale, en considérant une à une les différentes observations.

*2.***Hypothèse d'indépendance des cellules** La principale hypothèse des grilles d'occupation est que l'état de chaque cellule est supposé indépendant de l'état des autres cellules, connaissant la séquences des observations déjà réalisées. Ainsi la complexité de l'estimation de la grille complète est "cassée" : grâce à cette propriété, la probabilité d'occupation de chaque cellule peut être estimée indépendamment de celle des cellules voisines.

Estimer la grille complète revient à appliquer N fois l'estimation d'une seule cellule, N étant le nombre de cellules composant l'environnement complet du robot.

Si cette hypothèse est nécessaire pour assurer une estimation de la grille d'occupation en un temps raisonnable, elle peut entraîner un manque de précision des cartes obtenues.

*3.***Utilisation des grilles d'occupation** La figure 5.1 montre un exemple typique d'utilisation des grilles d'occupation : la modélisation en deux dimensions d'environnements statiques et structurés *[Thrun 1998b]*. Cette modélisation est allocentrée, puisqu'elle se fait dans un repère extérieur au robot. Sur cette figure, obtenue sur simulateur, la hauteur d'un point correspond à la

[1] *Occupancy Grid* en anglais.

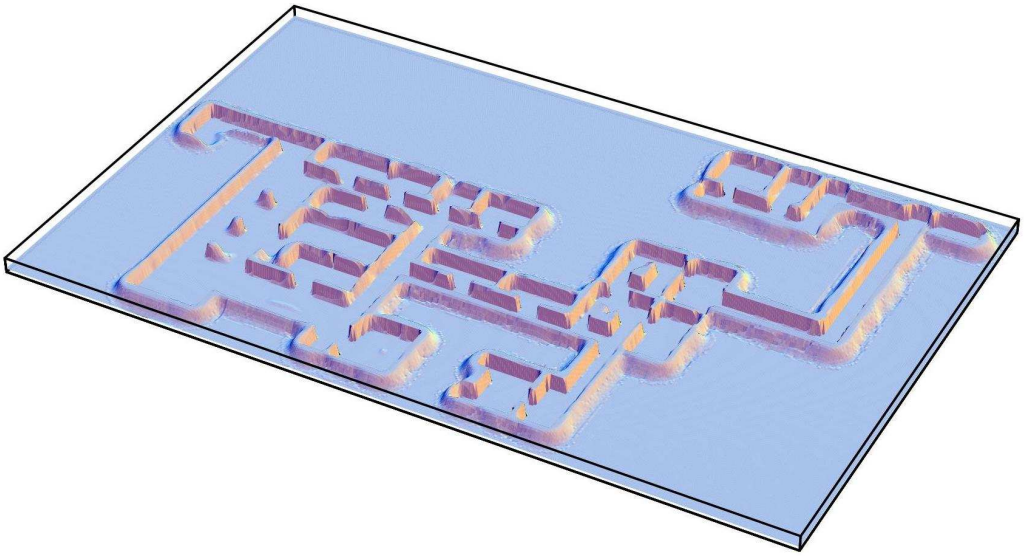


FIG. 5.1: Exemple de grille d'occupation obtenue sur simulateur *[Thrun 1998b]*. La hauteur d'un point représente la probabilité d'occupation de la cellule. Dans cet exemple, l'environnement du robot mesure 105 mètres par 63.

probabilité d'occupation de la cellule correspondante. Pour construire l'ensemble de la carte, le robot a parcouru son environnement pendant 45 minutes, la grille étant mise à jour à chaque observation des capteurs. Pour obtenir une telle précision, le simulateur considérait une odométrie, et donc une localisation du robot, parfaite. Les imprécisions d'un capteur de type sonar étaient quant à elles prises en compte. Les résultats obtenus avec un vrai robot sont, bien entendu, moins précis, compte tenu des erreurs de localisation.

À partir de cette représentation, la localisation du robot dans son environnement, la planification et l'exécution de trajectoire peuvent être réalisées. Les grilles d'occupation ont ainsi été à la base de visites guidées d'un musée, un robot jouant le rôle du guide *[Buhmann et al. 1995, Thrun et al. 2000]*.

Une des grandes qualités des grilles d'occupation provient de leur estimation incrémentale. Il est donc facile de tenir compte d'une succession d'observations capteur pour construire la grille. Cette qualité peut être utilisée pour filtrer les objets dynamiques d'une série d'observations capteur *[Yamauchi & Langley 1997]*. Dans cet exemple, le but de l'application était la localisation dans un environnement comportant des objets mobiles. La carte de l'environnement statique avait été préalablement construite.

*4.***Bilan** Les grilles d'occupation fournissent une carte dense de l'environnement non résumée aux amers, et donc très bien adaptée à la planification de trajet. Cependant, pour être efficace et précise, cette carte nécessite un système de localisation précis. Le principal défaut des représentations par grilles d'occupation est leur coût calculatoire dans le cas général, et dans une moindre mesure, leur précision limitée à la résolution de la grille.

*5.***Construction de carte en utilisant la logique floue** Les grilles d'occupations reposent sur une modélisation probabiliste du problème de construction de carte. De façon alternative, il est possible d'utiliser des techniques reposant sur la logique floue pour obtenir une sémantique et des résultats similaires *[Oriolo et al. 1997; 1998]*.

5.2.2 Estimation d'une carte d'amers

Contrairement aux grilles d'occupation, les techniques de construction d'une carte d'amers ne cherchent pas à estimer l'état complet de l'environnement, mais juste celui des amers.

On peut soit estimer directement l'état des amers, c'est le principe des cartes stochastiques, soit estimer des propriétés dont on pourra déduire cet état, c'est le principe des cartes d'invariants.

a) Carte stochastique et filtre de Kalman

La difficulté de la construction autonome d'une représentation de l'environnement provient de la nécessité de maintenir à la fois une estimation de la position du robot et une estimation de la carte. C'est le problème, désormais classique, du SLAM (Simultaneous Localization And Mapping *[Dissanayake et al. 2001, Leonard & Durrant-Whyte 1991]*) ou CLM (Concurrent Localization and Mapping *[Newman 1999]*).

*1.***Définition** Pour résoudre le problème du SLAM, l'approche classique consiste à construire une carte stochastique[2]. Une carte stochastique est une variable probabiliste réunissant l'état de tous les amers, ce qui implique que cet état doit pouvoir être décrit par une variable réelle. Dans le contexte du SLAM, puisqu'il est aussi nécessaire d'estimer la position du robot, cet état est généralement inclus dans la variable stochastique.

Pour illustrer les dernière avancées dans le domaine du SLAM, on peut citer, en particulier, les travaux de Durrant-Whyte *[Dissanayake et al. 2001, Newman 1999, Williams et al. 2003]*, de Nebot *[Guivant et al. 2002, Guivant & Nebot 2001; 2002]* ou de Thrun *[Montemerlo et al. 2002, Thrun 2002]*.

*2.***Estimation d'une carte stochastique** L'outil idéal pour estimer l'état d'une variable gaussienne observée au cours du temps est le filtre de Kalman.

Étant donné un modèle d'observation et un modèle de transition (cf. section 2.3.3), le filtre de Kalman estime l'état de la carte le plus probable au vu de toutes les observations réalisées.

Dans le contexte du SLAM, le modèle de transition est assez immédiat : pour la partie de la variable correspondant à l'état du robot, on utilise le modèle cinématique de ce dernier, et pour les amers, on les suppose immobiles. Le modèle d'observation peut, quant à lui, être plus compliqué, car il est construit à partir du modèle de capteur et des positions estimées du robot et des amers.

[2]Une variable stochastique est une variable probabiliste réelle, éventuellement multidimensionnelle, associée à une distribution de probabilité gaussienne.

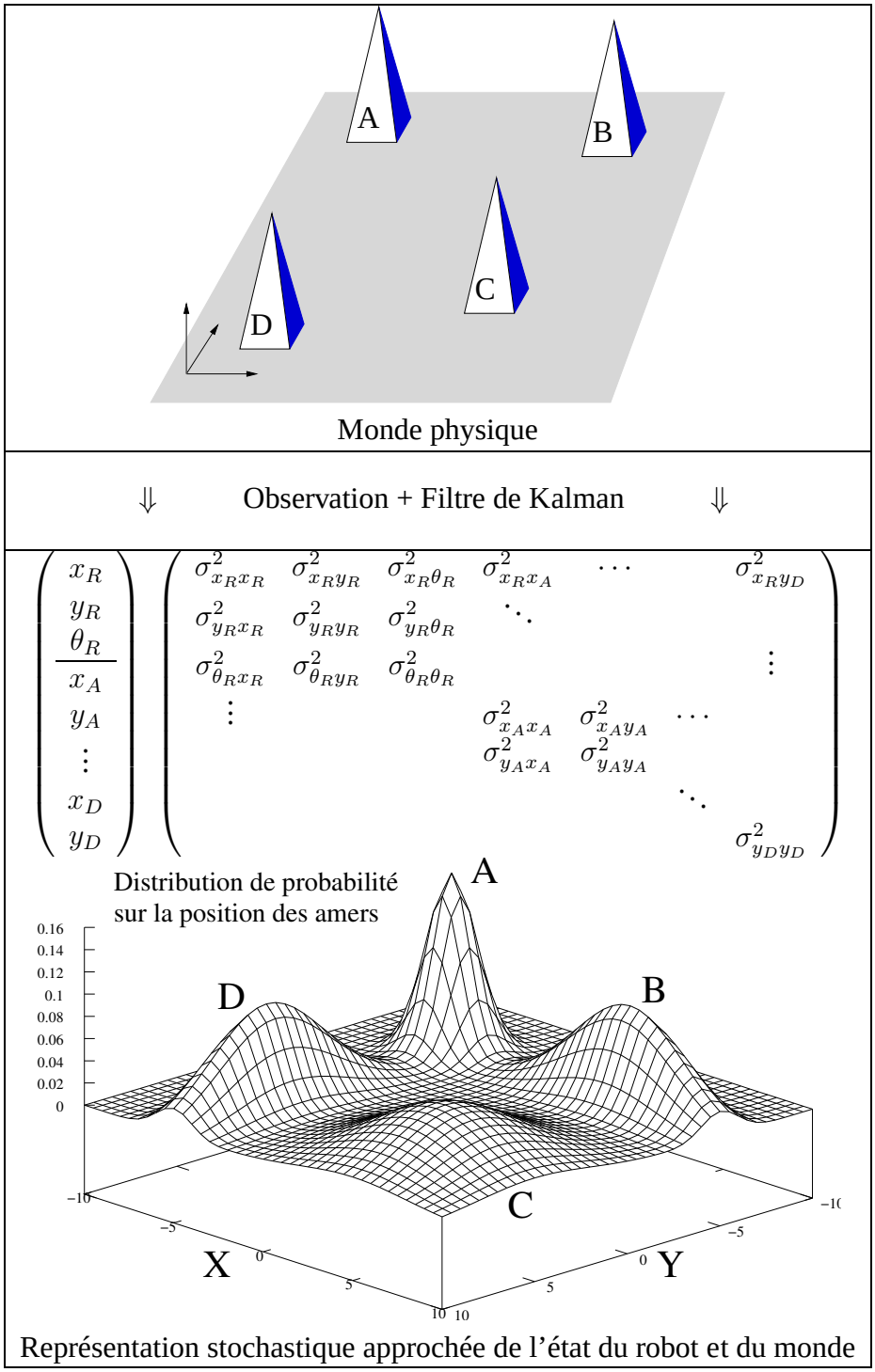


FIG. 5.2: Principe de base du SLAM en utilisant une carte stochastique.

*3.***Illustration de l'algorithme d'estimation** La figure 5.2 illustre le principe du SLAM en utilisant une carte stochastique : en évoluant dans le monde physique, le robot collecte des informations, à la fois sur son mouvement (odométrie) et sur les amers (pyramides en haut de la figure 5.2).

Ces informations sont ensuite fusionnées dans un filtre de Kalman (si les modèles de transition ou d'observation ne sont pas linéaires, on utilise un filtre de Kalman étendu et des modèles linéarisés) afin d'obtenir une représentation stochastique conjointe de l'environnement et de l'état du robot. Cette gaussienne est caractérisée par une moyenne donnant l'état conjoint le plus probable et une matrice de covariance, représentant l'incertitude associée à cette connaissance de l'environnement.

Le bas de la figure 5.2 donne une représentation intuitive approchée de l'information contenue dans la carte stochastique sur l'état de l'environnement : il s'agit de la densité de probabilité de la position d'un amer. On y trouve quatre modes qui correspondent aux quatre amers présents dans l'environnement : la position de l'amer "A" est connue beaucoup plus précisément que celle de l'amer "C", et pour les amers "B" et "D", le filtre n'a pu estimer qu'une seule dimension avec précision.

*4.***Bilan** La plupart des articles sur le SLAM cités plus haut utilisent la notion de carte stochastique et un mécanisme de mise à jour inspiré du filtre de Kalman. Ces algorithmes présentent l'avantage de générer une estimation précise de la carte des amers.

Le principal défaut de cette approche est son coût calculatoire. En effet, dans la formulation du filtre de Kalman, il est nécessaire d'inverser une matrice de mêmes dimensions que la matrice de covariance de l'état. Si on nomme n la dimension de la carte stochastique, la complexité *a priori* de cette inversion est $O(n^3)$. Or, pour un système évoluant dans un environnement relativement grand, le nombre d'amers peut rapidement devenir important, et ainsi rendre le coût de l'inversion incompatible avec une application temps-réel.

Pour contrer ce défaut des cartes stochastiques, des moyens d'alléger le coût calculatoire de la mise à jour ont été proposés, en particulier par l'équipe de Nebot *[Guivant et al. 2002, Guivant & Nebot 2001; 2002]*. Grossièrement, il s'agit de maintenir à la fois une estimation de l'environnement global et de l'environnement local. L'environnement local est mis à jour avec chaque observation. Cette mise à jour est peu coûteuse car la taille de cet environnement est petite et bornée. Cette estimation locale est ensuite insérée dans l'estimation globale lorsqu'elle contient suffisamment d'information. Cette opération est coûteuse mais n'est réalisée que de temps en temps.

Par ailleurs, la description de l'état de la carte par une matrice de covariance est à l'origine d'une spécificité des cartes stochastiques : cette matrice traduit l'existence de liens de dépendance probabiliste – ou corrélations –, non seulement entre l'état du robot et la carte, mais aussi entre les amers eux-mêmes. L'intérêt de ces corrélations est qu'elles permettent de transmettre l'information d'un bout à l'autre de la carte. En effet, lorsqu'on observe un amer, le filtre de Kalman affine non seulement sa connaissance sur l'état de cet amer, mais aussi, grâce aux corrélations, sur l'état global de la carte et même sur l'état du robot.

Ces liens de dépendance sont à la fois précieux et dangereux. Ils sont précieux quand ils per-

mettent de répartir dans toute la carte la connaissance acquise par une observation. Ils sont aussi dangereux car ils vont diffuser dans toute la carte l'erreur de mise à jour due à une observation mal interprétée, et ainsi transformer une erreur locale en incohérence globale de la carte.

b) Carte d'invariants géométriques

Comme nous venons de le voir, la construction d'une carte d'amers en utilisant une carte stochastique présente deux spécificités qui la rendent indésirable : son coût calculatoire et son manque de robustesse aux observations incorrectes ou incorrectement identifiées. Nous allons maintenant considérer une approche de construction de carte motivée par ces problèmes et particulièrement intéressante par la représentation originale de l'espace qu'elle implique.

*1.***Le filtre relatif** En 1997, M. Csorba (*[Csorba & Durant-Whyte 1997a]*) a présenté un nouvel algorithme de SLAM : le *Filtre Relatif*[3]. Au lieu de construire une carte absolue corrélée avec la position du robot, il propose de construire une carte de primitives indépendantes de la position du robot : les primitives invariantes. De cette manière, la variable aléatoire décrivant la carte est indépendante de celle décrivant l'état du robot.

Alors que l'algorithme classique du SLAM recherche les corrélations, le filtre relatif les évite à tout prix en s'assurant que les variables aléatoires correspondant aux primitives invariantes sont indépendantes. Cette indépendance est obtenue en ne mettant à jour que l'état d'une partie des primitives invariantes après chaque observation. La thèse de Csorba *[Csorba & Durant-Whyte 1997b]* et l'article de Deans *[Deans & Hebert 2000]* donnent plus de détails sur cette spécificité du filtre relatif.

*2.***Le filtre à projection géométrique (FPG)** En 1999, P. Newman *[Newman 1999]* a étendu ce filtre pour obtenir le Filtre à Projection Géométrique[4] (FPG) qui fournit un moyen de reconstruire une carte d'amers cohérente à partir des primitives relatives issues du filtre relatif de Csorba. Finalement, M. Deans *[Deans & Hebert 2000]* à fait faire un pas de plus au FPG en en développant une version non linéaire. En résumé, le FPG fournit un filtre efficace et précis malgré les approximations qu'il implique, et légers en terme de complexité calculatoire.

*3.***Fonctionnement du FPG** En réponse à la figure 5.2, la figure 5.3 illustre le fonctionnement du filtre à projection géométrique.

Contrairement à l'algorithme classique du SLAM où l'état des amers était estimé directement, seules les distances entre les amers sont estimées par le filtre relatif, car ce sont les primitives invariantes pour ce système.

A partir des observations, le filtre relatif estime, grâce à des filtres de Kalman unidimensionnels, les longueurs des segments observables dans l'environnement. Ces estimations vont donc être connues avec une précision quantifiée (fig. 5.3, milieu).

[3]Relative Filter.

[4]Geometric Projection Filter.

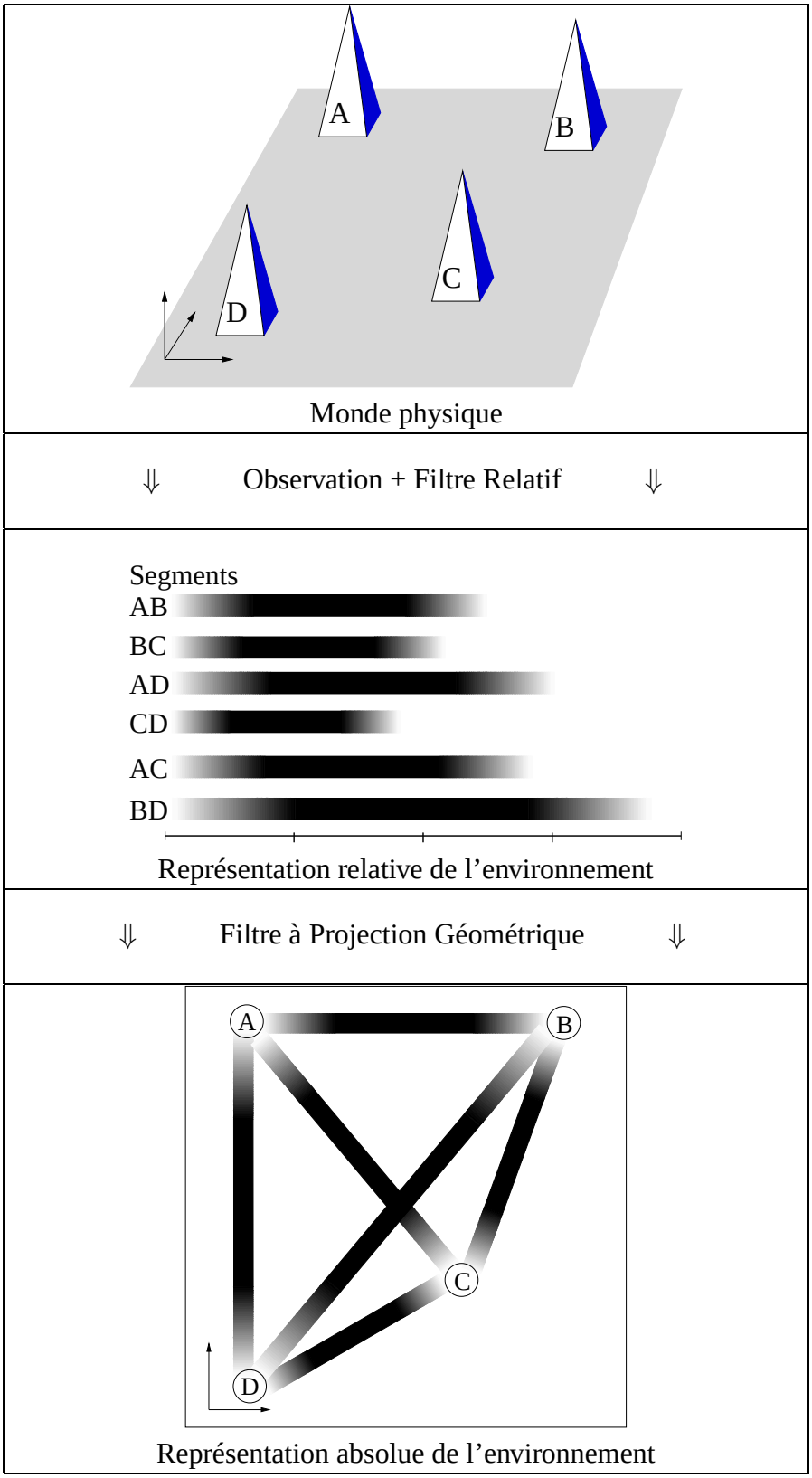


FIG. 5.3: Principe de base de l'estimation de carte par FPG.

Pour obtenir une représentation plus lisible, et utilisable pour la localisation absolue du robot, l'ensemble de segments estimés par le filtre relatif doit être converti en un ensemble d'amers, c'est le rôle de la partie *projection géométrique* du FPG. En utilisant des techniques d'optimisation multidimensionnelle (Levenberg-Marquardt par exemple), on peut construire un ensemble d'amers dont les inter-distances sont aussi proches que possible de celles estimées (fig. 5.3, bas). Pour un ensemble d'amers $\mathcal{L} = \{L_i, i = 1 \dots n\}$, on veut trouver

$$\mathcal{L}^\star = \underset{\mathcal{L}}{\operatorname{argmin}} \sum_{i,j} (d(L_i, L_j) - \hat{d}(L_i, L_j))^2$$

où $d(L_i, L_j)$ est la distance entre L_i et L_j et $\hat{d}(L_i, L_j)$ est la distance estimée.

Puisque les segments estimés par le filtre relatif ne dépendent pas de la position du robot, il existe une transformation planaire libre entre la carte absolue résultant de l'optimisation et la réalité. Afin de déterminer uniquement ces paramètres libres, nous avons choisi de considérer que le premier amer ajouté à la carte absolue l'est à sa position réelle. Ceci fixe la translation libre. Pour la rotation libre, nous la déterminons en fixant la direction entre le premier et le second amer.

*4.***Bilan** Le Filtre à Projection Géométrique est un algorithme efficace et robuste d'estimation d'une carte d'amers. Ces deux propriétés sont dues principalement à l'absence de corrélations dans la carte. En contrepartie, l'information y circule moins bien, et chaque observation n'apporte qu'une information locale qui ne se propage pas dans le reste de la carte. Pour obtenir un même niveau de précision, il faudra donc faire plus d'observations avec le FPG qu'avec une carte stochastique et un filtre de Kalman.

Par ailleurs, l'intérêt conceptuel de ce filtre est la représentation de l'espace original qu'il construit. Au lieu de référencer tous les objets par rapport à un point de référence unique et arbitraire, il ne retient de l'espace que les relations entre les objets.

5.3 Intégration de la mise en correspondance : le FPGMCI

Les algorithmes d'estimation d'une carte d'amers présentés dans notre étude bibliographique ont tous les deux une hypothèse fondatrice en commun : les observations qu'ils traitent sont identifiées. Ainsi, il est facile de comparer l'observation et l'estimation d'un amer et d'en déduire la mise à jour à effectuer.

Toutefois, comme nous l'avons vu en section 4.2, l'identification des observations est un réel problème, relativement coûteux en termes de ressources calculatoires.

Notre objectif, dans cette section, est de montrer comment intégrer de façon harmonieuse et efficace le processus d'identification des observations dans l'algorithme d'estimation de carte.

5.3.1 Choix des algorithmes

Si on met en parallèle l'algorithme de mise en correspondance par graphe de correspondances, présenté en section 4.2, et les algorithmes d'estimation de carte présentés plus haut, on

constate une très forte adéquation entre le Filtre à Projection Géométrique et la technique des graphes de correspondances. En effet, les propriétés invariantes sont non seulement à la base de notre méthode de mise en correspondance, mais aussi à la base du FPG.

Dans la suite, nous allons voir comment, avec des structures de données adaptées, la mise en correspondance et la construction de carte d'invariants pourront être fusionnées en un seul processus cohérent que nous appellerons le Filtre à Projection Géométrique avec Mise en Correspondance Intégrée (FPGMCI).

Notons que si nous avions choisi d'utiliser un algorithme de SLAM fondé sur une carte stochastique, il aurait alors été plus judicieux d'utiliser une mise en correspondance de type JCDA, qui, dans son implantation usuelle, est particulièrement bien adaptée à l'utilisation de données issues de cartes stochastiques *[Christensen 2002, Leonard & Durrant-Whyte 1991, Neira & Tardos 2001]*.

5.3.2 Principe du FPGMCI

Pour implanter le FPG tel que présenté dans la section 5.2.2 et dans *[Deans & Hebert 2000]*, nous avons besoin d'estimer et de mettre à jour deux structures de données principales : la base de données (ou carte) relative (BDR), c'est a dire la liste des primitives invariantes, et la carte absolue des amers (CAA), qui n'est autre que l'estimation des positions des amers dans un repère absolu. Après chaque observation, certaines primitives de la BDR seront mises à jour, et de temps en temps, la carte absolue est mise à jour en utilisant la BDR.

Comme nous l'avons vu, la mise en correspondance par graphe de correspondances utilise une base de données de primitives invariantes dans laquelle les primitives observées sont recherchées. Dans la suite de cette section, nous verrons que la structure construite par le FPG peut être utilisée directement comme base de données pour la mise en correspondance.

*1.***Cartes relatives** A la différence de *[Deans & Hebert 2000]*, nous avons choisi d'estimer deux bases de données relatives : une pour stocker les paires d'amers et les distances les séparant, et une pour stocker les triplés d'amers et les triangles qu'ils forment. La figure 5.5 permet de mieux se représenter ces structures.

*2.***Construction de la carte** Quand un nouvel amer est observé, sa position dans le repère absolu est estimée à partir de la position estimée du robot. L'amer est ensuite inséré à cette position dans la carte absolue. Il est seulement nécessaire de l'insérer dans un voisinage de sa "position réelle" car le processus d'optimisation du FPG ajustera sa position plus tard. Ensuite, les primitives invariantes mettant en jeu cet amer sont calculées et ajoutées à la BDR. En pratique, on n'ajoute que les segments et triangles correspondant à des amers qui ne sont pas trop éloignés les uns des autres, car les amers trop éloignés ont peu de chances d'être observés simultanément.

*3.***Salle d'attente** Dans un environnement réel, même avec un capteur presque parfait, il peut arriver que des amers inexistants soient détectés. Afin d'éviter d'insérer ces amers fantômes dans la carte absolue, nous avons instancié un mécanisme de salle d'attente. En pratique, nous utilisons

deux bases de données spécifiques, de structure identique aux cartes relatives et absolues pour stocker les amers dont nous ne sommes pas encore sûr de l'existence. Une des bases de données servira à stocker la carte absolue des amers hypothétiques (CAAH) et l'autre servira à stocker les primitives invariantes impliquant au moins un amer hypothétique, nous l'appellerons BDRH pour *Base de Donnée Relative Hypothétique*.

Quand une observation est faite, on essaye d'abord de la mettre en correspondance avec les amers non-hypothétiques. Si cela échoue, on essaye de la mettre en correspondance avec les amers hypothétiques. Si cela échoue aussi, alors l'observation est peut-être un nouvel amer et on l'ajoute donc à la CAAH en mettant à jour la BDRH. Dans le cas contraire, l'amer hypothétique gagne un point de validation. Les amers hypothétiques ayant atteint un score suffisant sont validés et entrent dans la CAA. Ceux qui restent trop longtemps dans la CAAH sont éliminés. Pour les primitives invariantes, elles sont éliminées dès qu'un des amers dont elles dépendent est éliminé, et elles passent de la BDRH à la BDR dès que tous leurs amers sont validés.

*4.***Mise en correspondance et estimation de carte simultanées** Depuis le début de ce chapitre, nous avons pu constaté que les données nécessaires à la mise en correspondance et à la construction de carte sont très similaires. De plus, la mise en correspondance par la méthode des *graphes de correspondances* est un processus très flexible qui peut bien s'adapter au mécanisme de *salle d'attente*. Ainsi, au moment où l'on met une observation en correspondance avec la CAA, on construit un graphe de correspondances G dont on extrait une clique maximum C_m. L'observation est alors séparée en un ensemble d'amers identifiés ($\in C_m$) et non identifiés. Ceux qui ne sont pas identifiés sont alors mis en correspondance avec la BDRH en ajoutant des arêtes (de même que dans la section 4.2.3) à G et en recherchant dans G la clique maximum C_{em} qui contient C_m. Cela signifie que nous voulons un ensemble d'amers hypothétiques identifiés non seulement maximum, mais aussi cohérent avec l'ensemble d'observations identifiées aux amers validés.

L'algorithme ainsi obtenu, le FPGMCI, est résumé dans la table 5.1. La figure 5.4 illustre les relations entre les structures de données et les différents algorithmes mis en jeu. Elle met aussi en évidence le rôle central de la carte de primitives invariantes, utilisée à la fois pour la mise en correspondance et la construction de la carte des amers.

5.3.3 Implantation

Maintenant que nous avons le principe général de l'algorithme FPGMCI, il est réellement nécessaire de conduire une réflexion profonde sur le choix des structures de données ; le but étant d'obtenir un algorithme fonctionnant en temps réel sur notre robot mobile, avec des mises à jour aussi fréquentes que possible.

a) Structures de données

Pendant le fonctionnement de l'algorithme, nous devons stocjer des informations sur différentes structures de données. Ceci est illustré par la figure 5.5. Même si ces structures ne sont

TAB. 5.1 – Filtre à Projection Géométrique avec Mise en Correspondance Intégrée (FPGMCI).

[MISE EN CORRESPONDANCE ET LOCALISATION]

1 Mettre les observations en correspondance avec la CAA (en utilisant les primitives invariantes de la BDR).
2 Si la mise en correspondance réussit, calculer la position du capteur à partir des observations identifiées. Dans le cas contraire, la position n'est estimée qu'avec l'odométrie.

[CONSTRUCTION DE CARTE]

3 Mettre à jour la BDR.
4 Étendre le graphe de correspondances en mettant en correspondance les observations non encore identifiées avec les amers hypothétiques.
5 Ajouter les amers toujours non identifiés à la salle d'attente. Ils deviennent donc des amers hypothétiques.
6 Ajouter un point de validation à tous les amers hypothétiques identifiés.
7 Valider les amers hypothétiques ayant accumulé suffisamment de points de validation.
8 Éliminer les amers hypothétiques ayant dépassé l'âge limite.

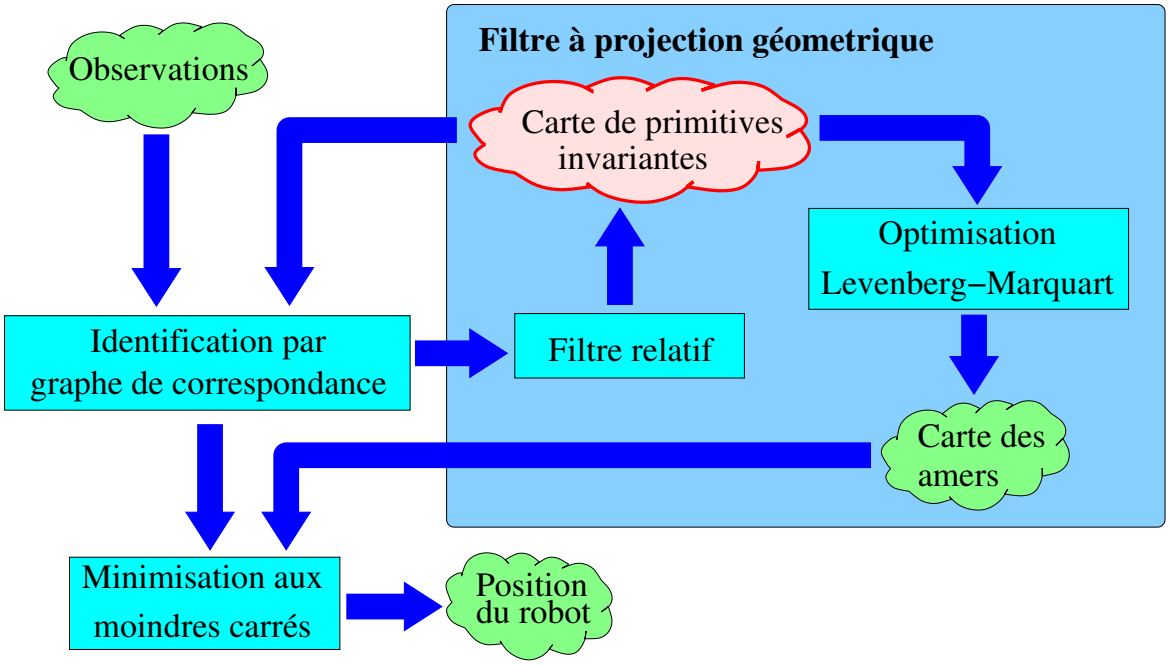


FIG. 5.4: Relations entre structures de données et algorithmes dans le FPGMCI.

pas extrêmement complexes, la façon de s'en servir sera déterminante pour l'efficacité de l'algorithme.

*1.***Carte absolue des amers (CAA)** Lorsqu'on met a jour la carte absolue et lorsqu'on ajoute de nouveaux amers à la BDR, on doit pouvoir faire une traversée linéaire de cette structure. Pendant la mise en correspondance, la construction et le traitement du graphe de correspondance seront grandement simplifiés avec un accès aléatoire aux amers. Par ailleurs, lorsque l'on valide ou que l'on élimine un amer, on a besoin d'ajouter et de supprimer des amers. Il faut noter cependant que ces derniers évènements sont plutôt rares, comparés aux autres opérations. Pour ces raisons, un tableau dynamique paraît être la structure la plus adaptée (la *classe vector* de la STL[5] en est un bon exemple).

*2.***Observation** De même que pour les amers, on a principalement besoin d'une traversée linéaire efficace pour cette structure. Cependant, l'accès aléatoire rend les choses plus faciles sans avoir d'influence notable sur la performance. Une structure de tableau dynamique paraît de nouveau la mieux adaptée.

*3.***Triangles et segments** Pendant la mise en correspondance, il sera nécessaire de répondre à de nombreuses requêtes de recherche (rechercher par exemple un segment dont la longueur est

[5]La *STL* est une bibliothèque de classes de base définies par la société *SGI*. Son principal intérêt est de fournir des algorithmes spécifiés de façon très rigoureuse avec des garanties de complexité temporelle. Par exemple, la classe *vector* permet de stocker des objets en mémoire et garantit que la complexité de l'accès à ces objets est $O(1)$

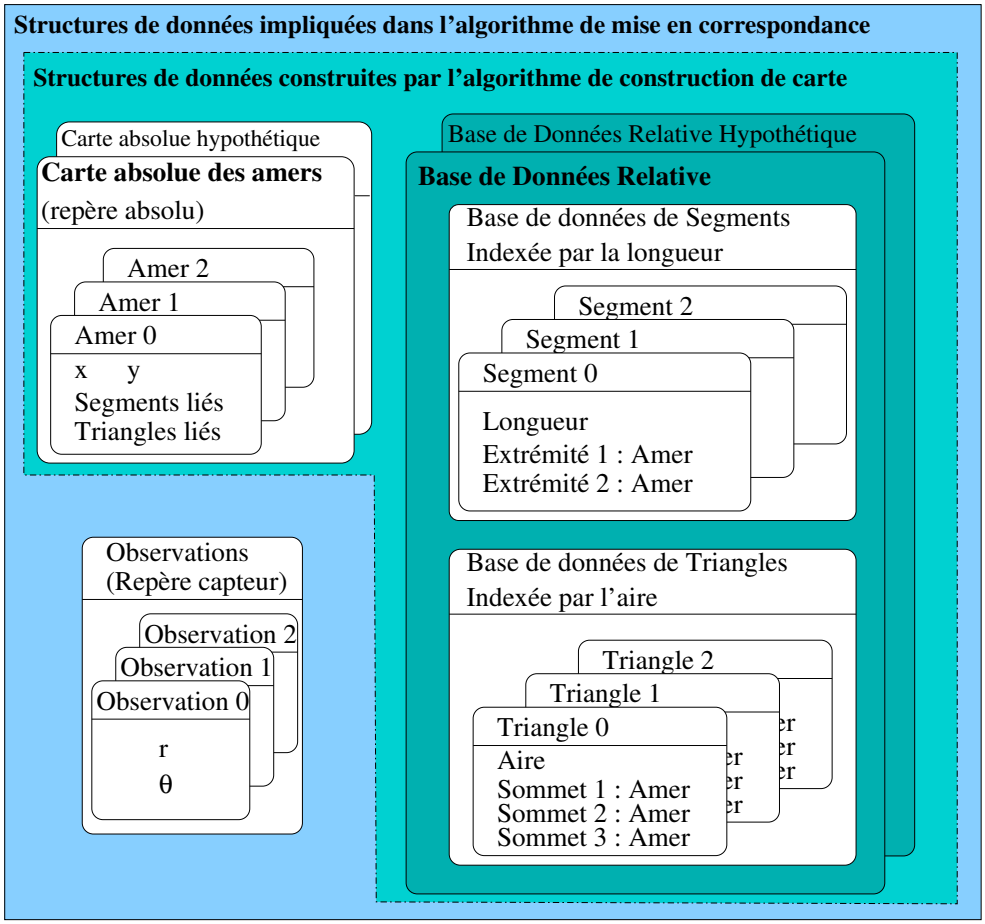


FIG. 5.5: Résumé des structures de données nécessaires à l'algorithme FPGMCI.

proche de celle d'un segment observé). Lorsqu'on valide ou que l'on élimine les amers hypothétiques, de nombreuses insertions et suppressions seront nécessaires. La meilleure façon de répondre à ces besoins est sans doute d'utiliser une arbre binaire de recherche équilibré (un arbre rouge/noir par exemple) indexé par les invariants (longueurs des segments et aires des triangles). Le *classe* STL *multimap* propose ces fonctionnalités.

*4.***Relations entre amers, segments et triangles** Chaque amer a besoin de connaître les segments et triangles dont il fait partie. Cette relation sera utilisée au moment de sa validation ou de sa suppression. Lorsqu'on valide un amer, chaque segment et chaque triangle dont il dépend doit savoir qu'il a un amer hypothétique de moins, et éventuellement passer de la BDRH à la BDR. Lorsqu'on supprime un amer, il faut détruire tous les segments et triangles qui en dépendent. On a donc besoin d'un parcours linéaire, d'une insertion et d'une suppression efficaces. Notre choix se porte donc sur une liste doublement chaînée qui propose ces services de façon tout à fait satisfaisante.

b) Complexité de l'algorithme

Nommons n_o le nombre d'observations réalisées, n_a et n_{ah} le nombre d'amers validés et hypothétiques, n_m le nombre d'identifications après l'étape 1 (voir table 5.1) et n_{em} le nombre d'identifications après l'étape 4.

TAB. 5.2 – Complexité des étapes du FPGMCI.

Étape	Complexité
1	Pour chaque triangle observé, le rechercher dans la BDR : $O(n_o^3 \log(n_l) + c(C_m))$.
2	Calculer la position (voir 4.2.2) : $O(n_m)$.
3	Trier les segments identifiés selon la covariance de l'estimation de leur longueur : $O(n_m^2 \log(n_m))$. Se reporter à *[Deans & Hebert 2000]* pour des détails.
4	Pour chaque triangle observé, le rechercher dans la base de données hypothétique : $O(n_o^3 \log(n_{hl}) + c(C_{em}))$.
5	Tester chaque observation : $O(n_o)$.
6	Tester chaque amer hypothétique : $O(n_{hl})$.
7	Pour chaque amer validé (αn_{hl} amer), construire les triangles et les segments correspondants : $O(\alpha n_{hl}(n_l + n_{hl})^2)$, avec $\alpha \ll 1$.
8	Pour chaque amer éliminé (βn_{hl} amers), détruire les segments et les triangles correspondants : $O(\beta n_{hl}(n_l + n_{hl})^2)$, avec $\beta \ll 1$.

Avec ces notations, on peut exprimer la complexité temporelle de chaque étape (cf. table 5.2) et obtenir ainsi la complexité globale suivante (sans tenir compte de l'optimisation de la CAA, réalisée de manière asynchrone) :

$$C = O(n_o^3 \log(n_l) + c(C_m) + c(C_{em})), \tag{5.1}$$

où $c(C_m)$ et $c(C_{em})$ sont les complexités de l'extraction de la clique maximum C_m et de son extension maximum C_{em}. La complexité *a priori* de ces dernières complexités est exponentielle (la recherche de clique maximum est un problème NP-complet). Cependant, lorsque suffisamment d'observations sont disponibles, l'utilisation de nos primitives invariantes dans la construction des arêtes du graphe donne des contraintes suffisamment fortes pour rendre le problème traitable en temps réel. En pratique, la recherche de la clique maximum dans notre graphe est faite en recherchant le plus gros sous-ensemble de noeuds ayant des degrés et un cardinal compatibles avec une clique (ce qui se résume à un tri des noeuds du graphe), puis en vérifiant qu'il s'agit bien d'une clique. De cette manière, on obtient la complexité au mieux :

$$\begin{aligned} c(C_m) &\approx O(n_o(n_l + n_{hl}) \log(n_o(n_l + n_{hl})) + n_m^2) \\ c(C_{em}) &\approx O(n_o(n_l + n_{hl}) \log(n_o(n_l + n_{hl})) + n_{em}^2 + n_{em} n_m) \end{aligned}$$

D'où, puisque $n_m + n_{em} < n_o$ et $n_{hl} \ll n_l$,

$$C = O(n_o^3 \log(n_l) + n_o n_l \log(n_o n_l)) \tag{5.2}$$

En pratique, n_o n'est jamais très grand : moins de dix observations sont présentes en même temps.

5.3.4 Résultats expérimentaux

Les résultats expérimentaux présentés dans cette section proviennent de données collectées avec notre véhicule autonome (cf. section 7.3) alors que celui-ci était conduit manuellement sur le parking de l'INRIA Rhône-Alpes. Lors de ces expérimentations, la vitesse du véhicule était limitée à $2m/s$ et le nombre d'amers dans l'environnement était limité à 11 (pour une raison matérielle de disponibilité et de coût des amers).

a) Mesure du temps moyen d'exécution

Le premier résultat à signaler concerne les performances en termes de rapidité d'exécution de l'algorithme. Durant nos tests sur un PC cadencé à 1.2 GHz, son temps moyen d'exécution était inférieur à 2 millisecondes (avec des pics à 50 millisecondes au moment de l'optimisation de la carte absolue).

Le protocole expérimental ayant permis de mesurer la valeur ci-dessus est le suivant : tout d'abord, on considère une séquence datée de données capteur enregistrée sur le robot pendant les phases de conduite manuelle. Cette séquence se compose d'un ensemble de mesure de déplacement (fréquence d'acquisition : $20Hz$) et d'un ensemble de mesures laser (fréquence d'acquisition : $8 \pm 2Hz$). Toutes ces mesures sont associées à une date d'acquisition, ce qui permet d'appliquer l'algorithme FPGMCI sur une station de bureau, mais dans les conditions d'une exécution réelle sur le processeur embarqué.

Ensuite, l'algorithme est instrumenté par un outil spécialisé dans l'estimation de temps d'exécution. Cela signifie qu'on ajoute à toutes les fonctions utilisées dans le programme une petite fonction chronomètre qui note les temps d'entrée et de sortie de la fonction. Évidemment,

cette fonction chronomètre, comme toute séquence d'instruction machine, représente un coûts en terme de temps d'exécution. Les temps d'exécution mesurés seront donc légèrement supérieurs à la réalité.

Finalement, l'algorithme FPGMCI a été appliqué sur une séquence de 1500 observations laser et 5000 mesures odométriques. Le processus d'optimisation de la carte absolue n'est mis en oeuvre qu'une fois toutes les 30 observations laser.

b) Trajectoire et carte obtenue

La figure 5.6 présente une carte absolue et une trajectoire estimées en temps réel au cours de la conduite manuelle. La seule information dont dispose le système à l'initialisation est le fait que la distance minimale séparant deux balises est supérieure à 50 centimètres. Lors de l'exécution de cette trajectoire, l' estimation de position était disponible à tout instant. Cette propriété sera particulièrement intéressante en association avec les algorithmes de suivi de trajectoire au chapitre 7.

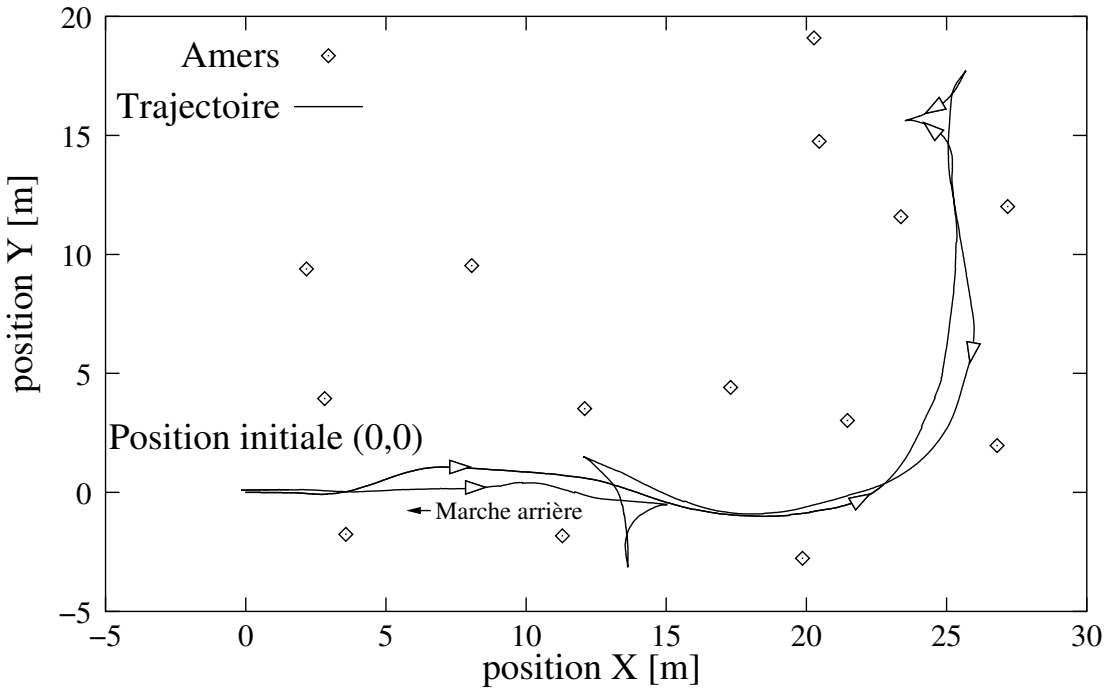


FIG. 5.6: Trajectoire et carte des amers calculées en temps réel.

c) Évaluation de la qualité de la localisation

Afin de valider notre module de localisation, il est nécessaire de le comparer à un autre module qui servira de référence. On pourra alors mesurer l'erreur entre la position renvoyée par notre module et celle affirmée par le module de référence. Malheureusement, comme souvent en robotique mobile, le module de localisation présenté dans cette section était destiné à être le

module de localisation le plus précis installé sur le robot et aucun module de référence n'était donc disponible.

*1.***Méthodologie** Deux solutions apparaissent alors : valider l'algorithme sur un système expérimental simulé ou déterminer un processus indirect de validation. Éliminons tout d'abord le système simulé : pour qu'un tel système représente bien la réalité, il faut mettre en oeuvre des techniques de calibration complexes (capteurs, actionneurs) dont il faut ensuite valider la calibration. Même si cette calibration est possible, il sera ensuite difficile de quantifier la part d'imprécision due à l'algorithme de celle due à la calibration. Le système simulé ne sera donc vraiment utile que pour la première validation : vérifier qu'avec des données parfaites, l'algorithme arrive à un résultat parfait. C'est heureusement le cas !

Finalement, pour valider le FPGMCI, nous avons choisi d'observer les conséquences de sa précision. Le système de localisation traite chaque donnée laser et en déduit une estimation de la position à laquelle la mesure a été réalisée. Or les données laser brutes forment une séquence relativement dense (2 points/degrés) de mesures de distance. Il est donc possible de construire une image sur laquelle seront superposées toutes les séquences de mesures laser, dessinées à partir de l'estimation de leur position d'acquisition. On obtient ainsi une image de l'environnement qui sera plus ou moins précise selon la précision de la localisation, on parle ici de précision en termes de finesse des contours. La figure 5.7 présente une telle figure.

*2.***Résultats** La qualité de notre module de localisation est révélée par trois aspects de la figure 5.7. Tout d'abord, la plupart des impacts laser sur les amers (numérotés de 1 à 14) sont concentrés dans un cercle de 30 centimètres de diamètre, alors que les amers sont des cylindres de 15 centimètres de diamètre et que la mesure de position des amers est estimée[6] à 0.5% de la distance (10 centimètres à 20 mètres). Il s'agit donc d'un très bon résultat. En second lieu, le véhicule marqué d'un "C" sur la figure est particulièrement intéressant. Trois ensembles d'impacts peuvent y être distingués, certains correspondant à des distances supérieures à 20 mètres. Le premier ensemble correspond à des observations faites depuis la position P_1, le deuxième correspond à un point de vue situé en P_2 et le troisième en P_3. Cependant, même si ces points de vues sont éloignés, non seulement dans l'espace, mais aussi dans le temps, les ensembles d'impacts correspondants sont cohérents et surtout alignés (la déformation qui apparaît dans l'ensemble des points vus depuis P_2 est due à la détection de l'aile arrière du véhicule par le capteur laser). Cela montre à la fois la stabilité et la précision de l'algorithme. Ces propriétés sont aussi illustrées par le troisième aspect de la figure : les impacts lasers sur le mur du bâtiment marqué d'un "B" sont, eux aussi, bien concentrés et très bien alignés (les décrochements visibles sur la figure correspondent à la forme réelle du batiment).

d) Robustesse de l'algorithme

Pour finir sur les performances de cet algorithme de localisation, voyons quelles sont ces limitations : tout d'abord, la localisation n'est possible que dans les espaces ou au moins deux

[6] Grâce à une campagne de calibration expérimentale.

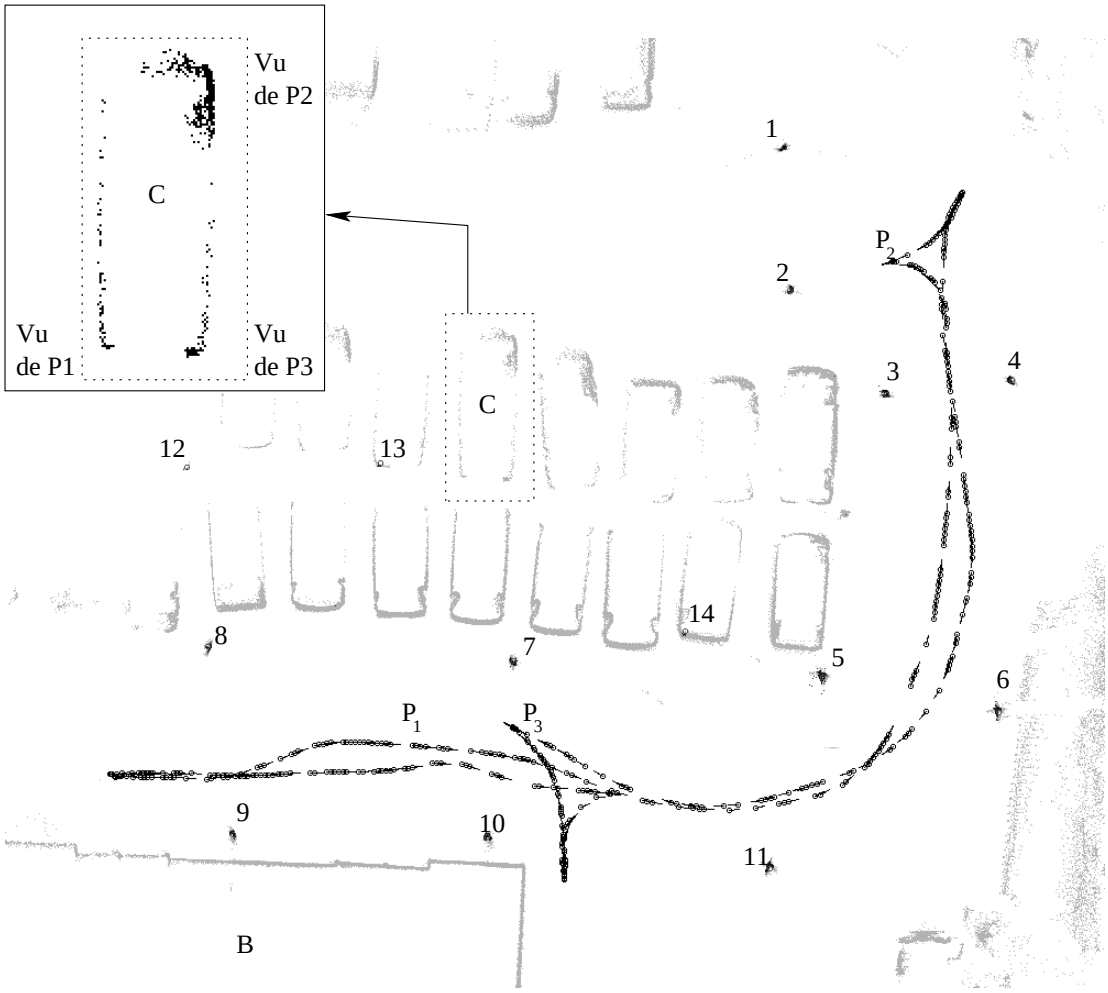


FIG. 5.7: Localisation : validation expérimentale.

amers sont visibles. De plus, pour que la construction de la carte des amers soit robuste et efficace, il faut observer toutes les arêtes de la triangulation de l'ensemble des amers. Cela signifie que si le système voit un couple d'amer (A, B) puis un couple (C, D), mais que le capteur ne peut jamais observer ni (A, C) ni (B, D), alors il pourra construire deux cartes locales (A, B) et (C, D) mais la relation entre ces cartes restera très vague[7]. En pratique, pour ne pas être exposé à ces problèmes, il faut que les amers soient placés judicieusement dans l'environnement. Nous nous sommes penchés sur ce problème dans *[Pradalier & Sekhavat 2002b]*, toutefois ces résultats sortent du cadre du présent travail.

Une seconde difficulté de cette algorithme vient du problème des trajectoires fermées[8]. Quand le robot se déplace sur un long périmètre puis observe de nouveau la partie de la carte qu'il a observée initialement, l'accumulation des erreurs fait qu'il est souvent difficile de savoir s'il s'agit d'un lieu inconnu ou de l'observation de la carte connue. De nombreux travaux ont été, et sont encore, réalisés pour résoudre ce problème *[Gutmann & Konolige 2000]*. Dans notre cas, ce problème n'a pas été observé dans notre environnement restreint tant que les contraintes précédentes étaient respectées. Cela ne signifie en aucun cas que l'algorithme présenté dans cette section est une solution parfaite au problème des trajectoires fermées. Il s'agit en effet d'un problème particulièrement complexe, et il est peu probable qu'il soit possible de le résoudre sans un traitement spécifique, ce qui n'a pas été fait dans le cadre de cette thèse.

Pour finir, signalons que pour la configuration d'amers et la trajectoire présentée dans la figure 5.7, l'identification des observations a *toujours* été possible.

5.4 Conclusion

Ce chapitre nous a permis de présenter notre contribution au problème de construction de carte et localisation simultanée (SLAM). Nous proposons de rassembler la mise en correspondance par graphe de correspondances et l'algorithme du filtre à projection géométrique, de façon à obtenir un fonctionnement efficace et robuste de l'algorithme conjoint : le Filtre à Projection Géométrique avec Mise en Correspondance Intégré (FPGMCI).

Cette intégration est parfaitement justifiée par la forte proximité des structures de données manipulées par les deux algorithmes.

Les résultats obtenus traduisent l'efficacité du FPGMCI, aussi bien en termes de précision de la localisation que d'efficacité calculatoire.

[7] C'est le principe d'une carte topologico-métrique.

[8] traduction approchée de "closing-the-loop problem", à ne pas confondre avec un système en boucle fermé "closed-loop system".

Chapitre 6

Suivi de trajectoire sécurisé

6.1 Introduction

Comme nous l'avons vu en introduction, la réalisation d'une tâche de navigation intentionnelle demande la mise en place d'une boucle localisation/contrôle, ou plus généralement d'une boucle sensorimotrice. Nous avons présenté notre vision de la partie sensorielle de cette boucle dans les chapitres sur la localisation (chap. 4) et la construction de carte (chap. 5). Nous allons maintenant nous intéresser à la partie motrice.

Dans le chapitre 3, nous avons vu deux types de représentation d'une tâche de navigation ; une représentation sous forme de comportement et une représentation sous forme de trajet. Lorsque la tâche est définie par un comportement, la partie motrice de la boucle est immédiate, par simple application du comportement. Au contraire, lorsqu'il s'agit d'un trajet (route, chemin, trajectoire), il est nécessaire de définir un moyen de **suivre ce trajet**.

Par ailleurs, lorsqu'un robot exécute une tâche de navigation autonome, la question de la **sécurité** se pose nécessairement. Le robot doit, en effet, garantir la sécurité des entités présentes dans son environnement (piétons, cyclistes, autres véhicules...) et faire attention à la sienne.

Les deux problématiques ci-dessus sont des questions classiques de la robotique autonome, et en conséquence, de nombreuses solutions ont déjà été proposées dans la littérature. Pour le suivi de trajectoire, on peut citer en particulier les travaux de Samson *[Artus et al. 2003, Samson & Ait-Abderrahim 1991]* et Kanayama *[Kanayama et al. 1990]* pour la proximité de leur contexte applicatif. Pour ce qui est de l'évitement d'obstacles, de nombreux travaux utilisent la notion de *champs de potentiel[Borenstein & Koren 1989; 1991, Khatib 1985, Ulrich & Borenstein 1998; 2000]* ou la méthode *Dynamic Window[Fox et al. 1997]*.

Les solutions que nous allons proposer dans ce chapitre ont été conçues dans l'optique d'une intégration dans une architecture de contrôle bayésienne (cf. chapitre 7). **Notre objectif est donc de montrer que l'on peut aussi exprimer les problèmes de suivi de trajectoire et d'évitement d'obstacles sous formes de problèmes d'inférence bayésienne et de mettre ainsi en évidence les possibilités sémantiques qui en découlent.**

6.2 Contexte applicatif

Le robot que nous considérons dans ce chapitre est un véhicule autonome de type CyCab (cf. fig. 6.1).

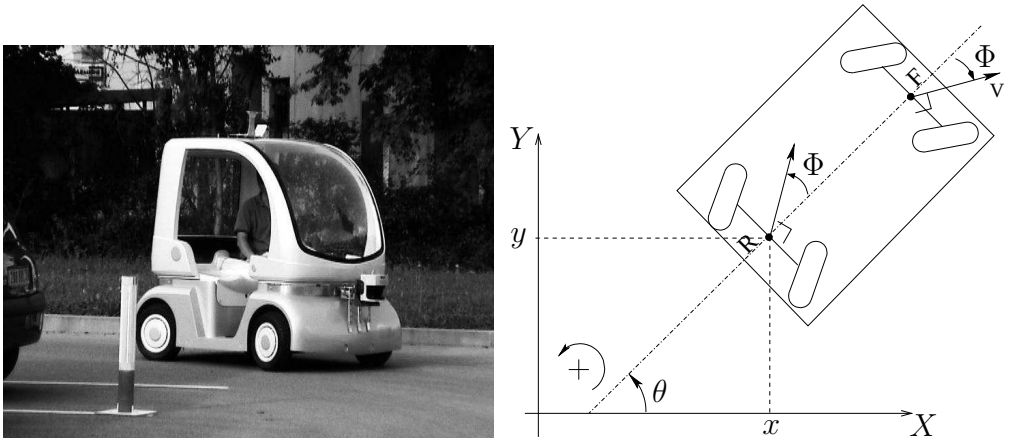


FIG. 6.1: Le CyCab et son modèle.

Nous considérerons que le CyCab évolue dans le plan (x, y). Il peut donc être décrit par une configuration tridimensionnelle : ses deux coordonnées de position x et y et sont orientation θ (cf. fig. 6.1).

Par ailleurs, pour piloter le robot, nous lui envoyons un couple d'ordre moteur (V, Φ). Φ est l'angle de braquage appliqué sur les deux essieux et V est la vitesse désirée pour le milieu de l'essieu avant (point F sur la figure 6.1).

6.3 Suivi de trajectoire

6.3.1 Spécification du problème

Définition 24 – Trajectoire *En premier lieu, nous définissons une trajectoire comme une fonction du temps dans le produit cartésien de l'espace des configurations et de l'espace moteur.*

$$\begin{array}{rccc} T: & \mathbb{T} & \longrightarrow & \mathbb{C} \times \mathbb{U} \\ & t & \longmapsto & (x, y, \theta, V, \Phi) \end{array}$$

Définition 25 – Erreur de suivi *A un instant t, la trajectoire T définie une configuration de référence $C_r(t)$ et commande de référence $U_r(t)$: $(C_r, U_r) = T(t)$. Au même instant, le robot réel est dans une configuration $C(t)$.*

L'erreur de suivi est défini comme la différence entre $C(t)$ et $C_r(t)$. On la note

$$\delta C(t) = C(t) - C_r(t) = (\delta X, \delta Y, \delta\theta)$$

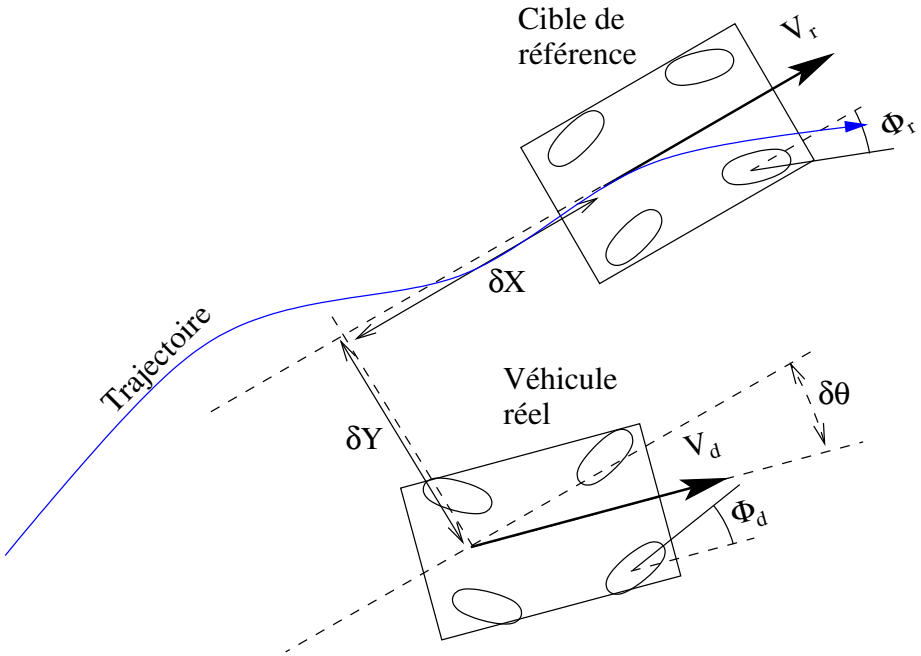


FIG. 6.2: Variables utilisées dans le suivi de trajectoire.

La figure 6.2 donne un aperçu graphique de ces variables.

L'objectif de cette section est de définir un programme bayésien pour calculer les commandes (V_d, Φ_d) à appliquer pour suivre la trajectoire, connaissant les commandes de référence (V_r, Φ_r), et l'erreur de suivi $(\delta X, \delta Y, \delta\theta)$.

Remarque : Pour pouvoir appliquer le suivi de trajectoire que nous allons définir, il n'est pas indispensable que la trajectoire soit définie dans l'espace des configurations. Il est juste nécessaire que les variables $(\delta X, \delta Y, \delta\theta)$ et (V_r, Φ_r) soient accessibles.

6.3.2 Repères bibliographiques

Le problème mentionné ci-dessus est un grand classique de la robotique autonome. Parmi les solutions les plus connues, on peut citer les lois de commandes de Kanayama et de Samson.

a) Contrôleur de Kanayama

Dans son article de 1990 *[Kanayama et al. 1990]*, Kanayama décrit une loi de commande prévue pour contrôler un véhicule sur une trajectoire dans des conditions similaires aux nôtres. La seule différence est le remplacement du contrôle en angle de braquage par un contrôle en vitesse de rotation, notée ω. Toutefois, comme on peut déduire ω de Φ et réciproquement, cela n'empêcherait pas d'appliquer cette loi.

La solution proposée par Kanayama est donnée par l'égalité suivante :

$$\begin{pmatrix} V_d \\ \omega_d \end{pmatrix} = \begin{pmatrix} V_r \cos\delta\theta + K_x \delta X \\ \omega_r + V_r(K_y \delta Y + K_\theta \sin\delta\theta) \end{pmatrix} \tag{6.1}$$

b) Contrôleur de Samson

Samson *[Samson & Ait-Abderrahim 1991]* a lui aussi proposé une loi de commande pour un robot mobile sous les mêmes hypothèses que Kanayama. L'équation de contrôle qu'il propose est la suivante :

$$\begin{pmatrix} V_d \\ \omega_d \end{pmatrix} = \begin{pmatrix} V_r \cos\delta\theta + K_x \delta X \\ \omega_r + K_y V_r \delta Y \frac{\sin\delta\theta}{\delta\theta} + K_\theta \delta\theta \end{pmatrix} \tag{6.2}$$

c) Bilan

Malgré l'efficacité de ces contrôleurs, plusieurs raison nous ont amené à choisir d'implémenter un contrôleur sous forme de problème d'inférence bayésienne :

- Tout d'abord, nous devons rappelé que notre objectif est d'intégrer notre contrôleur dans un suivi de trajectoire avec évitement d'obstacles. Pour ce type d'application les contrôleurs traditionnels ont révélé un manque de souplesse : étant très précis, ils supportent difficilement de se voir restreindre, par l'évitement d'obstacles, l'ensemble des commandes admissibles.
- Par ailleurs, nous avons constaté expérimentalement que ces contrôleurs avaient du mal a gérer de très grandes erreurs. Il est tout à fait possible que cela soit dû à une implantation imparfaite des contrôleurs. Toutefois, notre intuition est que ce problème est lié à la borne sur la vitesse de rotation, imposée par la cinématique du véhicule.
- La dernière raison, et non la moindre, est notre volonté de définir une architecture de contrôle entièrement bayésienne, du traitement de la perception à la génération des commandes.

6.3.3 Spécification du comportement de suivi de trajectoire

De même que le système d'évitement d'obstacles que nous présenterons par la suite, notre comportement de suivi de trajectoire a été conçu comme un problème d'inférence probabiliste. Pour spécifier le comportement, nous utilisons le mécanisme de fusion par diagnostic décrit en 2.3.5.

Rappel : Si A et B sont deux variables, nous définissons une variable de diagnostic I_A^B pour exprimer la cohérence entre A et B.

A et B sont alors les *variables diagnostiquées* par I_A^B.

a) Principe

Le principe fondamental de notre comportement de suivi de trajectoire est la fusion d'ordre moteurs élémentaires. Nous allons donc définir cinq modèles élémentaires – trois pour corriger l'erreur de suivi et deux pour tenir compte des commandes de référence – et utiliser ensuite un modèle de fusion pour réunir ces modèles et trouver une commandes respectant au mieux tous les modèles.

b) Modèles élémentaires pour la correction de l'erreur de suivi

*1.***Correction de l'erreur longitudinale** Intuitivement, le comportement élémentaire que nous voulons définir est le suivant : lorsque le robot est derrière le véhicule de référence, il avance pour réduire son erreur, d'autant plus rapidement que son erreur δX est grande.

Ce comportement ne met donc en jeu que la vitesse désirée V_d et l'erreur δX. Pour traduire la relation entre ces variables, nous introduisons une variable de diagnostic $I_{V_d}^{\delta X}$. Cette variable doit exprimer la cohérence entre l'erreur δX et la commande V_d. Dans ce contexte, nous dirons qu'une commande V_d est cohérente avec une erreur δX si cette commande permet de réduire l'erreur δX.

Formellement, on définit $P(I_{V_d}^{\delta X} \mid V_d\ \delta X)$ par la distribution en cloche illustrée dans la figure 6.3 :

$$P(I_{V_d}^{\delta X} \mid V_d\ \delta X) = e^{-\frac{1}{2}(V_d - \mu(\delta X))^2 S(\delta X)^{-2}}$$

On peut faire deux remarques sur cette définition :

- La valeur de $P([I_{V_d}^{\delta X} = 1] \mid V_d\ \delta X)$ atteint son maximum pour $V_d = \mu(\delta X)$. Cette valeur de V_d correspond donc à la commande idéale à appliquer pour corriger δX.
- La largeur de la cloche, définie par $S(\delta X)$ permet de définir un niveau de priorité pour la commande idéale : plus $S(\delta X)$ est petit, plus cette commande sera prioritaire dans le processus de fusion.

Pour le suivi de trajectoire, les fonctions μ et S sont définis ci-dessous :

- D'une part, on pose

  $$\mu(\delta X) = \alpha^X.\delta X$$

  Cette définition signifie que $P(I_{V_d}^{\delta X} \mid V_d\ \delta X)$ atteint son maximum pour une valeur de V_d proportionnelle à δX. Comme pour tout contrôle proportionnel, il faudra faire attention aux relations entre les gains et la constante de temps du système pour éviter les problèmes éventuels d'oscillations.
- D'autre part, on pose (cf. fig. 6.3) :

  $$S(\delta X) = \max((1 - \beta^X|\delta X|)\sigma_{\max}^X,\ \sigma_{\min}^X)$$

  Cette équation définit le niveau de priorité associé à cette commande : plus l'erreur est grande, plus le système est sûr que la correction proportionnelle est la bonne, donc plus la commande doit être prioritaire et plus $S(\delta X)$ doit être proche de $\sigma_{\min}^X$.

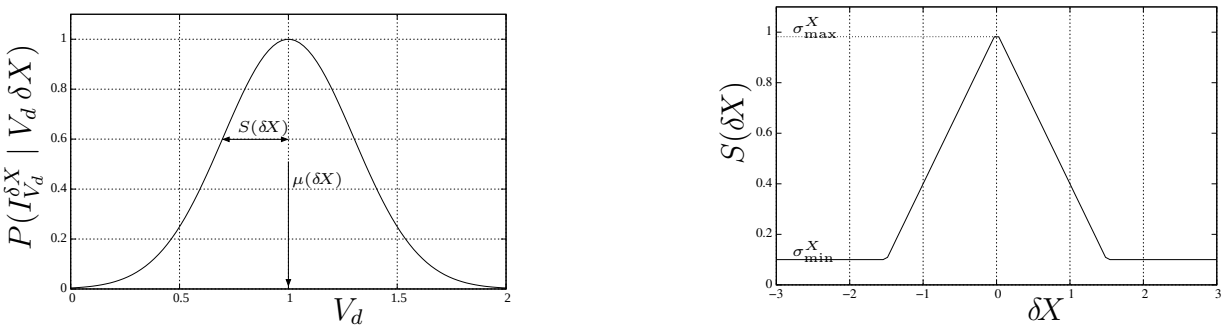


FIG. 6.3: Modèle élémentaire pour la correction de l'erreur longitudinale δX. A gauche, modèle de diagnostic paramétré par la valeur de l'erreur. A droite, évolution de la fonction $S(\delta X)$ en fonction de δX.

2.**Correction de l'erreur latérale** Nous définissons le modèle de correction de l'erreur latérale δY avec un schéma proportionnel similaire au modèle ci-dessus. Sa signification est la suivante : pour corriger une erreur latérale, le robot doit braquer, à gauche ou à droite, selon le signe de cette erreur. De nouveau, plus l'erreur est importante, plus le système est assuré de la nécessité de braquer.

Pour obtenir ce comportement, on introduit une variable de diagnostic liant Φ à δY : $I^{\delta Y}_{\Phi_d}$. On définit ensuite la distribution $P(I^{\delta Y}_{\Phi_d} \mid \Phi_d\, \delta Y)$ de façon similaire au modèle de correction de l'erreur longitudinale.

3.**Correction de l'erreur d'orientation** Finalement, le même raisonnement s'applique pour le modèle de correction de l'erreur d'orientation : pour corriger une erreur d'orientation, le robot doit braquer à gauche ou à droite, en fonction du signe de l'erreur et de la direction du mouvement (c'est à dire le signe de V_d).

On introduit donc la variable $I^{\delta\theta}_{\Phi_d}$ et on définit $P(I^{\delta\theta}_{\Phi_d} \mid \Phi_d\ \delta\theta\ V_d)$ de façon similaire aux modèles précédent.

c) Modèles élémentaires pour l'utilisation des commandes de référence

Intuitivement, tant que l'erreur de suivi est faible, le suivi de trajectoire peut se faire en appliquant directement les commandes de référence.

Pour obtenir ce comportement, nous introduisons deux nouvelles variables de diagnostic : $I^{V_r}_{V_d}$ et $I^{\Phi_r}_{\Phi_d}$. La cohérence entre une commande désirée et une commande de référence sera tout simplement leur proximité.

Formellement, on définit $P(I^{V_r}_{V_d} \mid V_d\, V_r)$ en posant :

$$P(I^{V_r}_{V_d} \mid V_d\, V_r) = e^{-\frac{1}{2}(V_d - V_r)^2 \sigma^{-2}_{V_r}}$$

De cette façon, la valeur de la probabilité $P([I^{V_r}_{V_d} = 1] \mid V_d\, V_r)$ atteint son maximum pour $V_d = V_r$, c'est à dire lorsque la commande désirée est égale à la commande de référence. Pour

exprimer le fait que l'application de cette commande n'est prioritaire que lorsque l'erreur de suivi est faible, on choisit une valeur de σ_{V_r} comprise entre $\sigma_{\min}$ et $\sigma_{\max}$ plutôt proche de $\sigma_{\max}$.

La distribution sur $I^{\Phi_r}_{\Phi_d}$ est définie en utilisant le même raisonnement.

d) Modèle de fusion

L'objectif du modèle de fusion est de rassembler les cinq modèles présentés ci-dessus. Le modèle obtenu permettra ensuite de choisir les commandes à appliquer en trouvant un compromis entre les différents modèles.

Formellement, le modèle de fusion est une distribution conjointe sur les 12 variables introduites dans les modèles élémentaires. On construit donc la décomposition suivante :

$$\begin{aligned} &P(V_d\,\Phi_d\,V_r\,\Phi_r\,\delta X\,\delta Y\,\delta\theta\,I^{\delta X}_{V_d}\,I^{V_r}_{V_d}\,I^{\delta Y}_{\Phi_d}\,I^{\delta\theta}_{\Phi_d}\,I^{\Phi_r}_{\Phi_d}) = \\ &\quad P(V_d\ \Phi_d)\ P(V_r\ \Phi_r)\ P(\delta X\ \delta Y\ \delta\theta)\ P(I^{\delta X}_{V_d} \mid V_d\ \delta X)\ P(I^{V_r}_{V_d} \mid V_d\ V_r) \\ &\quad P(I^{\delta Y}_{\Phi_d} \mid \Phi_d\ \delta Y)\ P(I^{\delta\theta}_{\Phi_d} \mid \Phi_d\ \delta\theta\ V_d)\ P(I^{\Phi_r}_{\Phi_d} \mid \Phi_d\ \Phi_r) \end{aligned} \tag{6.3}$$

Dans la distribution conjointe précédente, toutes les variables diagnostiquées sont supposées indépendantes et de distribution uniforme. Toute l'information concernant leurs relations sera donc contenue dans les distributions sur les variables de diagnostic.

A partir de cette distribution, en utilisant la règle de Bayes, on peut inférer :

$$P(V_d\,\Phi_d \mid (V_r\,\Phi_r)\,(\delta X\,\delta Y\,\delta\theta)\,[I^{\delta X}_{V_d}=1]\,[I^{V_r}_{V_d}=1]\,[I^{\delta Y}_{\Phi_d}=1]\,[I^{\delta\theta}_{\Phi_d}=1]\,[I^{\Phi_r}_{\Phi_d}=1]) \tag{6.4}$$

Dans cette expression, toutes les variables de diagnostic sont supposées vraies (égale à 1). Cela signifie que l'on souhaite que toutes les relations de cohérence soient respectées. Par exemple, $I^{\delta\theta}_{\Phi_d}=1$ signifie que l'on veut trouver un angle de braquage Φ_d cohérent avec $\delta\theta$, c'est à dire un angle de braquage qui tant à réduire l'erreur $\delta\theta$.

Lorsque deux modèles sont antagonistes, la commande qui aura une probabilité maximale représentera un compromis entre les commandes proposés par les deux modèles. C'est dans la résolution de ces antagonismes que les niveaux de priorité définis par les modèles élémentaires interviennent. En effet, dans ce cas le compromis est le barycentre des commandes proposées par chaque modèle, pondéré par les niveaux de priorité.

On peut voir dans la figure 6.4 une illustration de comportement de notre système de suivi. Pour chaque graphe, la commande résultante est celle qui maximise $P(V \mid \delta X\ V_c)$ ou $P(\Phi \mid \delta Y\ \delta\theta\ \Phi_c)$. Comme la courbe $P(V \mid \delta X V_c)$ est plus proche de $P(V \mid \delta X)$ que de $P(V \mid V_c)$, on constate que l'erreur longitudinale δX a bien plus d'influence sur la vitesse choisie par le véhicule que la vitesse du véhicule de référence. De la même manière, l'angle de braquage optimal est un compromis entre ce qu'il est nécessaire de faire pour corriger l'erreur latérale δY et l'erreur d'orientation $\delta\theta$. Il n'est influencé que très légèrement par l'angle de braquage de référence.

6.3.4 Convergence : expression analytique du contrôleur

Afin de pouvoir prouver les propriétés de convergence de notre contrôleur, nous avons cherché à en déterminer une expression analytique similaire aux lois de Samson et Kanayama.

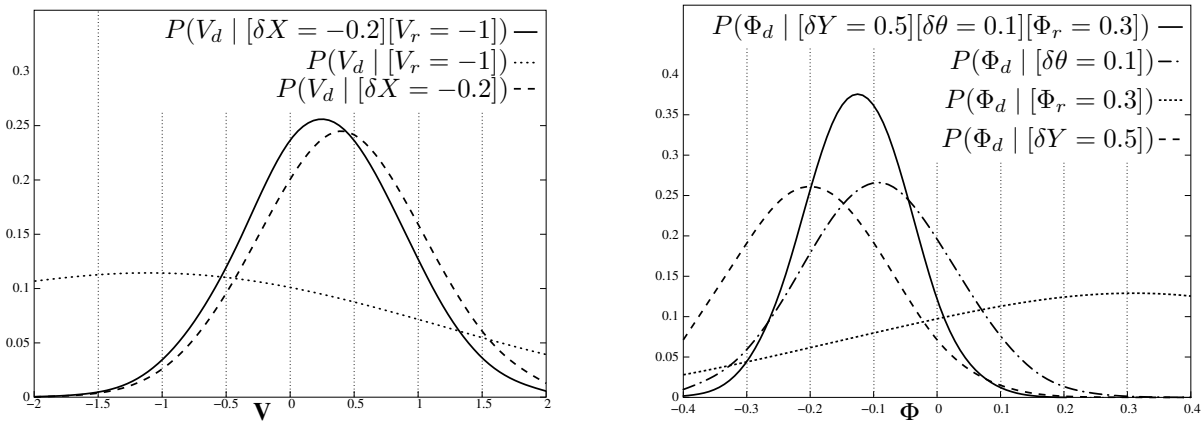


FIG. 6.4: Suivi de trajectoire : résultats de la fusion de commandes.

Dans la pratique, l'exécution de notre comportement de suivi de trajectoire se fait en deux temps. Dans un premier temps, la distribution 6.4 est estimée sur le domaine des commandes admissibles, puis, dans un second temps, la commande qui maximise cette distribution est choisie pour être appliquée au robot.

En développant l'expression

$$P(V_d\,\Phi_d \mid (V_r\,\Phi_r)\ (\delta X\,\delta Y\,\delta\theta)\ [I_{V_d}^{\delta X}=1]\ [I_{V_d}^{V_r}=1]\ [I_{\Phi_d}^{\delta Y}=1]\ [I_{\Phi_d}^{\delta\theta}=1]\ [I_{\Phi_d}^{\Phi_r}=1])$$

on constate deux points importants :

- D'une part, l'expression des fonctions de diagnostic sous forme de distance de Mahalanobis garantit que le maximum est unique : la distribution finale est un produit de distributions en cloche, similaires à des gaussiennes (cf. annexe A).
- D'autre part, on se rend compte que la dépendance entre V_d et Φ_d est relativement faible : V_d ne dépend pas de Φ_d et Φ_d ne dépend que du signe de V_d. On peut donc trouver deux expressions indépendantes pour V_d et Φ_d.

Puisqu'on travaille avec des distributions en cloche, il est facile de déterminer de façon analytique les commandes de probabilité maximale, il suffit de déterminer le maximum de l'expression ci-dessus. Après développement des calculs, on trouve :

$$V_d = \frac{\alpha^X\,\delta X\,\sigma_{V_r} + V_r\,S(V_d,\delta X)}{\sigma_{V_r} + S(V_d,\delta X)} \tag{6.5}$$

$$\Phi_d = \frac{\alpha^Y\,\delta Y\,S(\Phi_d,\delta\theta)\,\sigma_{\Phi_r} + \mathrm{sgn}(V_d)\,\alpha^\theta\,\delta\theta\,S(\Phi_d,\delta Y)\,\sigma_{\Phi_r} + \Phi_r\,S(\Phi_d,\delta Y)\,S(\Phi_d,\delta\theta)}{S(\Phi_d,\delta\theta)\,\sigma_{\Phi_r} + S(\Phi_d,\delta Y)\,\sigma_{\Phi_r} + S(\Phi_d,\delta Y)\,S(\Phi_d,\delta\theta)} \tag{6.6}$$

A partir de ces expressions il devrait être possible de déterminer les propriétés de convergence de notre contrôleur. Toutefois, la complexité des équations mises en jeu nous a empêché de mener les calculs à leur terme.

De façon empirique, il semblerait que le contrôleur stabilise le véhicule sur une position présentant un léger retard par rapport à la position de référence. Pour une trajectoire de référence rectiligne à $1.0ms^{-1}$, on observe une stabilisation expérimentale proche de $(\delta X = -0.5m, \delta Y = 0, \delta\theta = 0)$.

6.3.5 Convergence expérimentale

a) Protocole expérimental

Pour illustrer les performances de notre comportement de suivi d'un véhicule de référence, nous avons utilisé le protocole expérimental suivant :

- D'une part deux trajectoires particulière sont considérées : une trajectoire rectiligne à vitesse constante ($\Phi_r = 0$rad et $V_r = 0.8ms^{-1}$) et une trajectoire circulaire à vitesse et angle de braquage constants ($\Phi_r = 0.2$rad et $V_r = 0.8ms^{-1}$). De par la symétrie de notre plate-forme expérimentale (cf. 7.3), ces trajectoires peuvent être considérées comme représentatives des situations réelles auxquelles notre comportement de suivi sera confronté.
- D'autre part, le véhicule est placé successivement sur un ensemble de positions de départ correspondant à de grandes erreurs de suivi : ± 12 mètres d'erreur longitudinale, ± 6 mètres d'erreur latérale et ± 75 degrés d'erreur d'orientation. Pour chaque erreur initiale, l'historique de l'erreur de suivi est enregistré jusqu'à la stabilisation. Ces historiques, pour l'ensemble des erreurs initiales, sont ensuite rassemblés sur les graphes 6.5.

b) Résultats

La figure 6.5 montre les réponses de notre comportement de suivi face à ces situations expérimentales. Sur ces graphes, chaque ligne correspond à une trajectoire de convergence.

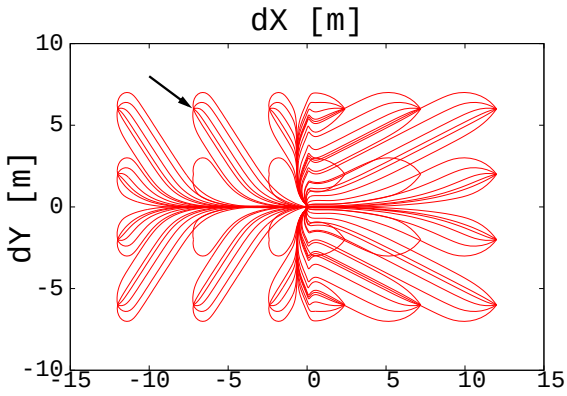
On constate que, quelle que soit l'erreur initiale, même si celle-ci est très importante, le comportement de suivi trajectoire génère des commandes qui amènent rapidement le véhicule sur sa trajectoire (erreur de suivi proche de zéro).

6.3.6 Conclusion

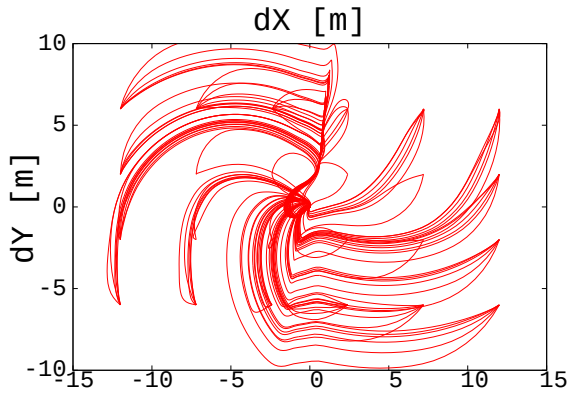
Dans cette section, nous avons montré comment utiliser le formalisme de l'inférence bayésienne pour décrire un comportement de suivi de trajectoire. Grâce à une expression sous forme de fusion par diagnostic, nous pouvons maîtriser les différents niveaux de priorité associés à la correction des différentes dimensions de l'erreur de suivi.

Le comportement de suivi de trajectoire que nous obtenons à plusieurs avantages par rapport à des contrôleurs classiques :

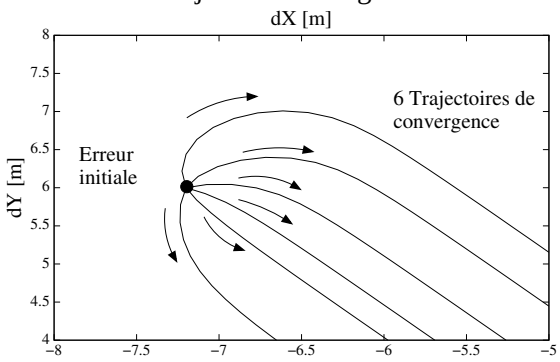
- Tout d'abord, étant exprimé comme un problème d'inférence bayésienne, il est prêt à être intégré dans les architectures de contrôle bayésiennes que nous présenterons au chapitre 7.
- Ensuite, même si nous avons fait le choix de spécifier *a priori* les modèles élémentaires, ce n'est pas la seule spécification possible. Il est tout à fait envisageable d'utiliser des techniques d'apprentissage bayésien *[Lebeltel et al. 2003]* pour apprendre ces modèles, par exemple à partir de l'observation d'un conducteur humain.
- Finalement, le comportement que nous obtenons s'est montré, expérimentalement, capable de réagir correctement à des erreurs de suivi très importantes.



Trajectoire rectiligne.



Trajectoire circulaire.



Ci-contre, zoom sur le début du suivi pour 6 trajectoires de convergence particulières. La position de départ, marquée d'une flèche en haut à gauche et d'une pastille noire ici, est la même pour toutes les trajectoires. Seule l'orientation initiale change, entre -75^o et $+75^o$.

FIG. 6.5: Suivi d'une trajectoire : ensemble des trajectoires de convergence de l'erreur de suivi $(\delta X, \delta Y)$ à partir d'un ensemble de grandes erreurs initiales.

6.4 Évitement d'obstacles réactif

6.4.1 Présentation du problème

Pour faire évoluer un robot dans un environnement inconnu, ou non contrôlé, il est nécessaire de lui fournir un moyen de préserver sa sécurité et celle des personnes évoluant autour de lui. En particulier, pour rendre plus robuste l'exécution de trajectoires décrites à un plus haut niveau de représentation, un système d'évitement d'obstacles doit être préparé à réagir à des changements imprévus de l'environnement. Cette section présente les principes de notre module d'évitement d'obstacles.

Objectif : Le but de la technique d'évitement d'obstacles présentée dans cette partie est de venir en complément d'un système de conduite de plus haut niveau, de type suivi de trajectoire planifiée ou suivi de trajectoire sensorimotrice.

6.4.2 Étude bibliographique

La plupart es méthodes d'évitement d'obstacles réactif sont des méthodes locales (*[Fox et al. 1997]* et *[Brock & Khatib 1999]*). Les approches locales n'essaient pas de modéliser tout l'environnement du robot. Elles ont plutôt pour but d'utiliser les mesures faites par les capteurs pour en déduire des commandes sûres. Ainsi, en étant plus simples et d'un coût calculatoire moindre, elles sont plus appropriées à une réaction rapide à un environnement non-statique. Cependant, on ne peut attendre de ces méthodes qu'elles génèrent une solution optimale. Dans certaines configurations des obstacles, il arrive même que le système soit bloqué dans une situation dont il ne peut sortir tout seul.

a) Champs de potentiel

L'idée générale des méthodes de champs de potentiel, proposées initialement par O. Khatib en 1986, est de construire une fonction résumant les buts de navigation (la plupart du temps, atteindre une configuration particulière) et le besoin d'éviter les collisions. Cette fonction doit décroître en se rapprochant de la position but, et croître en s'approchant des obstacles. Le problème de navigation se transforme alors en un problème d'optimisation : trouver les commandes qui permettent d'amener le robot au minimum global de la fonction. Cette fonction n'est souvent définie que par rapport au but et aux obstacles, cependant, d'autres contraintes peuvent y être incluses simplement, par addition ou produit selon leur nature.

De nombreuses extensions aux champs de potentiel ont été proposées depuis 1986. On peut citer par exemple les VFF (Champs de force virtuels) *[Borenstein & Koren 1989]*, les VFH (Histogramme de champs de vecteurs) *[Borenstein & Koren 1991]*, et leurs extensions telles que VFH+ *[Ulrich & Borenstein 1998]* et VFH* *[Ulrich & Borenstein 2000]*. Leur principe de base étant de trouver le meilleur chemin pour atteindre le but parmi les chemins sûrs.

b) Steering Angle Field (SAF)

La méthode SAF[1], proposée par Feiten et al en 1994, utilise les obstacles pour contraindre l'angle de braquage dans un domaine continu. En parallèle, le contrôle de la vitesse est un processus de négociation itératif entre le module de pilotage de haut niveau et le module d'évitement d'obstacles local.

Une des premières extensions à cette méthode a été publiée dans *[Simmons 1996]*. Elle formule le problème d'évitement de collision en un problème d'optimisation dans l'espace des vitesses (vitesses de translation et de rotation).

c) Dynamic Window

L'approche Dynamic Window[2] a été décrite dans *[Fox et al. 1997]*. Elle propose elle aussi d'éviter les obstacles en explorant l'espace des commandes de façon à maximiser une fonction

[1] Littéralement, Champs d'angles de braquage.
[2] Littéralement, Fenêtre dynamique.

objectif. Cette fonction prend en compte la progression vers la position but, la position des obstacles les plus proches et la vitesse actuelle du robot. Cette méthode est directement dérivée de la dynamique du robot, ce qui la rend particulièrement adaptée aux mouvements à grande vitesse.

Le problème lié à la complexité algorithmique du processus d'optimisation est résolu en utilisant les propriétés dynamiques du robot (accélération linéaire ou angulaire limitée) pour limiter l'espace de recherche . Ces contraintes qui réduisent l'espace de recherche sont appelées *Contraintes Dures* car elles doivent impérativement être respectées. Au contraire, lorsque la fonction objectif inclut des préférences sur le mouvement du robot, on parle de *Contraintes Molles*.

d) Environnements dynamiques et Velocity Obstacles

Dans le cas particulier d'obstacles en mouvement, des méthodes spécifiques ont été développées (*[Large et al. 2002],[Blondin 2002]*) en utilisant la notion de *Velocity Obstacle*. Grossièrement, cette notion consiste à projeter les obstacles perçus et leur mouvement prévu, dans l'espace des commandes sûres. Chaque objet mobile génère donc un ensemble d'obstacles dans l'espace des commandes. Ces obstacles correspondent aux commandes qui vont provoquer une collision dans un futur plus ou moins lointain.

Dans le cas général, les paramètres du mouvement des obstacles ne sont pas connus *a priori* et doivent être déduites des observations. Les réactions d'évitement d'obstacles sont alors calculées en fonction de ces prévisions. Actuellement, ce qui rend les méthodes d'évitement d'obstacles dynamiques difficilement applicables est la difficulté d'obtenir une prévision fiable de trajectoire à partir des observations. Pour cette raison, nous ne nous intéresserons ici qu'aux méthodes, dites réactives, qui proposent une commande sûre uniquement à partir de l'observation courante.

e) Évitement d'obstacles et suivi de trajectoire

Lorsque l'on cherche à réaliser des manoeuvres d'évitement d'obstacles pendant la réalisation d'une trajectoire planifiée, plusieurs problèmes apparaissent : D'abord, si on considère un robot non-holonome, on peut supposer que la planification a tenu compte de la non-holonomie et ainsi a planifié une trajectoire faisable[3]. Lorsqu'un algorithme d'évitement d'obstacles réactif génère des commandes d'évitement, le véhicule quitte sa trajectoire planifiée. Il n'y est alors plus garanti que l'objectif initial de la trajectoire reste atteignable.

Pour répondre à ce problème, des solutions particulières ont été proposées dans *[Lamiraux et al. 2002; 2004]*. Le principe de ces méthodes est de chercher à déformer la trajectoire dans son ensemble afin d'éviter l'obstacle imprévu tout en restant certain que la trajectoire atteint l'objectif initial et vérifie les contraintes cinématiques du véhicule.

Ces méthodes semblent très prometteuses. Toutefois, pour l'instant elle sont rendues difficilement applicables par leur complexité calculatoire, en particulier sur une voiture autonome. Dans les expérimentations que nous avons menées *[Lefebvre et al. 2004]*, après détection de l'obstacle, le véhicule s'arrêtait pendant quelques minutes pour calculer la trajectoire déformée, puis repartait pour terminer l'exécution.

[3]Faisable : respectant à tout instant les contraintes cinématiques du véhicule.

Afin d'obtenir une plus grande réactivité, nous avons choisi de sacrifier la garantie de terminaison du suivi de trajectoire. L'évitement d'obstacles que nous allons présenter dans cette section ne calculera sa commande qu'à partir de la perception courante.

Contribution Au vu de tout le travail déjà réalisé sur l'évitement d'obstacles, nous devons signaler que notre objectif est plus de réfléchir sur l'expression de l'évitement d'obstacles que d'y proposer une nouvelle solution. **Nous avons donc conçu, avec C.Koike, un système d'évitement d'obstacles exprimé comme un problème d'inférence probabiliste. L'originalité de notre approche se situe principalement dans son expression et dans la sémantique qu'elle permet de spécifier.**

6.4.3 Situation expérimentale

Rappelons que le CyCab peut être piloté grâce à une vitesse linéaire V et à un angle de braquage Φ. Il est équipé d'un télémètre laser à balayage sur 180^o avec une résolution angulaire de 0.5^o. Cependant, cette résolution entraîne des volumes de données trop importants pour la gestion de la tâche d'évitement d'obstacles en temps réel. En conséquence, nous résumons l'information à huit valeurs, les distances à l'obstacle le plus proche dans des secteurs angulaires de 22.5 degrés (cf. fig. 6.6). Nommons $D_k, k = 1 \dots 8$ les variables probabilistes correspondant à ces 8 mesures.

En outre, nous supposons que le robot est piloté par un système de plus haut niveau (conduite au joystick ou suivi de trajectoire par exemple) qui lui fournit une commande désirée sous la forme d'un couple (V_d, Φ_d).

Finalement, comme dans notre cas particulier le système n'a pas de perception sur l'arrière, il est impossible de le laisser se déplacer en marche arrière. Notons toutefois que la méthode présentée ici pourrait être adaptée à une perception panoramique.

Objectif : Dans ce contexte, notre objectif est de rechercher les commandes à transmettre au système d'asservissement bas-niveau, de façon à garantir la sécurité du véhicule tout en appliquant les commandes désirées autant que possible. Nous nous plaçons dans le cas où les obstacles ne sont pas agressifs : le but des obstacles est aussi de rester en sécurité (on retrouve le concept de navigation *réflexive[Kluge 2003]* : c'est à dire qu'on suppose que le comportement des objets dans l'environnement est similaire à notre comportement).

Dans le reste de cette section, nous allons présenter deux types d'inférence pour atteindre cet objectif : la première inférence est construite à partir de règles qui expriment les commandes à appliquer pour éviter les obstacles. On parle alors de programmation prescriptive. La seconde inférence utilise plutôt une approche proscriptive : l'évitement d'obstacles ne fait qu'interdire les commandes dangereuses, mais ne donne pas d'indications sur la commande à appliquer.

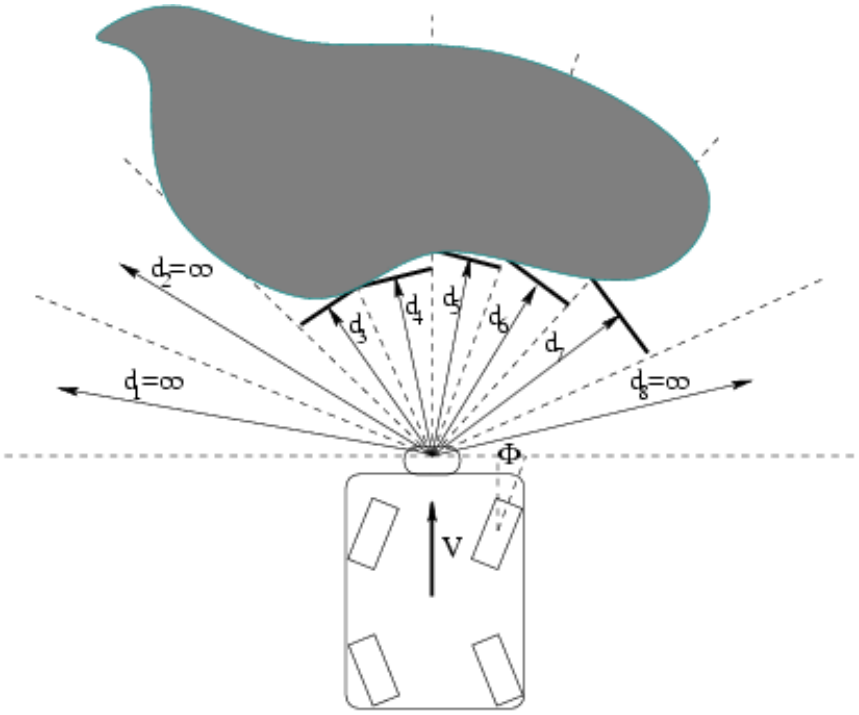


FIG. 6.6: Évitement d'obstacles : situation et variables.

6.4.4 Programmation prescriptive

Le modèle que nous allons décrire maintenant transforme le problème de l'évitement d'obstacles en un problème de fusion bayésienne de sous-modèles prescriptifs. On commence par exprimer le comportement à adopter pour réagir à chaque donnée capteur indépendamment, puis on utilise un processus de fusion de commandes pour générer un comportement qui tient compte de toutes les données.

a) Définition de sous-modèles

Dans un secteur angulaire i, on définit une distribution de probabilité sur (V, Φ) connaissant les commandes désirées et la distance D_i mesurée dans ce secteur :

$$P_i(V\Phi \mid V_d\Phi_d D_i) = P_i(V \mid V_d D_i) P_i(\Phi \mid \Phi_d D_i) \tag{6.7}$$

où $P_i(V \mid V_d\Phi_d D_i)$ et $P_i(\Phi \mid \Phi_d D_i)$ sont des distributions gaussiennes centrées respectivement sur $\mu_V(V_d, D_i)$ et $\mu_\Phi(\Phi_d, D_i)$, avec des écarts types $\sigma_V(V_d, D_i)$ et $\sigma_\Phi(\Phi_d, D_i)$. Les fonctions $\mu_V, \mu_\Phi, \sigma_V, \sigma_\Phi$ sont définies de façon à ressembler à des sigmoïdes (cf. fig. 6.7 et 6.8).

On peut faire deux remarques principales dans les figures 6.7 et 6.8. D'abord, pour ce qui est de μ_V et μ_Φ, on peut voir que les fonctions sont conçues de façon à exprimer l'idée simple suivante : plus l'obstacle est éloigné, plus on essaye de suivre la commande désirée, et de façon inverse, plus l'obstacle est proche plus les fonctions μ chercheront à mettre le système en sécurité. En ce qui concerne le contrôle de la vitesse, nous considérons que le système est en sécurité

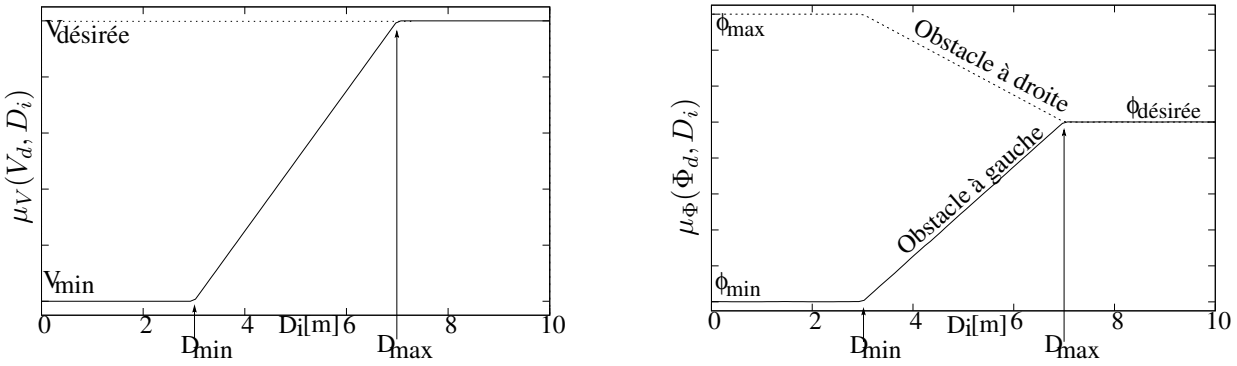


FIG. 6.7: Définition des moyennes de $P_i(V \mid V_d D_i)$ et $P_i(\Phi \mid \Phi_d D_i)$ selon les distances mesurées.

lorsque sa vitesse est nulle. Pour l'angle de braquage, plus l'obstacle est proche plus le robot cherche à s'en éloigner en braquant au maximum de ses possibilités. Si l'obstacle est à droite, le robot braquera à gauche et si l'obstacle est à gauche, il braquera à droite. L'existence de ces deux solutions est traduite par les deux courbes sur le graphe de droite de la figure 6.7.

Ensuite, les écarts types de la figure 6.8 peuvent être vus comme des niveaux de contrainte. Par exemple, quand un obstacle est très proche du robot (distance D_i petite), la vitesse de ce dernier doit être *impérativement* contrainte à zéro. L'adverbe "impérativement" est traduit par un écart type proche de zéro. Au contraire, quand l'obstacle est loin (ou inexistant), le robot peut éventuellement suivre la commande désirée, mais le fait d'appliquer une commande différente n'augmente pas le risque de collision. Ce faible niveau de contrainte (adverbe "éventuellement") est le résultat d'un écart type important.

Voyons maintenant comment ces différents niveaux de contrainte sont fusionnés dans un cadre probabiliste.

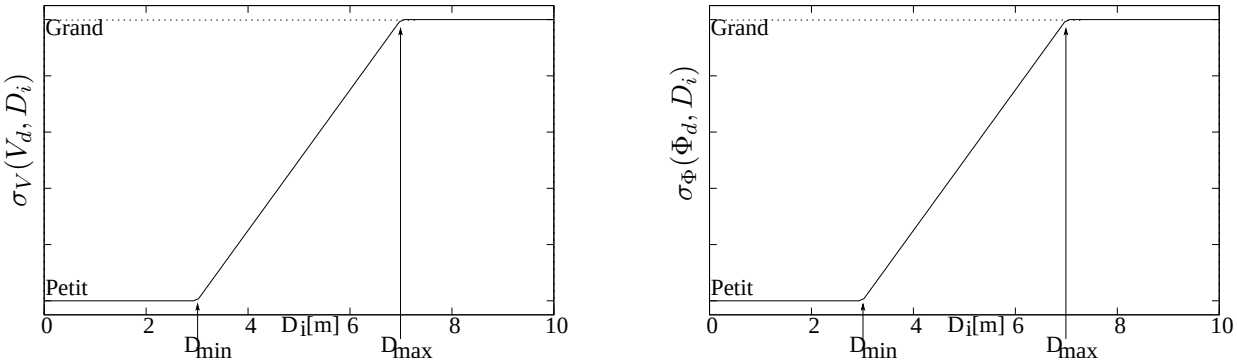


FIG. 6.8: Définition des écarts types de $P_i(V \mid V_d D_i)$ et $P_i(\Phi \mid \Phi_d D_i)$ selon les distances mesurées.

b) Fusion des commandes

Connaissant les commandes désirées et la plus courte distance mesurée dans son secteur, chaque sous-modèle, défini par $P_i(V\Phi \mid V_d\Phi_d D_i)$, nous fournit une distribution de probabilité sur les commandes à appliquer. Comme nous ne pouvons appliquer qu'une commande, nous allons devoir fusionner ces huit distributions pour qu'il n'en reste qu'une, et extraire de cette dernière une commande qui répond au mieux à la commande désirée et aux contraintes de sécurité.

Pour cela, nous nous plaçons dans le contexte de la PBR (cf. chapitre 2) et nous définissons la distribution conjointe suivante :

$$\begin{aligned} P(V\ \Phi\ V_d\ \Phi_d\ D_1 \ldots D_8\ S) \ &= \ P(D_1 \ldots D_8)\ P(V_d\ \Phi_d) \\ &\quad\ P(S)\ P(V\ \Phi\ \mid\ V_d\ \Phi_d\ D_1 \ldots D_8\ S) \end{aligned} \tag{6.8}$$

où la variable $S \in [1 \ldots 8]$ est définie de façon à ce que lorsque $S = i$, le sous-modèle i contrôle le robot. $P(D_1 \ldots D_8)$ et $P(V_d\Phi_d)$ sont des distributions non définies : comme nous savons que nous n'aurons pas besoin d'elles dans les futurs calculs, il n'est pas nécessaire de les spécifier.

De plus, comme il n'est pas nécessaire de favoriser un sous-modèle particulier, nous définissons $P(S)$ par une distribution uniforme. Finalement, la sémantique de S est mise en exergue par la définition de $P(V\Phi \mid V_d\ \Phi_d\ D_1 \ldots D_8\ S)$:

$$P(V\ \Phi \mid V_d\ \Phi_d\ D_1 \ldots D_8\ [S = i]) = P_i(V\ \Phi \mid V_d\ \Phi_d\ D_i)$$

En utilisant l'équation 6.8, nous pouvons maintenant exprimer la distribution qui nous intéresse réellement :

$$\begin{aligned} &P(V\ \Phi\ \mid\ V_d\ \Phi_d\ D_1 \ldots D_8) = \\ &\quad \sum_S (P(S)P(V\ \Phi\ \mid\ V_d\ \Phi_d\ D_1\ \ldots\ D_8\ S)) \end{aligned} \tag{6.9}$$

Cette équation est en fait le lieu où les différents niveaux de contrainte exprimés par les fonctions σ_V et σ_Φ vont prendre tout leur sens. Plus un sous-modèle exprimera une contrainte de sécurité forte (obstacle proche), plus sa distribution sur les commandes sera piquée. Ainsi les sous-modèles qui ne voient pas d'obstacles dans leur secteur contribueront à la somme par une distribution plutôt plate, alors que ceux qui perçoivent des obstacles dangereux ajouteront une distribution piquée, et de cette façon, auront plus d'influence (voir fig. 6.9). Finalement la commande réellement exécutée par le robot sera celle qui maximise $P(V\ \Phi\ \mid\ V_d\ \Phi_d\ D_1 \ldots D_8)$ (équation 6.9).

c) Résultats

La figure 6.10 illustre le résultat du système d'évitement d'obstacles appliqué sur un exemple simulé. Le CyCab simulé est conduit manuellement (au moyen d'un joystick) dans un environnement carré. Dans cette situation spécifique, le conducteur demande en permanence une vitesse maximale et un angle de braquage nul. Sur la trajectoire en pointillé, on peut observer que l'évitement d'obstacles courbe la trajectoire aux abords des murs. De plus, en regardant la densité de points, on peut imaginer la vitesse du robot : il freine lorsqu'il s'approche des murs et suit la vitesse désirée quand les obstacles ne sont pas trop menaçants.

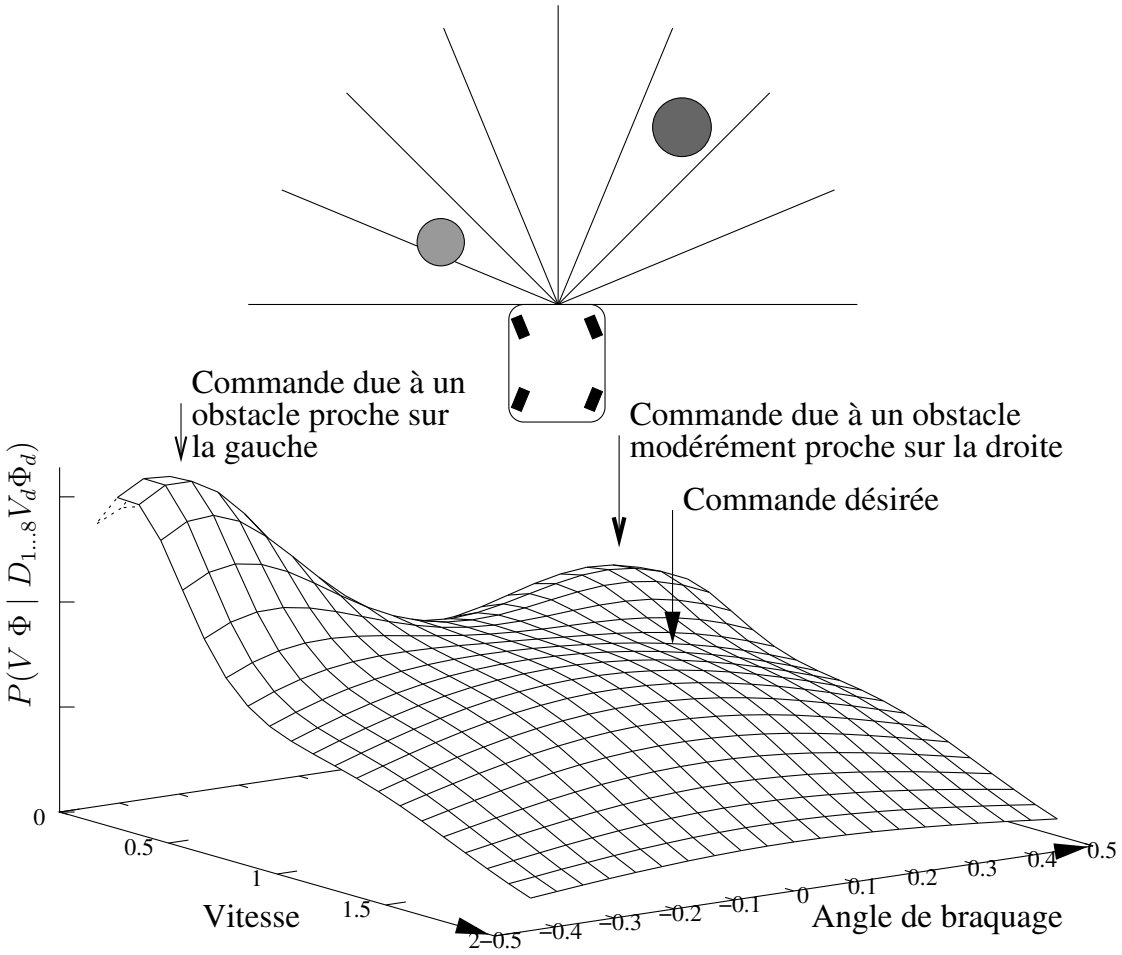


FIG. 6.9: Distribution de probabilité sur la vitesse et l'angle de braquage, résultant du processus d'évitement d'obstacles.

6.4.5 Amélioration de la sémantique : la programmation proscriptive

L'approche précédente fournit un résultat satisfaisant en termes d'évitement d'obstacles. Cependant, on peut se demander si la sémantique d'un tel module est réellement pertinente. **En effet, lorsqu'on désire qu'un robot se déplace sans collision, il peut être plus judicieux de spécifier ce qu'il** ne doit pas **faire plutôt que ce qu'il** doit **faire.**

Nous allons maintenant spécifier un modèle de fusion bayésienne fondé sur des sous-modèles proscriptifs.

a) Définition de sous-modèles

Dans la spécification prescriptive des sous-modèles, on voulait exprimer les commandes à appliquer en fonction des distances mesurées ainsi que le niveau de contrainte associé à ces commandes. La relation entre (V, Φ) et D_i était donc d'ordre fonctionnel : on voulait donc exprimer quelque chose qui ressemble à $(V, \Phi) = f(D_i)$. Il était donc naturel de construire une distribution

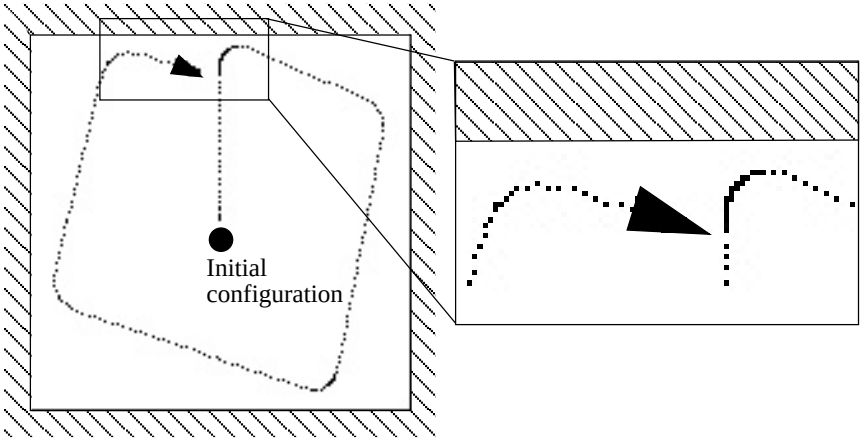


FIG. 6.10: Trajectoire du robot lorsque la commande désirée est constante : vitesse maximum et angle de braquage nul. La trace est réalisée en dessinant un pixel toutes les 10 millisecondes.

sur (V, Φ) **connaissant** D_i.

Dans le cas d'une approche proscriptive, la sémantique à exprimer est différente : on cherche à exprimer les commandes qui sont sûres au vu d'une mesure de distance D_i. On veut donc *une relation*, au sens mathématique du terme[4], mais cette relation n'est pas nécessairement une fonction. Pour exprimer ce type de modèle, le plus simple est d'utiliser une méthode de fusion par diagnostic (cf. chapitre 2).

*1.***Évitement d'obstacles** Dans un secteur angulaire i, on définit une variable de diagnostic I_i, variable booléenne qui exprime la compatibilité en termes de sécurité entre (V, Φ) et D_i. En pratique, pour calculer $P(I_i \mid (V, \Phi)\ D_i)$, on définit une fonction $V_{max}(\Phi, D_i)$. Étant donné un angle de braquage Φ, $V_{max}(\Phi, D_i)$ est la vitesse maximale applicable sans collision avec le plus gros obstacle ayant pu provoquer la mesure D_i. La figure 6.11 illustre la construction de cette fonction.

On construit ensuite :

– soit une distribution en plateau (éq. 6.10),

$$P(I_i = 1 \mid (V, \Phi)\ D_i) = (V <= V_{max}(\Phi, D_i)) \tag{6.10}$$

– Soit une sigmoïde en deux dimensions (éq. 6.11) si on veut une distribution continue.

$$P(I_i = 1 \mid (V, \Phi)\ D_i) = 1 - \frac{1}{1 + e^{-4\alpha(V - V_{max}(\Phi, D_i))}} \tag{6.11}$$

α est en relation avec la pente de la sigmoïde au point d'inflexion. Il est important de choisir une pente assez forte pour deux raisons : d'abord, cela permet d'exprimer le caractère impératif de la contrainte de sécurité, en rapprochant l'équation 6.11 de l'équation 6.10.

[4]Au sens mathématique, une relation entre deux ensembles A et B est un sous-ensemble de leur produit cartésien. Pour $a \in A$ et $b \in B$, on a $a\,R\,b \triangleq (a, b) \in R$. Une fonction est une forme spéciale de relation.

De cette manière, le respect des contraintes de sécurité est bien considéré comme une contrainte dure. Ensuite, une pente forte au point d'inflexion impose des plateaux très plats, ce qui permet de considérer de manière équiprobable toutes les commandes sans danger.

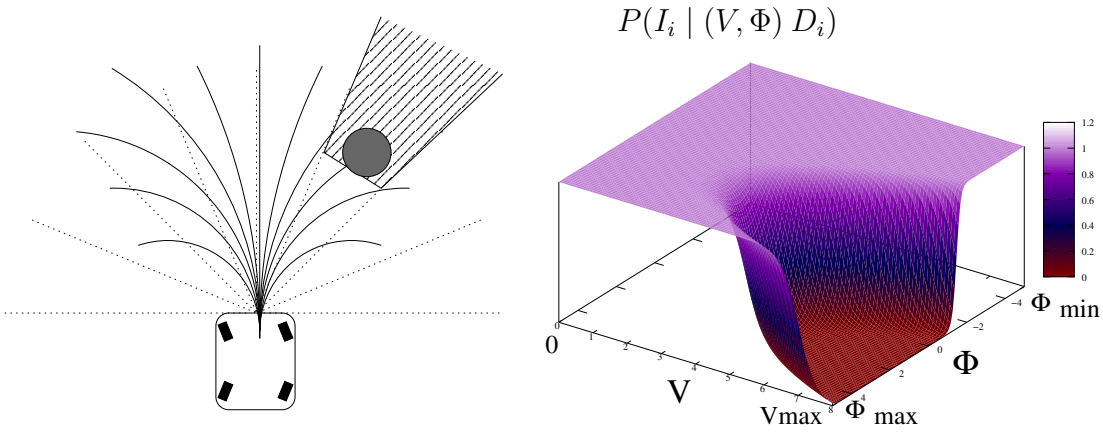


FIG. 6.11: Construction de $P(I_i \mid (V, \Phi)\ D_i)$ (à droite) : le modèle cinématique du robot est utilisé pour tester les commandes admissibles en fonction d'une mesure de distance (à gauche).

2.**Suivi de consigne** La commande qui sera générée par le module d'évitement d'obstacles doit d'une part être sûre, et d'autre part, respecter autant que possible la commande désirée par le pilote de haut-niveau. Il s'agit ici d'une contrainte molle, par opposition aux contraintes de sécurité. Pour exprimer cette contrainte molle, on commence par définir une variable de diagnostic I_d qui juge de la cohérence entre les commandes à appliquer (V, Φ) et les commandes désirée (V_d, Φ_d). On donne ensuite à $P(I_d \mid (V, \Phi)\ (V_d, \Phi_d))$ une forme de cloche assez évasée :

$$P(I_d \mid (V, \Phi)\ (V_d, \Phi_d)) = e^{\frac{1}{2}\left(\frac{V - V_d}{\sigma_V}\right)^2} e^{\frac{1}{2}\left(\frac{\Phi - \Phi_d}{\sigma_\Phi}\right)^2} \tag{6.12}$$

La différence entre σ_V et σ_Φ donne l'importance relative accordée au respect de V_d et Φ_d. Si σ_V est plus grand que σ_Φ, alors il est plus important de respecter l'angle de braquage que de suivre la vitesse. Face à un obstacle, le robot préférera s'arrêter plutôt que de changer son angle de braquage. Dans le cas contraire, c'est le respect de la vitesse qui passe avant la direction. Puisqu'il est prêt a changer souvent de direction pour ne pas ralentir, le robot adopte alors un comportement beaucoup plus agressif. Le choix des valeurs de ces deux paramètres dépendra de l'application. Cependant, il est généralement plus sûr d'avoir σ_Φ légèrement inférieur à σ_V.

b) Fusion des modèles

Pour fusionner un ensemble de sous-modèles de diagnostic, on utilise le programme bayésien illustré par la figure 6.12. Finalement, la commande appliquée est celle qui maximise la distribution de fusion car elle respecte au mieux toutes les contraintes : toutes ces variables de diagnostic valent 1 (cf. chapitre 2).

En considérant l'expression de la question résultant de ce sous-modèle, on constate un problème potentiel : étant exprimée comme un produit, il est possible que l'instanciation de cette question donne une distribution de probabilité dégénérée, nulle partout. Dans le cas spécifique d'un véhicule n'acceptant que des vitesses positives, il est facile d'éviter cette situation dégénérée en garantissant que $P(I_i \mid ([V = 0], \Phi)\ D_i) = 1$ quels que soient Φ et D_i, c'est à dire que $V = 0$ est toujours une commande sûre. Avec une perception omnidirectionnelle, on peut envisager d'accepter aussi des vitesses négatives. Dans ce cas, il n'est pas judicieux de conserver la contrainte précédente, car il peut être intéressant de fuir pour éviter un obstacle trop agressif. La détection d'une distribution de fusion dégénérée est alors un diagnostic d'échec de la fuite, ou de situation de collision inévitable.

Programme

Description

Spécification

Variables Pertinentes

(V, Φ)	:	Commandes recherchées
(V_d, Φ_d)	:	Commandes demandées
I_d	:	Variable de diagnostic associée à (V_d, Φ_d)
$D_1 \dots D_8$	:	Distances mesurées
$I_1 \dots I_8$	:	Variable de diagnostic associée aux $D_{i=1\dots8}$

Décomposition

$$P((V, \Phi)\ (V_d, \Phi_d)\ I_d\ D_1 \dots D_8\ I_1 \dots I_8) =$$
$$P(V, \Phi) P(V_d, \Phi_d) P(I_d \mid (V, \Phi)\ (V_d, \Phi_d))$$
$$\prod_{i=1}^{8} P(D_i) P(I_i \mid (V, \Phi)\ D_i)$$

Formes Paramétriques

$P(V, \Phi), P(V_d, \Phi_d), P(D_i)$	:	Uniformes
$P(I_i \mid (V, \Phi)\ D_i)$	:	Questions aux sous-modèles d'évitement d'obstacles
$P(I_d \mid (V, \Phi)\ (V_d, \Phi_d))$	:	Question au sous-modèle de suivi de consignes

Identification :

$P(I_i \mid (V, \Phi)\ D_i)$	:	Construction à partir du modèle cinématique
$P(I_d \mid (V, \Phi)\ (V_d, \Phi_d))$	:	*a priori*

Question :

$$P((V, \Phi) \mid (V_d, \Phi_d)\ D_1 \dots D_8\ [I_i = 1]_{i=1\dots8}\ [I_d = 1]) \propto$$
$$P([I_d = 1] \mid (V, \Phi)\ (V_d, \Phi_d)) \prod_{i=1..8} P([I_i = 1] \mid (V, \Phi)\ D_i)$$

FIG. 6.12: Fusion de sous-modèle par diagnostic

c) Résultats

Les figures 6.14 à 6.17 illustrent la construction progressive de la distribution de probabilité sur les commandes à appliquer. Dans chaque figure, on trouve à gauche, la distribution sur les commandes proposée par le sous-modèle, et à droite, celle résultant de la fusion du modèle courant avec tous les modèles précédents.

Pour cet exemple, on se place dans les conditions suivantes :

– Les commandes désirées correspondent à une vitesse V_d moyenne et un léger angle de braquage Φ_d vers la gauche.
– Aucun obstacle n'a été détecté en D_1, D_3, D_5, D_7 et D_8.
– D_2 correspond à un obstacle lointain, D_4 à un obstacle modérément proche et D_6 à un obstacle proche.

La figure 6.13 illustre cette situation.

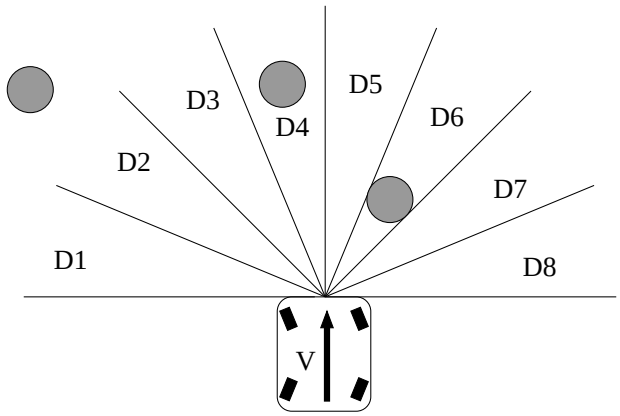


FIG. 6.13: Résultats de l'évitement d'obstacles : situation expérimentale.

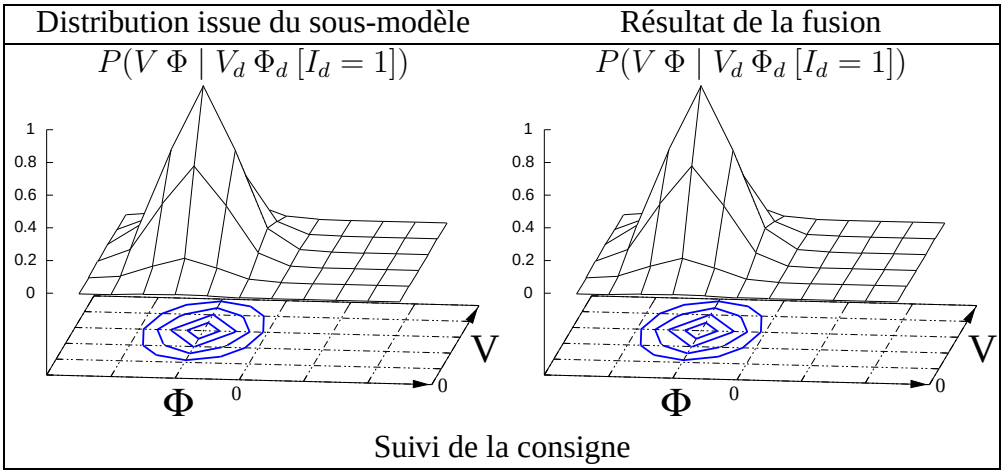


FIG. 6.14: Calcul progressif de $P((V, \Phi) \mid (V_d, \Phi_d)\ D_1 \ldots D_8\ [I_i = 1]_{i=1\ldots8}\ [I_d = 1])$, étape 1.

1. La figure 6.14 présente l'état initial du processus de fusion. A gauche, le sous-modèle de suivi des consignes propose une distribution en cloche, centrée sur (V_d, Φ_d). A droite, puisqu'il s'agit du premier modèle considéré, sa distribution se retrouve à l'identique dans la distribution de fusion.
2. Dans la figure 6.15, on constate le peu d'influence du modèle d'évitement connaissant D_2. A gauche, la distribution de commande sure proposée par le modèle autorise presque toutes les commandes sauf un léger angle de braquage à droite et une vitesse maximale. Comme

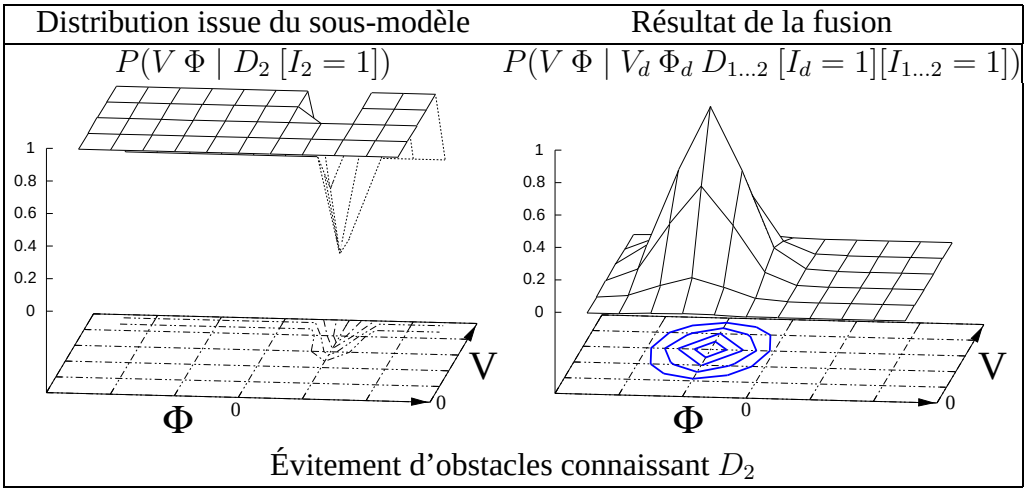


FIG. 6.15: Calcul progressif de $P((V, \Phi) \mid (V_d, \Phi_d)\ D_1 \ldots D_8\ [I_i = 1]_{i=1...8}\ [I_d = 1])$, étape 2.

ces restrictions ne s'opposent pas aux commandes désirées, la distribution de fusion n'est presque pas modifiée par la perception D_2. Ceci est particulièrement visible dans la non modification des lignes de niveaux en bleu sous la nappe de probabilité.

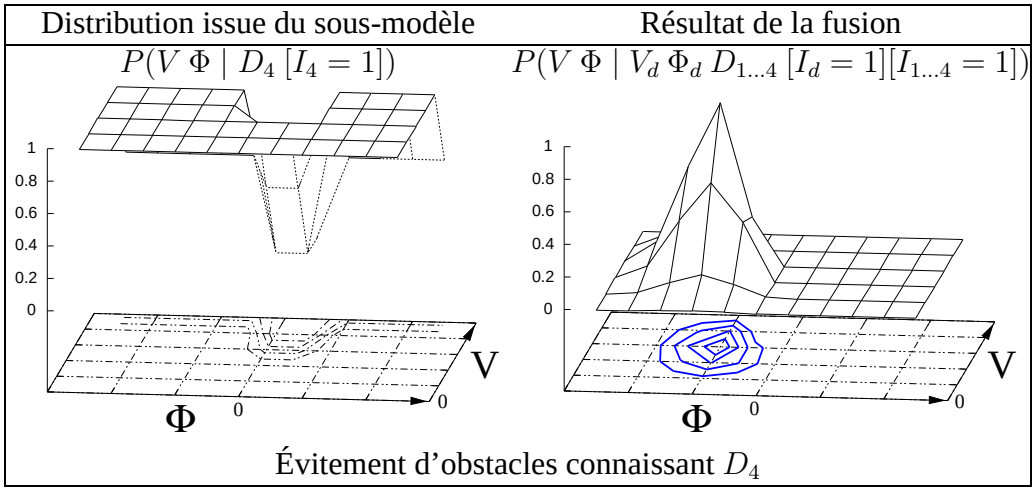


FIG. 6.16: Calcul progressif de $P((V, \Phi) \mid (V_d, \Phi_d)\ D_1 \ldots D_8\ [I_i = 1]_{i=1...8}\ [I_d = 1])$, étape 3.

3. Par rapport à la perception D_2, la perception D_4 a plus d'influence sur la distribution de fusion. Dans la figure 6.16, on constate que le sous-modèle correspondant à D_4 interdit une plus grande partie de l'espace moteur. Globalement, les commandes interdites correspondent à des mouvements en ligne droite avec une vitesse moyenne à grande. En observant les lignes de niveaux de la distribution de fusion, on constate que l'intégration de la distribution proposée par le sous-modèle déforme légèrement la distribution de fusion. Toutefois, le maximum de la distribution n'est pas déplacé, ce qui signifie que ce sous-modèle ne s'oppose pas aux commandes désirées.

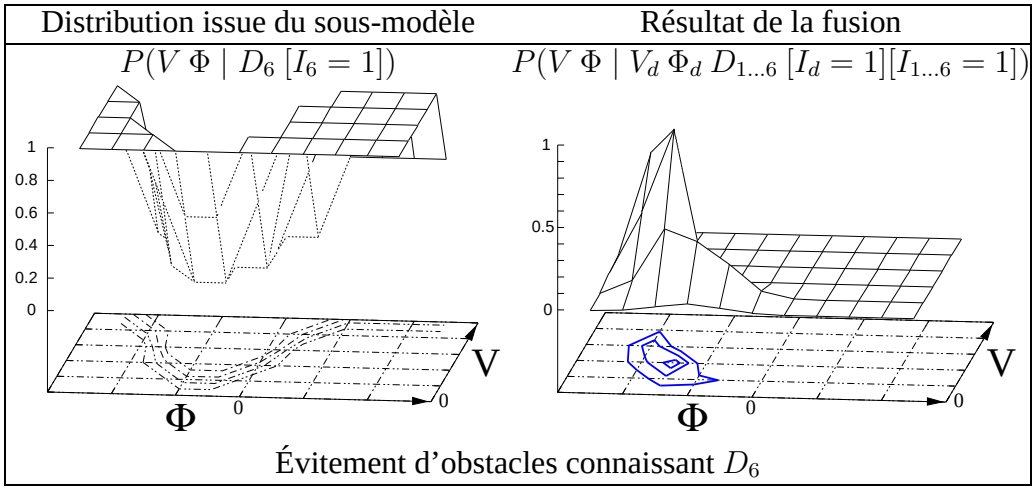


FIG. 6.17: Calcul progressif de $P((V, \Phi) \mid (V_d, \Phi_d)\ D_1 \ldots D_8\ [I_i = 1]_{i=1\ldots8}\ [I_d = 1])$, étape 4.

4. Finalement, la figure 6.17 illustre l'influence d'un obstacle proche (distance D_6 petite). En conséquence, le sous-modèle correspondant à D_6 propose une distribution de commande où les seules commandes possibles correspondent à un braquage maximal à gauche avec une très faible vitesse, ou bien à un braquage à droite avec une vitesse quelconque. Les restrictions imposées par ce modèle sont en opposition avec les commandes désirées. On constate donc que les lignes de niveaux sont fortement déformées sur la distribution de fusion. La commande correspondant au maximum de probabilité correspond à la commande la plus proche de la commande désirée et compatible avec les contraintes de sécurité.

6.4.6 Bayésien, prescriptif ou proscriptif : quelles différences ?

Après avoir décrit ces deux modèles bayésiens de comportement d'évitement d'obstacles, on arrive nécessairement aux deux questions suivantes : quelle différence y a-t-il entre le modèle proscriptif et le modèle prescriptif ? En quoi est-il intéressant d'utiliser un modèle bayésien pour l'évitement d'obstacles ?

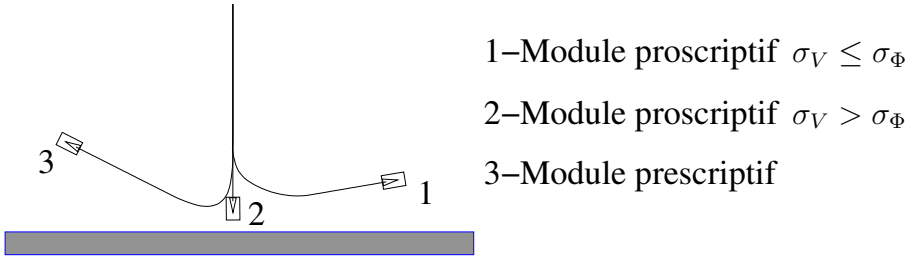


FIG. 6.18: Comparaison des trajectoires d'évitement d'obstacles.

Commençons par la différence entre prescriptif et proscriptif. La figure 6.18 donne un aperçu qualitatif de cette différence. Dans chaque trajectoire, les consignes demandées sont une vitesse

maximale et un angle de braquage nul. Avec une approche proscriptive, s'il est plus important de suivre la vitesse que de respecter l'angle de braquage (trajectoire 1), la trajectoire est déformée avec une courbure plus faible mais plus tôt que dans le cas prescriptif (trajectoire 3), afin de ne pas réduire la vitesse. Par ailleurs, dans le cas proscriptif, on constate que si on choisit de privilégier l'angle de braquage (trajectoire 2), le système ne déforme plus la trajectoire et, ainsi, s'arrête devant l'obstacle sans avoir dévier de sa trajectoire initiale.

Revenons maintenant à l'intérêt d'utiliser une formulation bayésienne du problème de l'évitement d'obstacles. Plusieurs points peuvent être évoqués pour justifier cette formulation :

- Tout d'abord, la simplicité et la clarté de la décomposition en sous-problèmes simples dont les résultats sont ensuite mis en commun,
- Ensuite, la simplicité de l'expression des niveaux de contrainte et la richesse de cette expression : toute déformation de la nappe de probabilité associée à un sous-modèle est synonyme d'information sur les contraintes à respecter, ou sur les libertés dont dispose le système.
- Notons aussi la richesse de la liaison avec le pilote de haut-niveau : on peut imaginer un pilote de haut niveau qui demande des commandes exotiques du type : tourner à gauche, exactement à 1m/s ou légèrement à droite sans dépasser 0.5m/s. Ce type de consigne s'exprime facilement comme distribution de probabilité (avec ou sans variable de diagnostic), et s'intègre non moins facilement dans la fusion globale des sous-modèles.
- Finalement, une expression sous forme d'inférence bayésienne s'intègre harmonieusement dans une architecture de contrôle bayésienne telle que celles présentées au chapitre 7.

6.5 Conclusion

Ce chapitre nous a permis d'exprimer deux problèmes classiques de la robotique sous la forme de problèmes d'inférences bayésiennes : le suivi de trajectoire et l'évitement d'obstacles. L'intérêt de ces expressions est leur sémantique à la fois riche et précise.

Cette richesse a été illustrée non seulement par l'expression de niveaux de contrainte variables, aussi bien dans le suivi de trajectoire que dans l'évitement d'obstacles, mais aussi par les notions de programmation prescriptive et proscriptive.

La précision résulte quant à elle de l'utilisation de la théorie des probabilités et du formalisme de la programmation bayésienne. Grâce à cela, toutes nos expressions ont une base mathématique solide et une sémantique bien définie.

Arrivés a la fin de ce chapitre, nous avons décrit un jeu de fonctionnalités indépendantes pouvant être utilisées comme briques élémentaires dans la construction d'une application complexe de navigation intentionnelle. Dans le dernier chapitre de cette thèse, nous allons maintenant nous préoccuper de l'implantation et de l'intégration de toutes ces facultés sur notre plate-forme expérimentale : le CyCab.

Chapitre 7

Plusieurs niveaux de représentation pour une tâche complexe

7.1 Programmation Bayésienne Orientée Objet

7.1.1 Insuffisance de la programmation bayésienne

La programmation bayésienne à été définie par Bessière *[Bessière et al. 1998a;b]* et utilisée dans de nombreux contextes depuis : *[Coué 2003, Diard 2003a, Lebeltel et al. 2003, Mekhnacha 1999b, Ramirez 2003]*. Ces principes ont été présentés dans le chapitre 2. Un programme bayésien décrit un problème en identifiant les variables pertinentes, en donnant une décomposition de la distribution conjointe sur ces variables et en identifiant des distributions impliquées dans cette décomposition, éventuellement à l'aide d'autres programmes bayésiens.

a) Problème considéré

La notion de programme bayésien est tout à fait adaptée pour modéliser un problème de façon formelle. De plus, *[Bessière & Group 2003]* ont montré que la capacité d'expression de cette notion lui permettait d'exprimer la plupart des autres approches probabilistes utilisées en robotique. **Malgré toutes ces qualités, deux problèmes sont apparus au moment de la conception logicielle de nos applications : un manque de généricité et l'absence de la notion d'abstraction.**

b) Illustration

Pour illustrer ces notions, prenons un exemple simple inspiré d'une situation classique en robotique : un robot ponctuel, caractérisé par son abscisse X. Ce robot est mobile et peut recevoir des ordres U qui seront calculés par rapport à X. Pour estimer X, le robot dispose de deux capteurs qui lui fournissent deux données D_1 et D_2, et d'un modèle de ces capteurs qui lie X à D_1 en utilisant un paramètre P_1 et X à D_2, en utilisant P_2. Le programme bayésien qui modélise le système est donné dans la figure 7.1. Si l'on souhaite utiliser une approche de programmation

plus modulaire, on peut utiliser le programme bayésien présenté par la figure 7.5, construit à partir des programmes 7.2, 7.3 et 7.4.

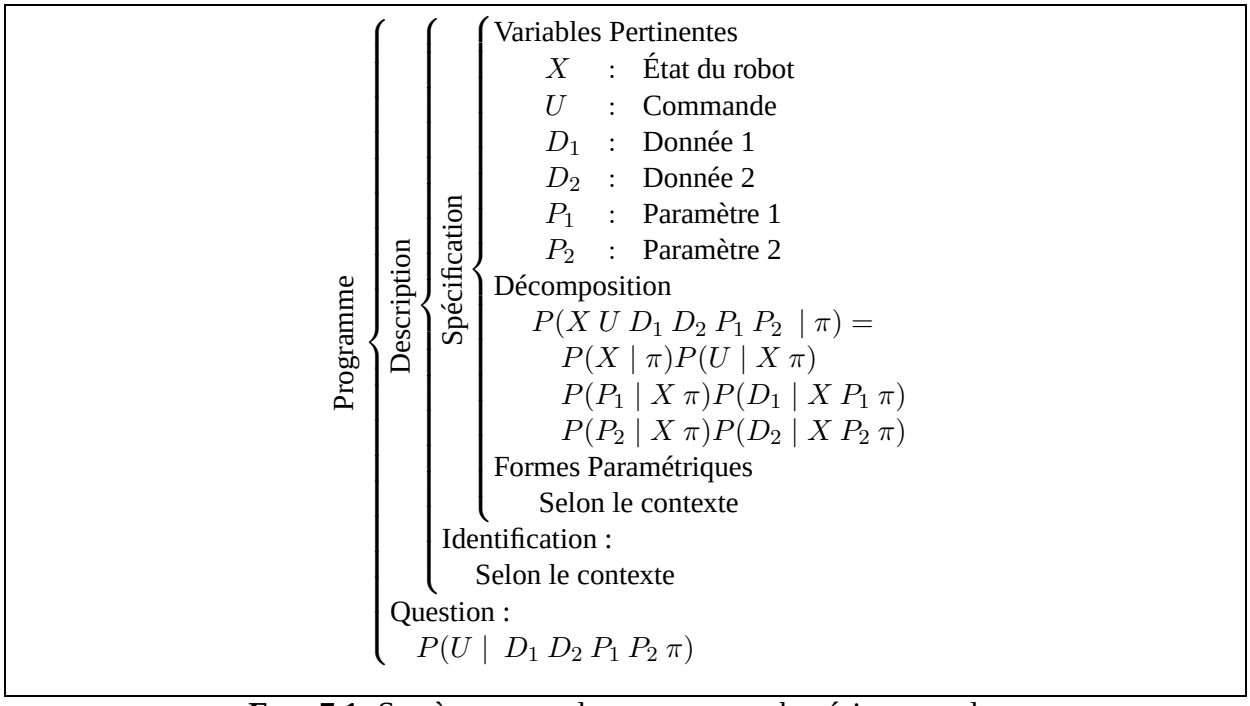


FIG. 7.1: Système exemple : programme bayésien complet

c) Analyse

*1.***Le manque de généricité** apparaît dans la redondance des définitions des modèles des capteurs 1 et 2. Comme ces programmes ont la même structure, on voudrait pouvoir décrire un méta-programme bayésien *capteur* qui serait instancié deux fois dans cet exemple.

*2.***L'absence d'abstraction** est illustré par la version modulaire du programme bayésien : les paramètres P_1 et P_2 sont des paramètres des modèles capteurs et il n'y a que peu d'intérêt à les faire apparaître dans le programme bayésien complet. De plus, pour ce qui est de la localisation, tout ce qu'on demande à un modèle capteur est d'être capable de lier une observation du système à un état du système. On souhaiterait donc ne voir apparaître dans le modèle global que la distribution $P(D_i \mid X\ \pi)$. Ainsi, les autres paramètres du modèle capteur devraient être oubliés à ce niveau d'abstraction. Un autre intérêt de ce type d'abstraction est la flexibilité qu'elle introduit : si on désire changer de modèle capteur, et prendre maintenant un modèle capteur qui dépend de trois paramètres, il n'est pas nécessaire de changer le modèle global, car celui-ci n'utilise que l'interface abstraite du modèle capteur, c'est à dire $P(D_i \mid X\ \pi)$.

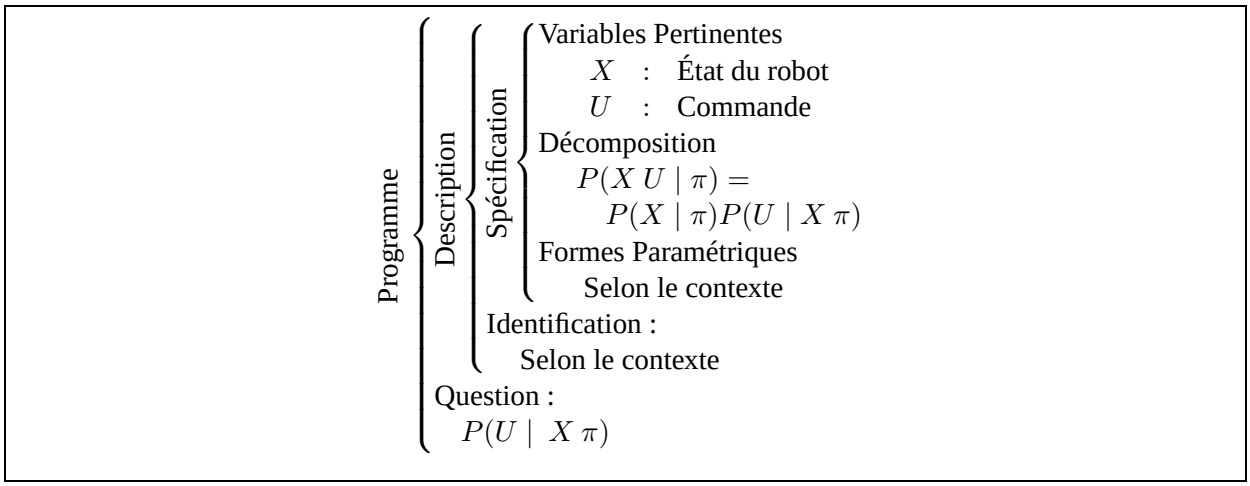


FIG. 7.2: Système exemple : programme bayésien de contrôle

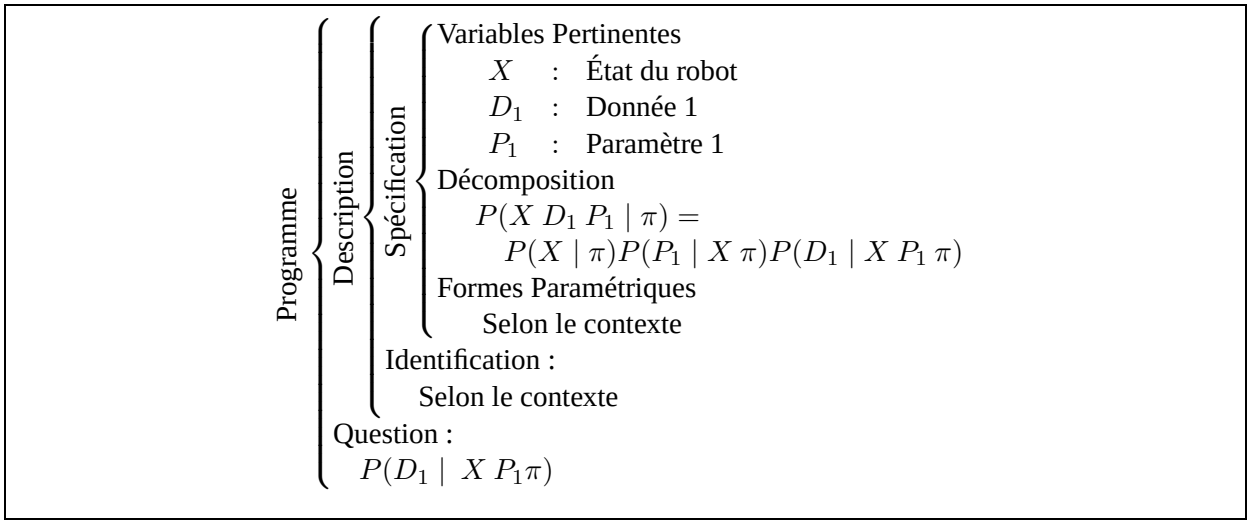


FIG. 7.3: Système exemple : programme bayésien du capteur 1

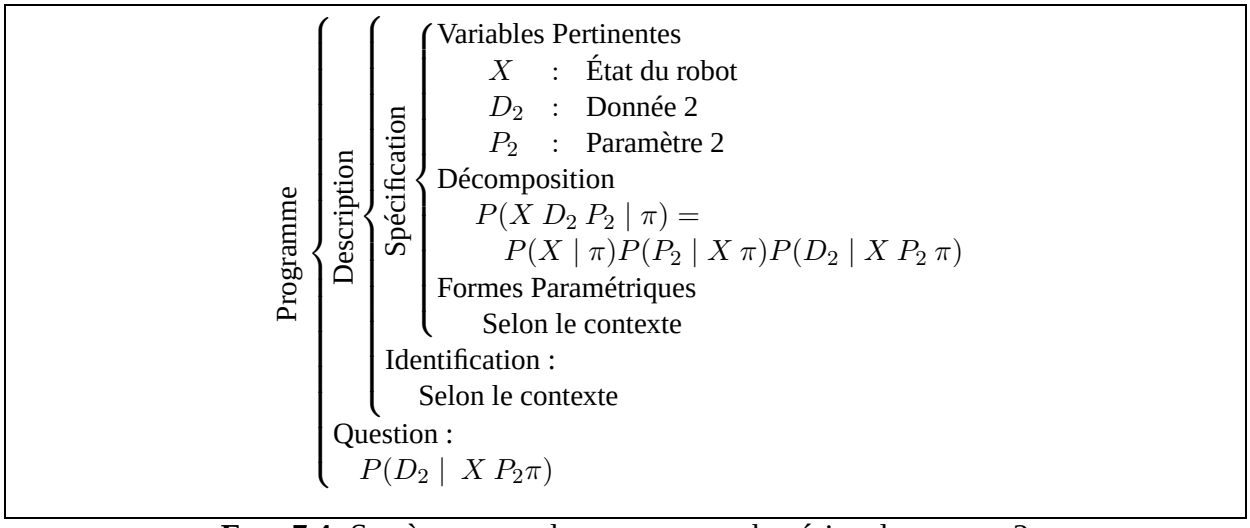


FIG. 7.4: Système exemple : programme bayésien du capteur 2

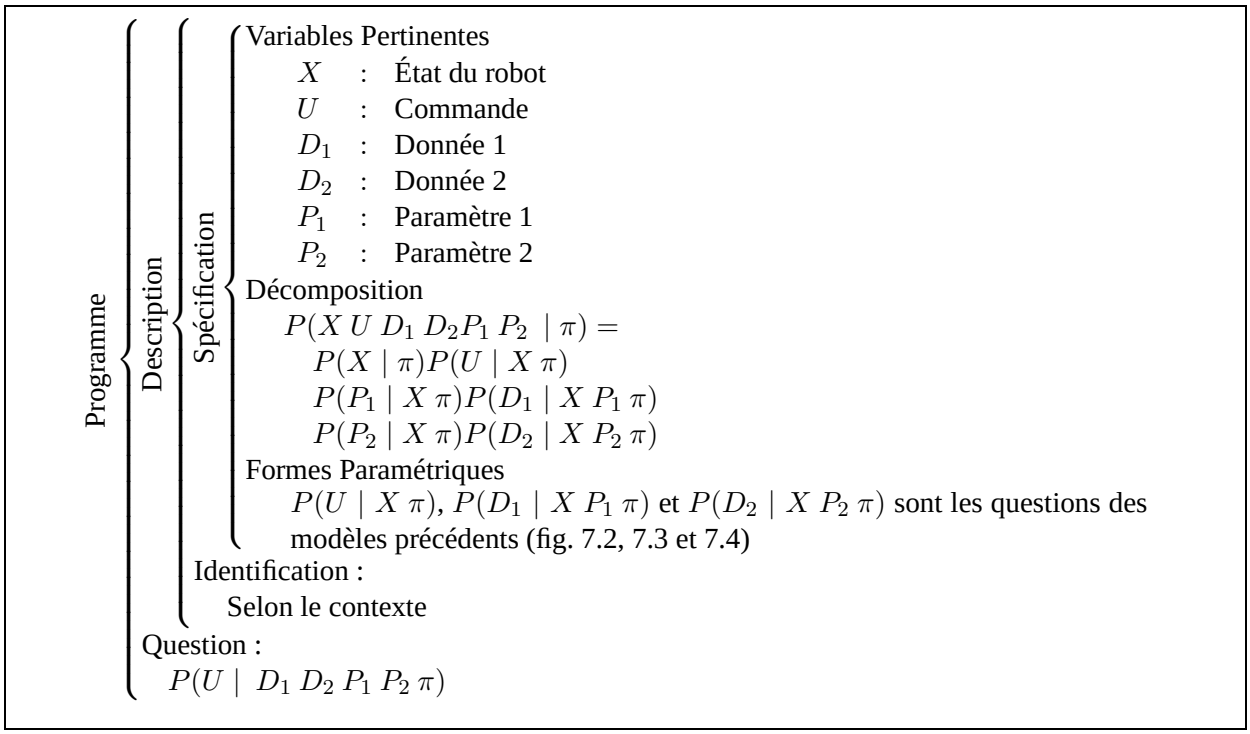


FIG. 7.5: Système exemple : programme bayésien complet modulaire

7.1.2 La PBOO : une méthodologie de modélisation bayésienne

Définition 26 – Classe bayésienne *On appelle classe bayésienne $\mathcal{C}$ un septuplé*

$$\mathcal{C} = \{L, R, T, K, \{C_i\}_{i=1..n}, \Pi, \pi\}$$

où L,R,T,K sont des variables aléatoires. Π est un programme bayésien associé à la classe : ses variables pertinentes sont L, R, T, K et sa question est $P(L \mid R\, K\, \pi)$. π représente l'ensemble des connaissances préalables utilisées par le programmeur pour définir $\mathcal{C}$. $\{C_i\}_{i=1..n}$ est une collection de classe bayésienne, éventuellement vide. Les classes de $\{C_i\}_{i=1..n}$ sont utilisées pour construire la distribution conjointe de Π.

La sémantique des variables est la suivante :

- *L : variable aléatoire sur laquelle $\mathcal{C}$ exporte une distribution, doit nécessairement exister.*
- *R : variable aléatoire dont dépend la distribution exportée. Si R n'existe pas, la distribution exportée est inconditionnelle, elle est conditionnelle dans le cas contraire.*
- *T : variable aléatoire de transfert (variable locale). Permet de transférer de l'information d'un objet probabiliste à un autre, ou de générer des marginalisations.*
- *K : variable aléatoire dont la valeur est connue/accessible par $\mathcal{C}$. K peut être une mesure venant d'un capteur, une décision sur la distribution d'une classe interne, un paramètre d'un modèle...*

Une classe bayésienne est utilisée pour exporter la distribution $P(L \mid R\, \pi)$.

Pour exporter sa distribution, la classe bayésienne $\mathcal{C} = \{L, R, T, K, \{C_i\}_{i=1..n}, \Pi, \pi\}$ utilise Π pour calculer

$$P(L \mid RK\pi) = \sum_T P(L \mid RKT\pi) P(T \mid \pi)$$

Comme, par définition, K est une variable dont la classe connaît la valeur k, on peut poser $\pi' = \pi \wedge [K = k]$. On obtient alors :

$$P(L \mid R\pi') = P(L \mid RK\pi)$$

Les variables K et T disparaissent de la distribution exportée. L'exportation de la distribution ajoute donc un niveau d'abstraction à la modélisation.

a) Réécriture du système exemple en PBOO

Nous allons maintenant utiliser le formalisme PBOO pour reformuler le système utilisé comme exemple au début de cette section. Les programmes bayésiens associés à ces classes sont tirés directement des figures 7.1 à 7.5.

Le premier modèle dont nous disposons est un modèle de capteur qui calcule $P(D \mid X)$. On en déduit une classe bayésienne `Capteur` (table 7.1.a). A partir de ce modèle, on peut définir une classe bayésienne de localisation `Localisation` (table 7.1.b) qui utilise deux instances du modèle capteur. On ajoute ensuite une classe de contrôle `Contrôle` (table 7.1.c) On peut finalement définir notre système global `Robot` (table 7.1.d) en utilisant une instance de la classe de localisation et une instance de la classe de contrôle. La figure 7.6 illustre cette architecture.

Classe (a) **Capteur**
Exporte P(D : donnée $\mid$ X : état π) Objets internes : $\emptyset$
Variables : L = D : donnée R = X : état T = $\emptyset$ K = P : paramètre

Classe (b) **Localisation**
Exporte P(X : état $\mid$ π) Objets internes : Capteur$_1(D_1 \mid X)$ Capteur$_2(D_2 \mid X)$
Variables : L = X : état R = $\emptyset$ T = $\emptyset$ K = D_1 D_2 : variables connues

Classe (c) **Contrôle**
Exporte P(U : commande $\mid$ X : état π) Objets internes : $\emptyset$
Variables : L = U : commande R = X : état T = $\emptyset$ K = $\emptyset$

Classe (d) **Robot**
Exporte P(U : commande $\mid$ π) Objets internes : Contrôle$(U \mid X)$ Localisation(X)
Variables : L = U : commande R = $\emptyset$ T = X : état K = $\emptyset$

TAB. 7.1 – Expression du système exemple sous forme de classes bayésiennes.

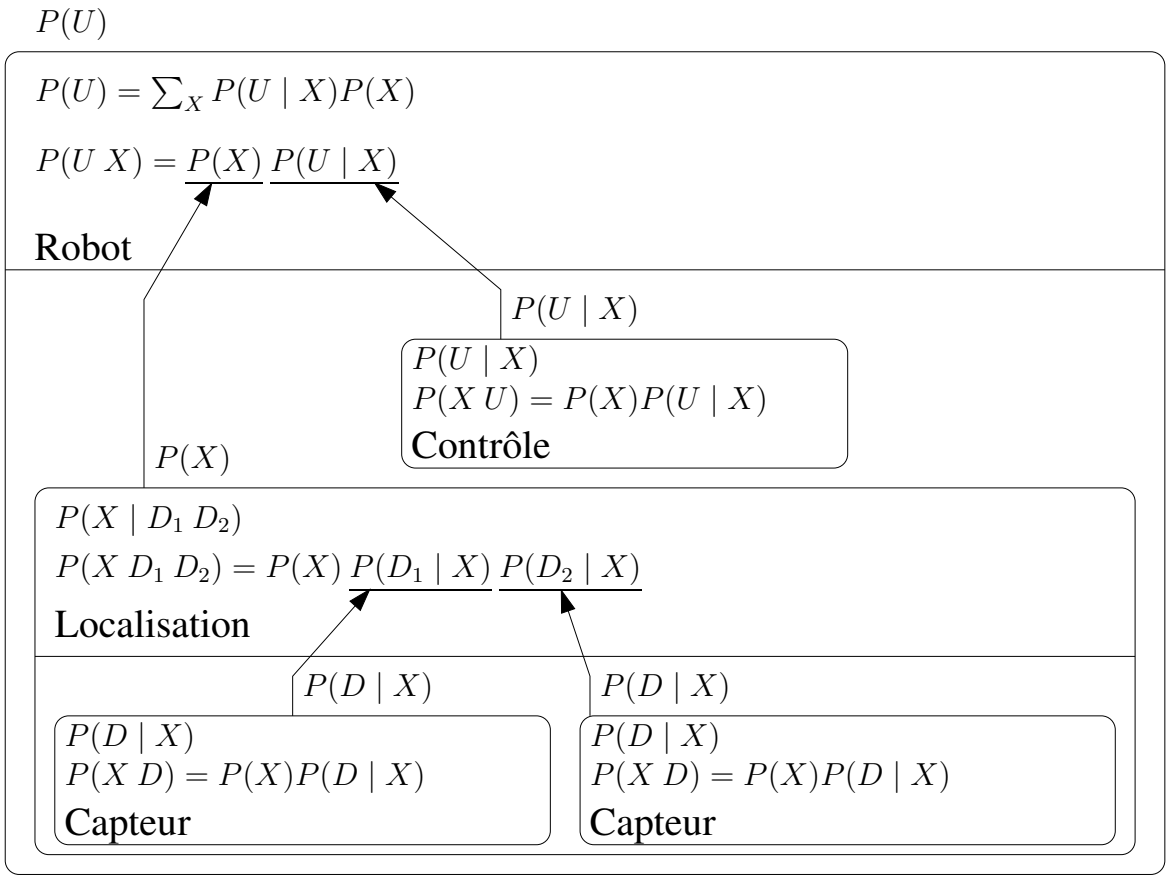


FIG. 7.6: Architecture de contrôle définie en PBOO.

L'utilisation, dans l'application précédente, de la position comme variable de transfert, provoque, lors du calcul de $P(U)$, une sommation sur X. Pour éviter le coût calculatoire de cette sommation, il est possible de prendre une décision sur la distribution $P(X)$. La variable X devient alors une variable connue de l'objet Robot2 (table 7.2) .

Classe **Robot2**
Exporte P(U : commande $\mid \pi$) Objets internes : Contrôle$(U \mid X)$ Localisation(X)
Variables : L = U : commande R = $\emptyset$ T = $\emptyset$ K = X : état

TAB. 7.2 – Classe bayésienne : Robot2

b) En résumé

Une classe bayésienne est donc une structure qui vient s'ajouter sur un programme bayésien pour définir une interface qui pourra être utilisée par d'autres modèles. L'implantation particulière de la classe bayésienne sera un programme probabiliste particulier. Pour le module utilisateur, la seule partie utile de la classe bayésienne est la partie exportée (ou publique), c'est à dire le couple (L, R). Il pourra donc utiliser indifféremment toute classe exportant $P(L \mid R)$ indépendamment de l'implantation réelle de la classe : $(\{C_i\}_{i=1..n}, T, K, \Pi, \pi)$.

Finalement la programmation bayésienne orientée objet (PBOO) est un formalisme efficace pour définir une architecture de décision bayésienne contenant plusieurs niveaux d'abstraction. Sa généricité permet de changer une partie de l'architecture (pour changer de méthode de calcul ou changer de modèle par exemple) sans avoir à modifier le reste. Par ailleurs, la notion de classe bayésienne s'intègre naturellement dans un langage orienté objet (type C++ ou Java), ce qui permet de bénéficier de la notion d'héritage. De la notion d'héritage de classes (C++), on arrive donc à la notion d'héritage de modèles. Un modèle hérité d'un autre propose la même interface – c'est à dire les mêmes variables L et R – mais peut réaliser son inférence bayésienne de façon différente : en utilisant une autre méthode algorithmique, d'autre paramètres, etc.

La PBOO nous servira de guide pour décrire les architectures intégrées qui seront présentées dans la suite de ce chapitre.

7.2 Classes bayésiennes implémentées dans les chapitres précédents

Cette section nous permet de reprendre l'ensemble des fonctionnalités décrites dans nos chapitres précédents pour les ré-exprimer sous formes de classes bayésiennes.

7.2.1 Modélisation de l'environnement

Classe (a) **SLAM classique**
Exporte P($(X_t, E_t) \mid \pi$) Objets internes : $\emptyset$
Variables : $L = (X_t, E_t)$ $R = \emptyset$ $T = \emptyset$ $K = Z_t \dots Z_0$

Classe (b) **Grille d'occupation**
Exporte P($G_t \mid \pi$) Objets internes : $\emptyset$
Variables : $L = G_t$ $R = \emptyset$ $T = \emptyset$ $K = Z_t \dots Z_0$

Classe (c) **Filtre relatif**
Exporte P($Inv(E_t) \mid \pi$) Objets internes : $\emptyset$
Variables : $L = Inv(E_t)$ $R = \emptyset$ $T = \emptyset$ $K = Z_t \dots Z_0$

Classe (d) **FPG**
Exporte P($E_t \mid \pi$) Objets internes : Filtre relatif
Variables : $L = E_t$ $R = \emptyset$ $T = \emptyset$ $K = Inv(E_t)$

TAB. 7.3 – Classes bayésiennes de modélisation de l'environnement.

*1.***SLAM classique** L'algorithme classique de SLAM(sec. 5.2.2, table 7.3.a) estime conjointement l'état du robot (X_t) et de son environnement (E_t) au cours du temps à partir des observations $Z_t \dots Z_0$. L'état de l'environnement est résumé par l'état d'un ensemble de primitives simples (segments, points,...). La distribution exportée prend la forme d'une gaussienne multidimensionnelle résumée par un vecteur moyen et une matrice de covariance.

*2.***Grille d'occupation** Le principe d'une grille d'occupation (sec. 5.2.1, table 7.3) est d'exporter une estimation de l'état de l'environnement. Pour un point donné de l'environnement, cette représentation donne la probabilité que ce point fasse partie de l'espace libre ou des obstacles. Il s'agit donc d'une représentation appropriée pour planifier une trajectoire sûre.

*3.***Filtre relatif** Le filtre relatif (sec. 5.2.2, table 7.3.c) exporte une représentation de l'environnement composée de propriétés invariantes ($Inv(E_t)$), et donc indépendante de l'état du robot. Une des spécificités du filtre relatif est son opposition systématique aux corrélations. En effet, la distribution exportée sur les propriétés invariantes est une distribution gaussienne sans aucune corrélation (sa matrice de covariance est diagonale).

*4.***Filtre à projection géométrique** Le filtre à projection géométrique (sec. 5.2.2, table 7.3.d) utilise le filtre relatif pour construire une représentation de l'environnement utilisable pour la localisation.

7.2.2 Localisation

Classe (a) **Triangulation**
Exporte P($X_t \mid \vec{Z}_t\ \pi$) Objets internes : $\emptyset$
Variables : L = X_t R = $\vec{Z}_t$ T = $\emptyset$ K = $E_t, \mathcal{I}$

Classe (b) **Localisation sans mise en correspondance**
Exporte P($X_t \mid \vec{Z}_t\ \pi$) Objets internes : Capteur($Z_i \mid X$)
Variables : L = X_t R = $\vec{Z}_t$ T = $\vec{F}, \vec{W}$ K = E_t

Classe (c) **Loc. temporelle**
Exporte P($\tau(t), \xi(t) \mid Z_t \ldots Z_0\ \pi$) Objets internes : Capteur($Z_t \mid \xi_t$)
Variables : L = $\tau(t), \xi(t)$ R = $Z_t \ldots Z_0$ T = $\tau(t-1), \xi(t-1)$ K = $\mathcal{T}_{sm}$

Classe (d) **Estimation de confiance**
Exporte P(Cert(t) $\mid Z_t\ \pi$) Objets internes : Fausse observation(Z_t) Observation sans modèle($Z_t \mid Z_{t-1}\ \xi$) Observation avec modèle($Z_t \mid E_t$)
Variables : L = Cert(t) R = Z_t T = Cert($t-1$), *Modèle* K = Z_{t-1}, ξ, E_t

TAB. 7.4 – Classes bayésiennes de localisation.

Quatre classes bayésiennes ont été présentées dans le chapitre 4 sur la localisation.

*1.***Triangulation** La localisation par triangulation (sec. 4.2, table 7.4.a) utilise un filtre de Kalman et un mécanisme déterministe de mise en correspondance robuste ($\mathcal{I}$) pour estimer la position du robot X_t à partir de l'observation Z_t.

2.**Localisation sans mise en correspondance** Pour éviter d'utiliser une mise en correspondance déterministe, nous avons utilisé un mécanisme de localisation probabiliste avec mise en correspondance implicite(sec. 4.3, table 7.4.b). Cette localisation estime elle aussi l'état du robot X_t en fonction des observations. Cependant, la mise en correspondance, représentée par les variables $\vec{F}$ et $\vec{W}$ n'est pas une variable connue (comme $\mathcal{I}$ en 7.4.a) mais une variable de transfert.

3.**Localisation temporelle** Aux classes de localisation spatiale, nous avons ajouté une classe bayésienne de localisation temporelle (sec. 4.4, table 7.4.c) qui estime la position temporelle $\tau(t)$ et l'erreur de suivi $\xi(t)$ dans une trajectoire sensorimotrice $\mathcal{T}_{sm}$ à partir de l'observation Z_t.

4.**Estimation de confiance** Nous avons aussi ajouté une classe bayésienne (sec. 4.5, table 7.4.d) qui estime au cours du temps un indice de confiance fondé sur la cohérence entre les observations Z_t réalisées et celles attendues connaissant une distribution de probabilité sur la position spatio-temporelle $(X_t, \tau(t))$. Cette classe utilise trois objets internes pour prévoir ces observations selon trois hypothèses : on ne dispose d'aucune information, on ne connaît que l'observation précédente et la différence de point de vue et on possède un modèle complet de l'environnement.

7.2.3 Comportements

Quatre classes bayésiennes visant à générer des commandes applicables sur le robot ont été présentées au chapitre 6 sur les comportements.

1.**Évitement d'obstacles** Deux classes se présentent comme une spécialisation d'une classe bayésienne abstraite d'évitement d'obstacles : les évitements d'obstacles proscriptifs (sec. 6.4.5, table 7.5.a) et prescriptifs (sec. 6.4.4, table 7.5.b). Ces classes calculent une distribution sur les commandes à appliquer $U = [V, \Phi]$, connaissant des mesures de distance venant de proximètres ($\{D_i\}_{i=1..8}$) et une commande demandée ($U_d = [V_d, \Phi_d]$) par un niveau d'abstraction supérieur.

2.**Réalisation d'un trajet** Afin de suivre un trajet, nous avons développé un comportement de suivi de trajectoire exprimé sous forme d'inférence bayésienne (sec. 6.3, table 7.5.c). Le suivi bayésien calcule une distribution sur les commandes U connaissant l'erreur de suivi ξ_t et les commandes de référence U_r.

7.2.4 Conclusions

En utilisant le formalisme de la PBOO, on peut décrire de façon formelle les briques de base dont nous disposons pour construire des applications complexes. L'expression "application complexe" est utilisée ici pour désigner une application réalisant l'intégration d'un ensemble de briques de base à travers plusieurs niveaux d'abstraction.

Le but des sections qui vont suivre est de présenter deux applications complexes réalisées à partir de notre bibliothèque de classes bayésiennes.

Classe (a) **Évitement d'obstacles proscriptif**
Exporte P($U \mid U_d\ \pi$)
Objets internes :
$8 \times$ Evit. élémentaire$(I_i \mid U\ D_i)$
Suivi des consignes$(I_c \mid U\ U_d)$
Variables :
L = U
R = U_d
T = $\emptyset$
K = Var. de diagnostic I_i
Proximètres

Classe (b) **Évitement d'obstacles prescriptif**
Exporte P($U \mid U_d\ \pi$)
Objets internes :
$8 \times$ Evit. élémentaire$(U \mid U_d)$
Variables :
L = U
R = U_d
T = S
K = Proximètres

Classe (c) **Suivi bayésien**
Exporte P($U = [V, \Phi] \mid \xi_t = [\delta x_t, \delta y_t, \delta\theta_t]$ π)
Objets internes :
Contrôle longitudinal$(V \mid \delta x_t\ V_r)$
Contrôle latéral$(\Phi \mid \delta y_t\ \delta\theta_t\ \Phi_r)$
Variables :
L = $U = [V, \Phi]$
R = $\xi_t = [\delta x_t, \delta y_t, \delta\theta_t]$
T = $\emptyset$
K = U_{ref}

TAB. 7.5 – Classes bayésiennes de contrôle.

7.3 Plate-forme expérimentale

7.3.1 Un véhicule autonome : le CyCab

La plupart des problématiques présentées dans ce travail sont inspirées des difficultés rencontrées en travaillant sur la plate-forme de l'INRIA Rhône-Alpes. Cette plate-forme (fig. 7.7) est un véhicule électrique doté de la capacité originale de pouvoir braquer indépendamment ces deux essieux. Le modèle général de véhicule qui en résulte est présenté sur le schéma 7.8. Cependant, pour utiliser un modèle simplifié tout en maximisant la manoeuvrabilité, nous avons choisi d'utiliser une amplitude de braquage identique à l'avant et à l'arrière. Dans ce cas, le modèle cinématique résultant est celui d'une voiture dont la distance inter-essieux serait égale à la moitié de la distance réelle, ce qui implique un rayon de braquage minimal.

Le CyCab est équipé d'un système informatique complexe et distribué, constitué de deux micro-contrôleurs et d'un ordinateur embarqué. Chaque micro-contrôleur est responsable de l'angle de braquage et de la vitesse appliquée sur un essieu. L'ordinateur embarqué supervise le système global en communiquant avec les micro-contrôleurs par l'intermédiaire d'un bus CAN.

Cette architecture présente l'avantage de pouvoir définir plusieurs niveaux de sécurité indé-

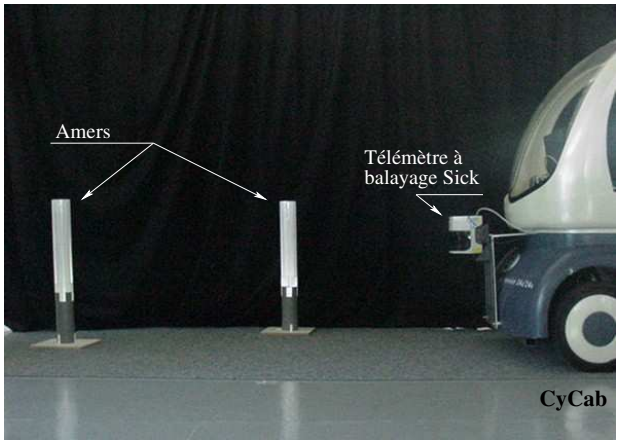
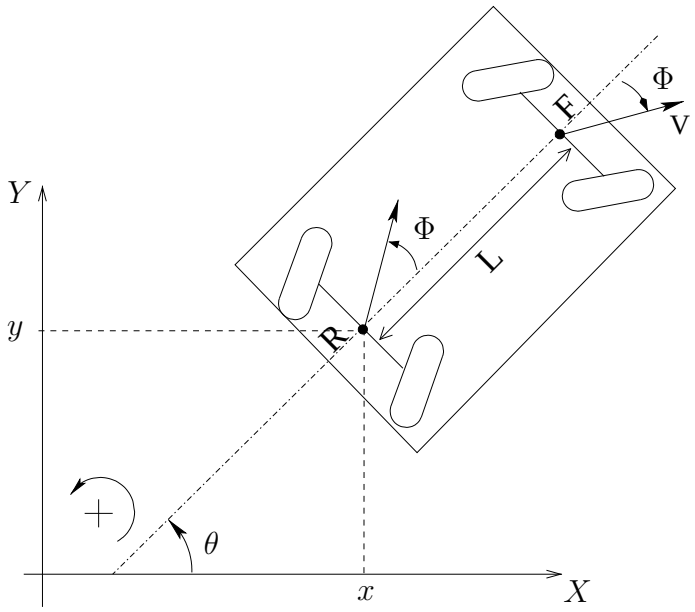


FIG. 7.7: Le CyCab, son télémètre laser et deux amers.



$$\begin{aligned} \dot{x} &= V\cos(\theta+\Phi) \\ \dot{y} &= V\sin(\theta+\Phi) \\ \dot{\theta} &= \frac{2V\sin(\Phi)}{L} \end{aligned}$$

FIG. 7.8: Modèle cinématique du CyCab.

pendants et redondants. Cependant, elle introduit des retards de communication entre l'émission d'une commande motrice sur l'ordinateur embarqué et son application réelle par les microcontrôleurs.

Finalement, comme le CyCab est un véhicule de $300kg$ pouvant atteindre une vitesse de $20km/h$, les expérimentations sur le parking de l'INRIA, au milieu des véhicules du personnel et éventuellement de spectateurs, ont été parfois particulièrement stressantes. Pour cette raison, un effort particulier a été porté sur la mise en place de systèmes de sécurité fiables et efficaces, et sur le développement d'un simulateur reproduisant les caractéristiques principales du CyCab et de ses capteurs. Ces développements sortent du cadre de cette thèse et ne seront donc pas plus détaillés dans ce document. De plus amples détails devraient cependant être publiés dans un rapport technique de l'INRIA.

7.3.2 Un capteur embarqué : le télémètre laser

Le CyCab que nous utilisons est équipé de capteurs proprioceptifs permettant de mesurer la vitesse et le déplacement de chaque roue ainsi que l'angle de braquage appliqué par chaque moteur de direction.

Nous avons aussi installé un télémètre laser à balayage[1] qui permet de mesurer la distance aux objets les plus proches, tous les demi-degrés, dans un secteur angulaire plan de 180^o, situé à l'avant du CyCab (partie grisée sur la figure 7.9). Pour les besoins des algorithmes de localisation qui seront présentés par la suite, nous avons choisi d'installer des amers artificiels dans notre environnement. Ces amers sont des cylindres recouverts d'une surface réfléchissante particulièrement bien adaptée à notre capteur. Deux amers sont visibles sur la figure 7.9.

Finalement, un algorithme de pré-traitement des données laser a été développé afin de détecter les amers présents dans le champ de vision du capteur. Ainsi, nous disposons de la position de chaque amer observé dans le repère capteur. La plupart des algorithmes présentés dans ce document considèrent donc le télémètre laser comme un détecteur d'amers en ignorant les données brutes. La figure 7.10 donne un exemple de données mesurées par notre capteur sur le parking de l'INRIA. Chaque petit cercle correspond à un écho laser. En haut de l'image, on distingue les silhouettes de quelques véhicules en stationnement. Deux amers sont détectés avec trois échos chacun. La séquence d'échos, en bas au milieu de l'image, correspond à un mur de l'INRIA. La qualité de l'alignement des échos dans cette séquence illustre la précision du capteur et sa fiabilité.

[1] Sick LMS200.

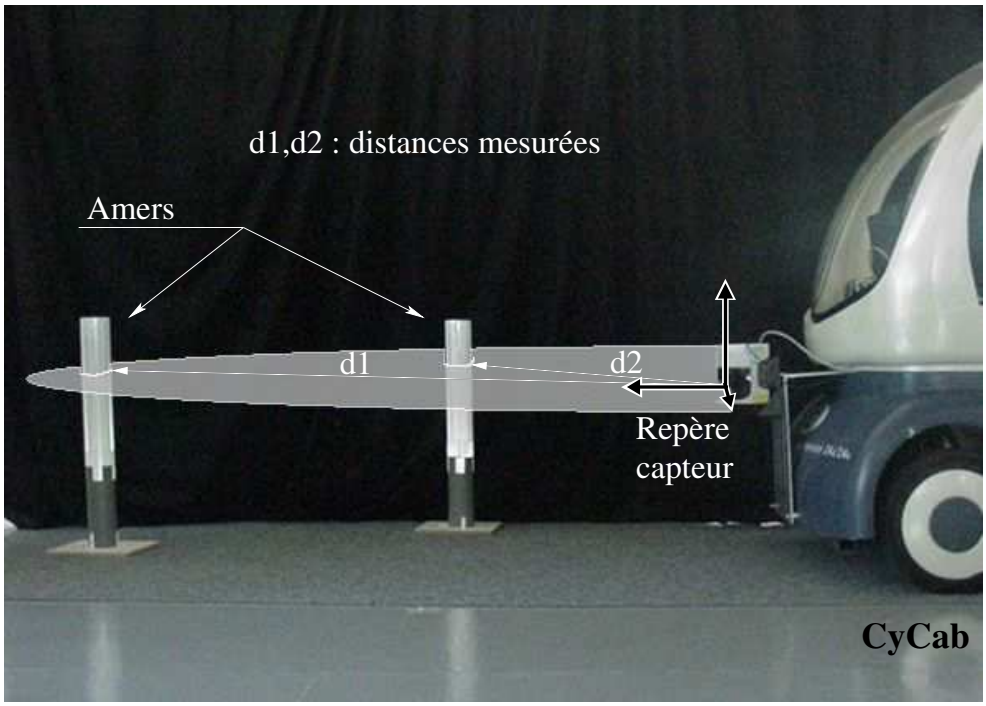


FIG. 7.9: Mesures avec un télémètre laser et deux amers.

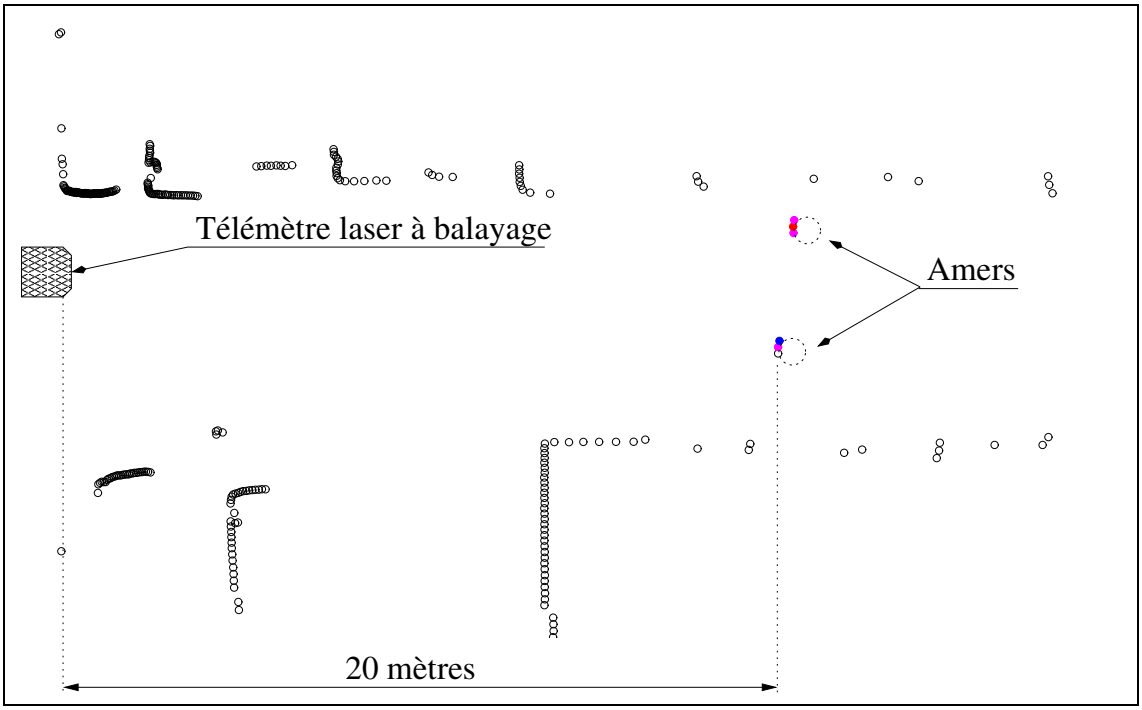


FIG. 7.10: Exemple de résultat renvoyé par le Sick sur le parking de l'INRIA.

7.4 Planification et suivi de trajectoires en environnement non contrôlé

7.4.1 Objectif

L'autonomie d'un véhicule en milieu urbain est une tâche complexe qui mobilise beaucoup de moyens actuellement, tant sur le plan humain que financier. Dans les chapitres précédents, nous avons implanté un ensemble de fonctionnalités qui nous paraissent absolument nécessaires pour augmenter l'autonomie des véhicules automatiques du futur : la construction d'un modèle interne de l'environnement, la localisation dans ce modèle et sa mise à jour au cours du mouvement, la planification d'un déplacement dans ce modèle, la réalisation de cette planification en tenant compte de modifications éventuelles de l'environnement.

7.4.2 Architecture système

Pour réaliser cette application, nous avons intégré un ensemble de cinq classes bayésiennes dans la structure illustrées par la figure 7.11.

7.4.3 Expérimentations

La figure 7.12 présente une situation expérimentale typique.

Scénario :

- Après l'installation rapide et arbitraire d'amers dans l' espace de travail, une première phase de conduite manuelle est exécutée afin de construire un modèle de l'environnement : une carte des amers pour la localisation et une carte d'occupation de l'espace libre pour la planification.
- Cette dernière et la position courante sont ensuite transmises au planificateur de trajectoire.
- Celui-ci, à partir d'une interface graphique, demande une position but à l'utilisateur et calcule, si possible, une trajectoire pour atteindre ce but.
- Après validation de l'utilisateur, la trajectoire est exécutée, et éventuellement déformée par le module d'évitement d'obstacles.

Résultats : On peut observer la réalisation d'une telle trajectoire avec la carte d'occupation de l'espace libre et les trajectoires des piétons sur la figure 7.13. Sur cette figure, la trajectoire planifiée part de la gauche de l'image (position notée "position initiale") et va se garer dans la place disponible en haut à droite de l'image. Pendant l'exécution, un premier piéton coupe la trajectoire en venant de la gauche. La première réaction du système est une manoeuvre d'évitement par la droite. Comme le piéton continue à se déplacer vers la droite, l'écart à la trajectoire devient trop important et une manoeuvre d'évitement par la gauche est appliquée avec succès. L'inverse se produit quand un second piéton coupe la trajectoire.

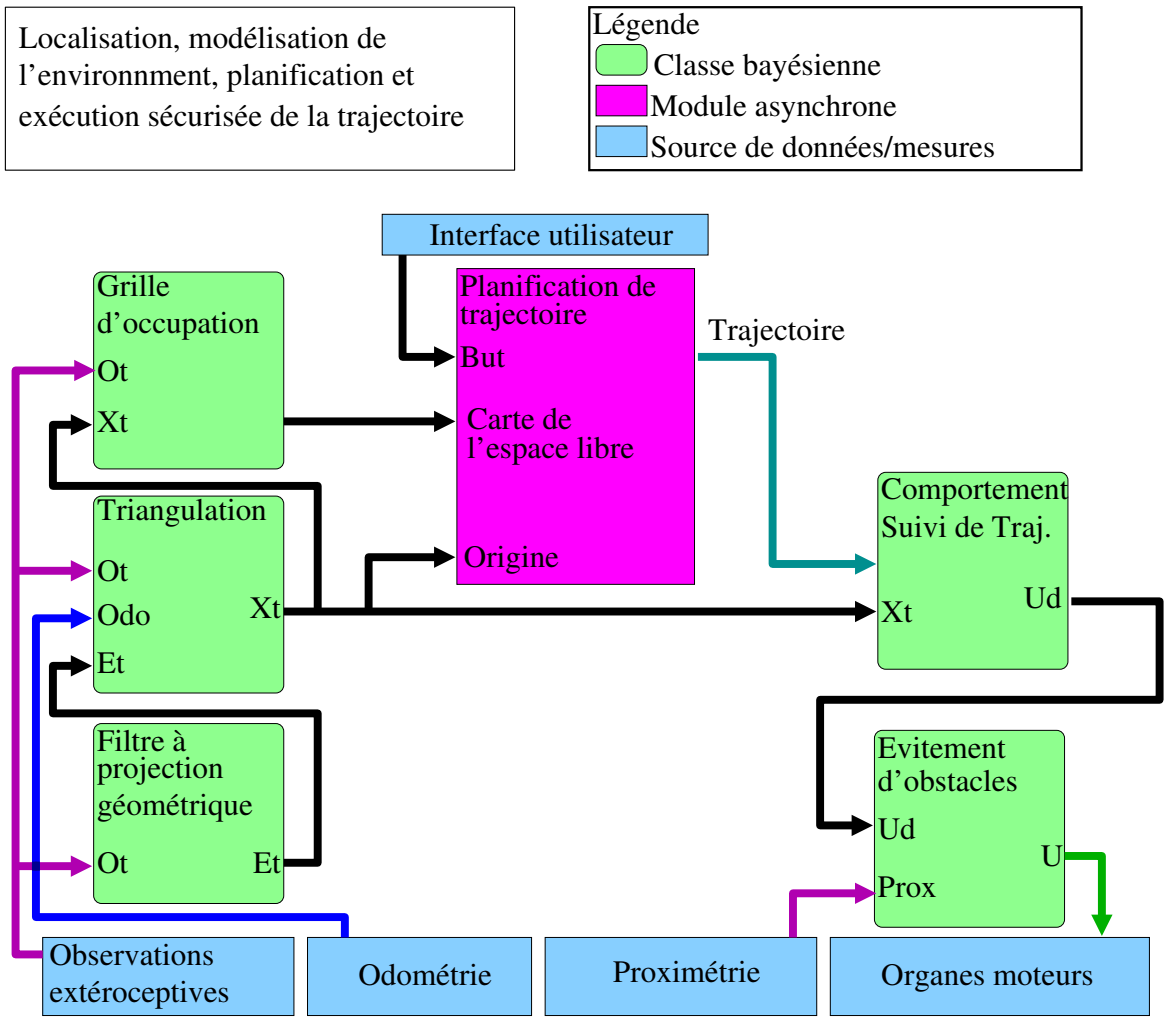


FIG. 7.11: Structure d'une application intégrée : localisation et modélisation de l'environnement, planification de trajectoire et réalisation de la trajectoire en présence d'obstacles.

FIG. 7.12: Images issues d'une expérimentation de navigation "sûre" sur une trajectoire planifiée.

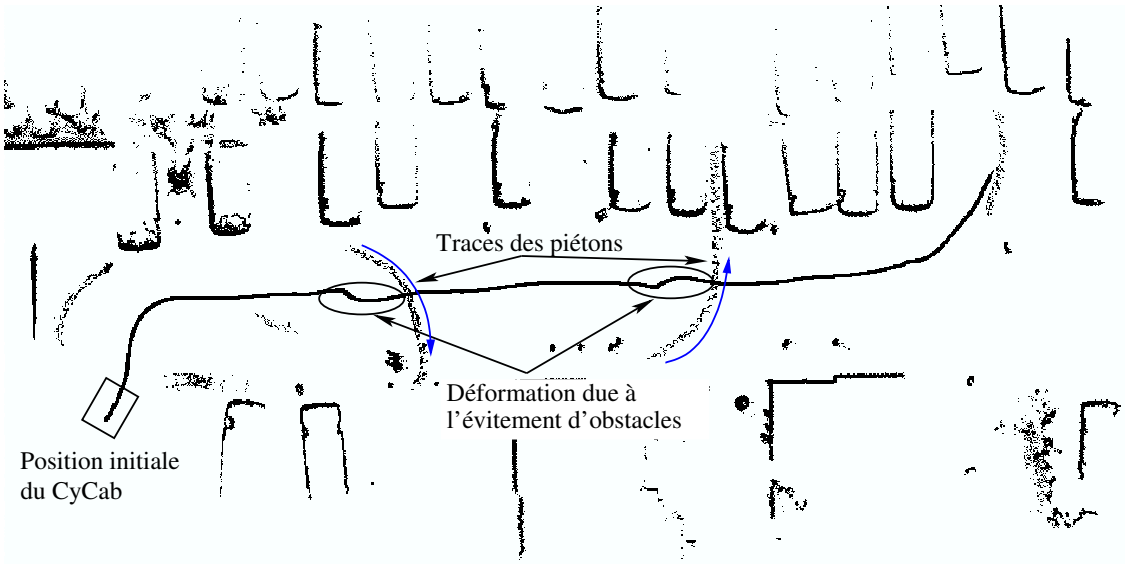


FIG. 7.13: Trajectoire réalisée au cours d'une navigation "sûre", en présence de piétons et d'autres véhicules inattendus, sur une trajectoire planifiée.

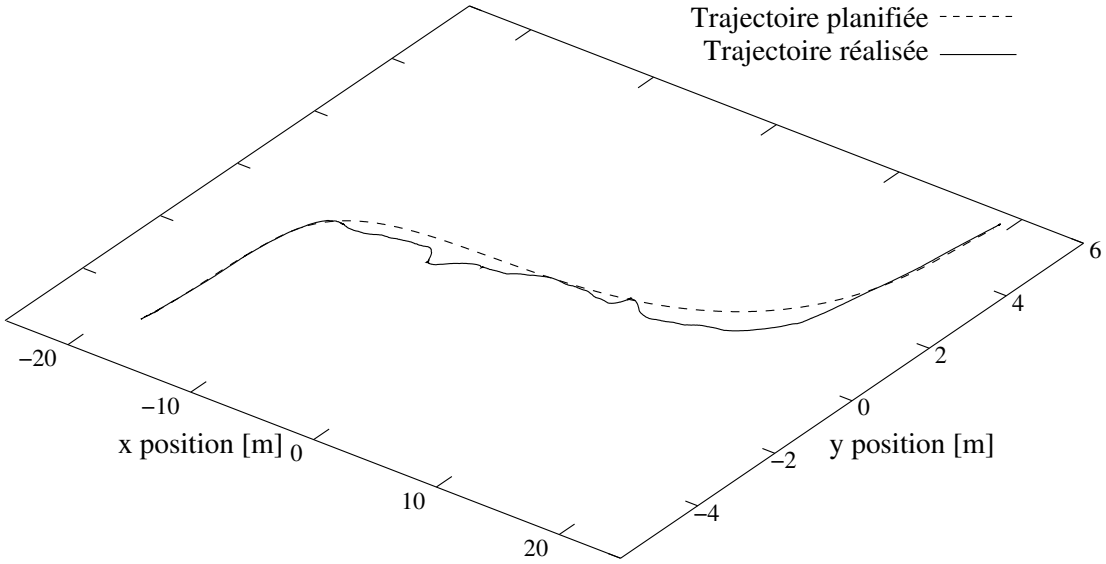


FIG. 7.14: Comparaison d'une trajectoire planifiée et de la trajectoire exécutée correspondante. La trajectoire est fortement déformée par la différence d'échelle entre les deux axes.

La figure 7.14 illustre les modifications que subit la trajectoire planifiée lorsqu'elle est exécutée en présence de piétons. Dans cet environnement, la trajectoire planifiée n'a pu être suivie avec précision qu'à ses extrémités.

Sur le reste de la trajectoire, la vitesse et l'angle de braquage sont ajustés par le comportement de suivi de trajectoire de façon à garantir la sécurité des piétons et du robot, tout en respectant la planification autant que possible.

7.5 Apprentissage et suivi sûr d'une trajectoire sensorimotrice

7.5.1 Objectifs

L'objectif du travail présenté dans cette section est de construire une application intégrée capable d'apprendre et de suivre un trajet sensorimoteur à partir d'une position quelconque dans le voisinage du trajet. Ce suivi doit de plus être réalisé de façon sûre, à la fois pour le robot et pour les autres entités présentes dans l'environnement. Cela signifie que l'on souhaite que le robot ne soit jamais responsable d'une collision avec un acteur de l'environnement et en particulier un piéton.

7.5.2 Implantation

On peut décomposer l'objectif ci-dessus en cinq tâches :

Apprentissage : Durant la phase d'apprentissage, l'application se limite à la structure décrite dans la figure 7.15 : le robot est piloté en conduite manuelle dans l'environnement pendant qu'il enregistre les perceptions extéroceptives et les commandes appliquées dans une base de données qui constituera la trajectoire sensorimotrice. Ainsi, à chaque instant, on enregistre la vitesse et l'angle de braquage appliqué par le robot et la perception extéroceptive courante : un ensemble de positions d'amers non identifiés dans le repère capteur.

Les quatre phases suivantes constituent l'application de suivi à proprement parler. Leurs interrelations sont illustrées par la figure 7.16.

Initialisation : Étant données une trajectoire sensorimotrice $\mathcal{T}_{sm}$ et une observation $O \in \mathbb{O}$, quelle est la position temporelle du robot $\tau(0)$? Cette tâche est accomplie par la classe bayésienne *Localisation temporelle initiale*.

Suivi : Étant données $\mathcal{T}_{sm}$, O et une estimation $\tau(t)$ de la position temporelle du robot à l'instant t, quelles sont les commandes qui permettent de suivre $\mathcal{T}_{sm}$? Ceci implique qu'il nous faut aussi conserver une estimation de la position temporelle $\tau(t)$ au cours du suivi. Les trois classes bayésiennes *Localisation temporelle*, *Localisation par mise en correspondance implicite* et *Comportement de suivi de trajectoire* travaillent conjointement pour réaliser cette tâche.

Diagnostic : Étant données $\mathcal{T}_{sm}$, O et $\tau(t)$, nous souhaitons que le système soit capable d'évaluer si les hypothèses (localisations, capteurs, environnement) sur lesquelles il fonde son comportement sont toujours valides. Pratiquement, cela signifie qu'il faudra maintenir une estimation de la confiance au cours du temps. C'est le rôle de la classe bayésienne d'*Estimation de confiance*.

Sécurité : En utilisant des capteurs de proximité, le robot doit être capable de garantir que les commandes qu'il applique sont sans danger de collision avec l'environnement, tant que les obstacles mobiles se déplacent à vitesse bornée. On utilise ici une classe bayésienne d'*Évitement d'obstacles*.

7.5.3 Liens avec les méthodes d'asservissement visuel

a) Principe

Cette application de suivi d'une trajectoire sensorimotrice nous paraît devoir être rapprochée des techniques d'asservissement visuel. Parmi les nombreux acteurs de ce domaine, on peut citer l'activité particulièrement marquée de Chaumette *[Chaumette 2004]* qui, lui aussi, a utilisé le CyCab comme plate-forme expérimentale.

Les hypothèses générales de l'asservissement perceptif sont les suivantes :

- On suppose que l'état du robot est décrit par une variable p et qu'il accepte des commandes grâce à une variable q.
- De plus, il dispose d'un système perceptif lui fournissant un vecteur d'observation O.
- L'élément clef de l'asservissement perceptif est de pouvoir exprimer la variation de l'observation comme une fonction, si possible linéaire, des commandes du système :

$$\dot{O} = f(q) \text{ ou } \dot{O} = F.q$$

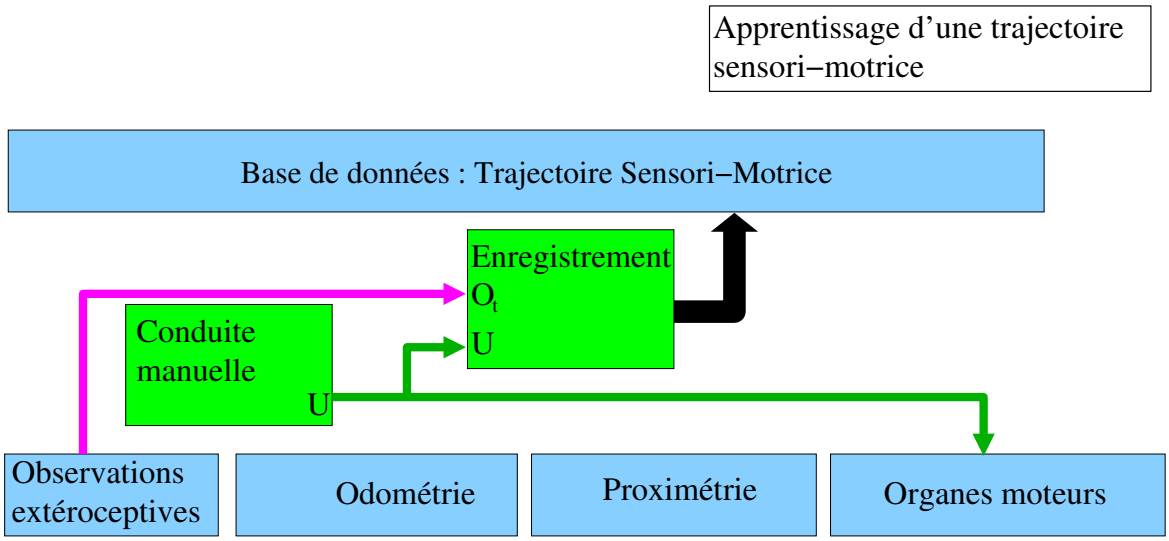


FIG. 7.15: Structure d'une application intégrée : apprentissage d'une trajectoire sensorimotrice.

Si f n'est pas linéaire, elle est généralement linéarisée autour du point de fonctionnement, et remplacée par son jacobien.

– On suppose par ailleurs que le système connaît une observation de référence O_r.

Le but est alors de trouver les commandes qui amènent le système à percevoir l'observation de référence, en la comparant avec l'observation courante O_c.

A partir de ces hypothèses, la solution de base des méthodes d'asservissement perceptif s'exprime de la façon suivante :

$$q = K.F^{-1}(O_r - O_c) \tag{7.1}$$

où K est un gain intervenant dans la dynamique de la réponse du système et dans ses propriétés de convergences.

b) Analyse

Dans notre contexte, puisqu'on suppose disposer d'une trajectoire sensorimotrice, on connaît une observation de référence $O_r(t)$ pour tout instant t. L'avantage qu'il y aurait à utiliser des techniques d'asservissement visuel serait principalement le gain en complexité obtenu en n'estimant pas l'erreur de suivi $\xi(t)$.

Toutefois, pour comparer directement deux observations il est nécessaire qu'elles aient une partie commune. Lorsque le capteur utilisé est doté d'un champ de vision relativement étroit (caméra par exemple), et lorsque l'évitement d'obstacles déforme le suivi de trajectoire, il est difficile de garantir que l'observation courante et l'observation de référence auront cette partie commune.

Au contraire, si on estime l'erreur de suivi, on peut s'éloigner de la trajectoire, et contrôler le robot même si l'observation de référence n'est pas observable dans l'observation courante. Dans ce cas, l'estimation de l'erreur de suivi ne se fera qu'avec l'odométrie jusqu'à ce que le véhicule soit suffisamment proche de la trajectoire de référence.

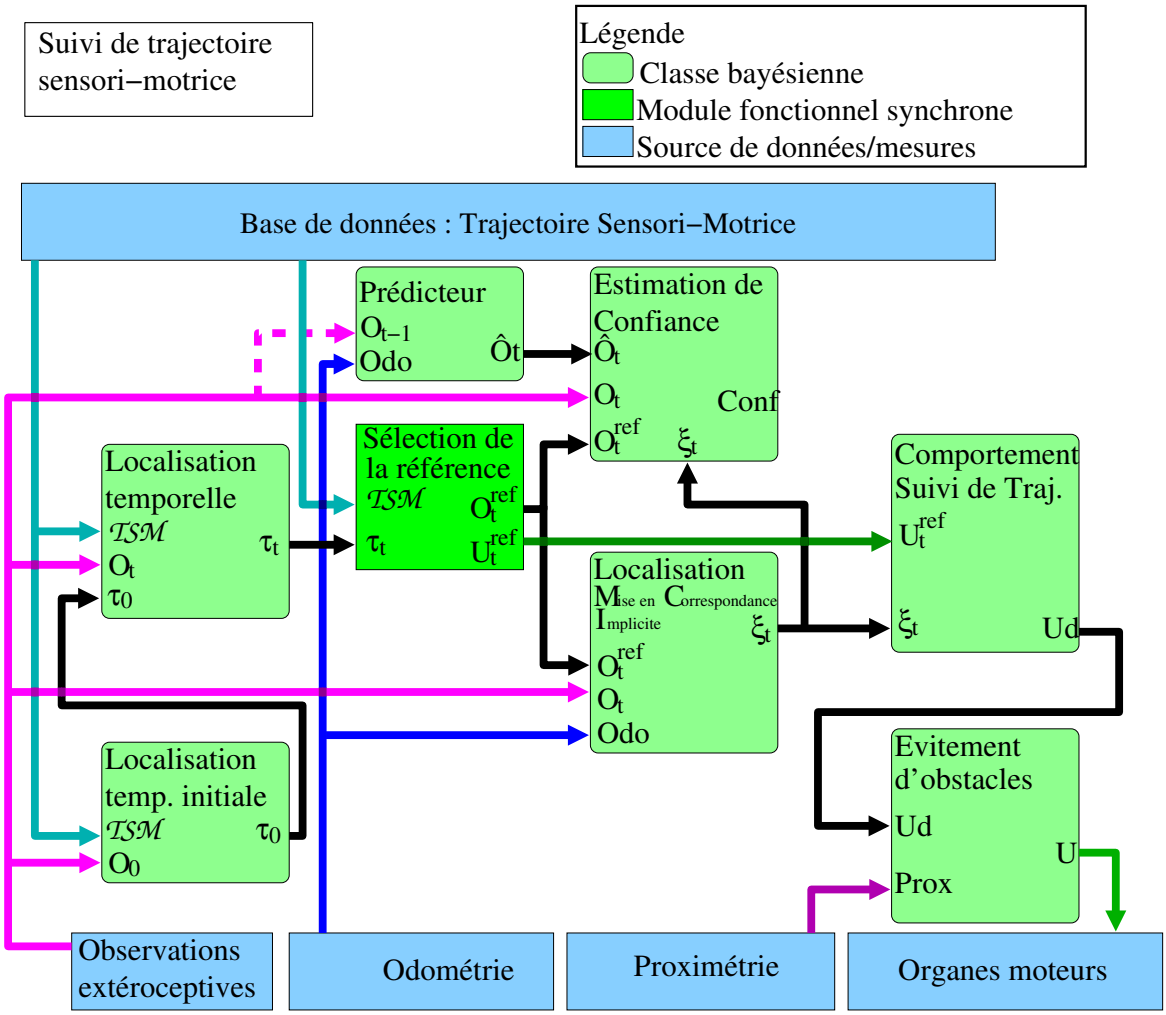


FIG. 7.16: Structure d'une application intégrée : suivi sûr d'une trajectoire sensorimotrice.

Par ailleurs, nous avons besoin de l'estimation de l'erreur de suivi dans notre algorithme d'estimation de la confiance.

7.5.4 Résultats simulés

Les figures 7.17 et 7.18 illustrent le fonctionnement de notre application intégrée dans deux situations simulées.

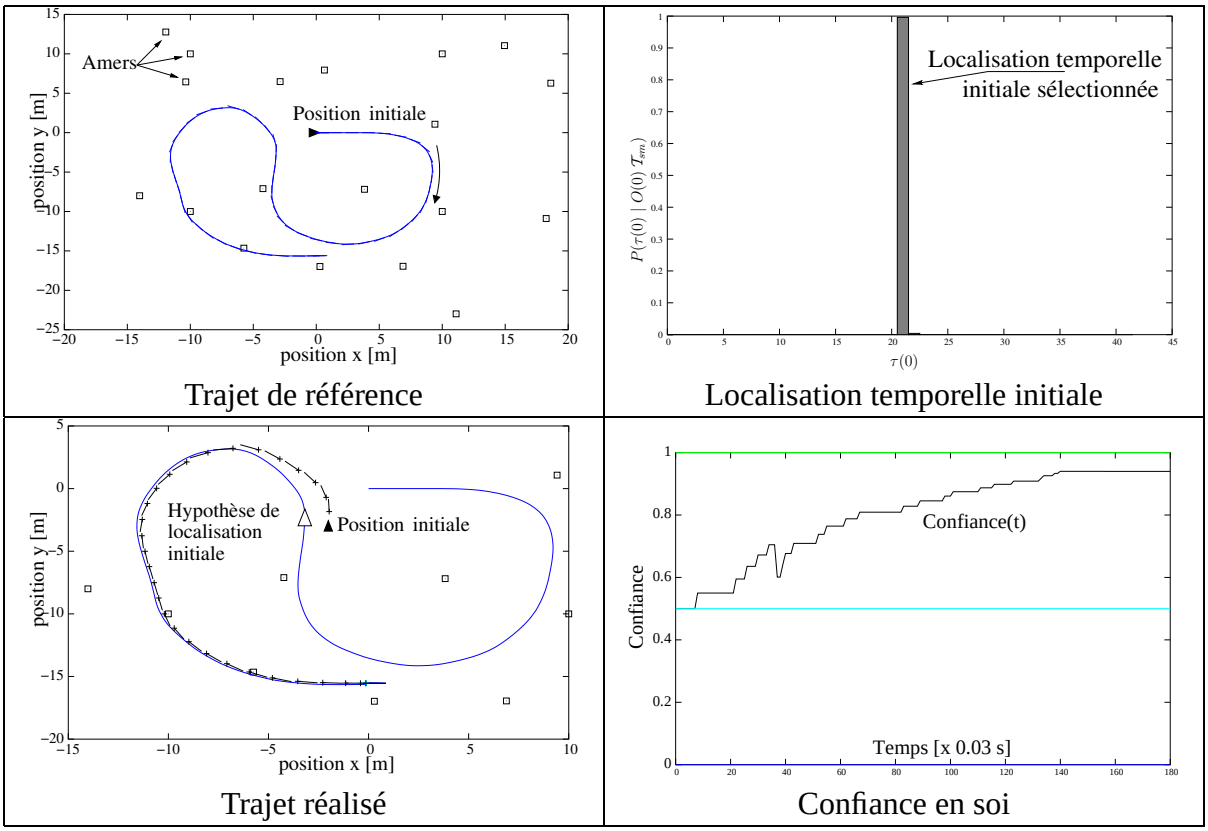


FIG. 7.17: Initialisation réussie et suivi du trajet sensorimoteur.

a) Protocole expérimental

On utilise la même trajectoire sensorimotrice de référence dans les deux applications. Celle-ci est enregistrée alors que le véhicule est conduit manuellement sur la trajectoire illustrée en haut à gauche des deux figures. Au cours de ce mouvement, on enregistre à la fois les vitesses et angles de braquage appliqués par le conducteur et les perceptions, sous la forme d'une séquence de positions d'amers non identifiés dans le repère capteur.

Avant de commencer le suivi de trajectoire, le véhicule est placé au milieu de la trajectoire, avec une erreur latérale de 1.5 mètre. Cette position initiale est visible en bas à gauche des figures 7.17 et 7.18. A partir de là, les deux situations diffèrent :

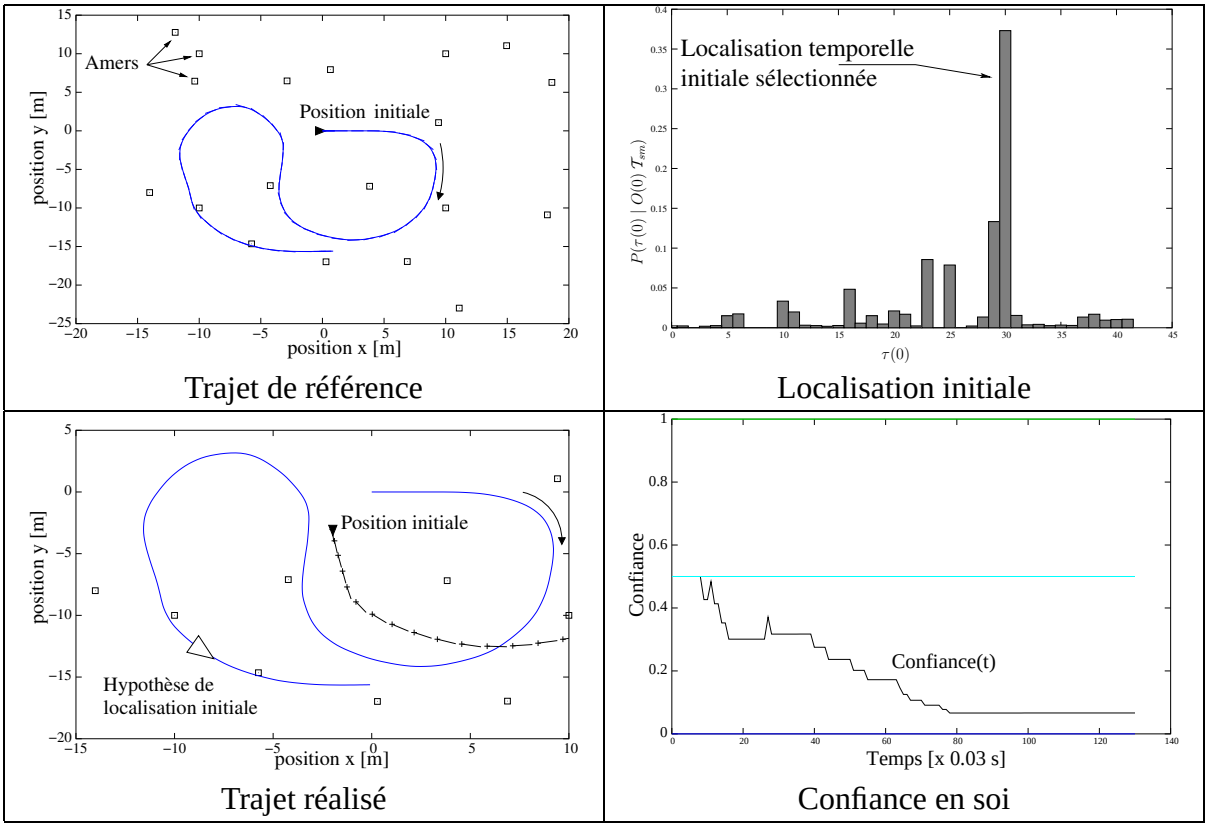


FIG. 7.18: Échec de l'initialisation et suivi du trajet sensorimoteur avec diagnostic d'erreur.

Situation 1 : Dans la figure 7.17, le véhicule est placé avec une orientation proche de celle de la trajectoire de référence dans cette région de l'espace.

Situation 2 : Au contraire, dans la figure 7.18, il a subi une rotation de 180 degrés.

b) Résultats

*1.***Dans la situation 1,** la localisation temporelle initiale (en haut à gauche de la figure) est très précise et très sure. Ceci est le résultat d'une très bonne correspondance entre l'observation initiale et la localisation temporelle sélectionnée. Cette localisation temporelle correspond au triangle blanc sur le trajet réalisé. A partir de cette localisation initiale, le comportement bayésien de suivi de trajectoire est appliqué de façon à rejoindre le trajet. A chaque instant, l'erreur de suivi n'est calculée qu'en comparant l'observation courante et les observations contenues dans la trajectoire sensorimotrice. On constate, en bas à gauche de la figure, que l'algorithme amène rapidement le véhicule sur sa trajectoire de référence.

Finalement, la confiance du système en lui-même est estimée au cours du mouvement (en bas à droite de la figure) en vérifiant la cohérence des observations réalisées avec celles résultant de la connaissance sur la localisation. On constate que la confiance monte rapidement au fur et à mesure que de nouvelles observations viennent confirmer les hypothèses de localisation.

*2.***Dans la situation 2,** la localisation temporelle initiale est beaucoup moins sûre que dans la situation précédente (en haut à gauche). Cependant une position temporelle se détache et est choisie comme position temporelle initiale. Elle est représentée par le triangle blanc sur le trajet réalisé. A partir de cette hypothèse, l'algorithme de suivi de trajectoire génère un mouvement. L'hypothèse de localisation initiale étant incohérente, le mouvement (en bas à droite) ne rejoint pas la trajectoire. Toutefois, on constate que cette situation est immédiatement détectée par l'estimateur de confiance (en bas à gauche).

Dans cette situation, l'estimation de confiance n'avait aucune influence sur le contrôle. Une façon basique d'en tenir compte serait de borner la vitesse par une vitesse maximale proportionnelle à l'espérance de la confiance. Ainsi le robot serait d'autant plus prudent qu'il a peu confiance en lui. Toutefois, la conception d'une intégration intelligente de la confiance dans le contrôle reste un sujet à explorer.

7.5.5 Intégration de l'évitement d'obstacles

Les résultats présentés dans les figure 7.17 et 7.18 correspondaient à des suivis de trajectoire exécutée dans un environnement sans obstacles. La figure 7.19 illustre le comportement de notre architecture de contrôle bayésienne dans un environnement plus complexe. Cette figure illustre en particulier la robustesse de notre approche par rapport aux modifications de l'environnement : des obstacles imprévus sont placés à proximité de la trajectoire pendant le suivi et quelques amers sont supprimés entre l'apprentissage et le suivi.

On constate que malgré ces modifications, le robot suit sa trajectoire sensorimotrice de façon assez précise. Toutefois, on constate aussi que les modifications d'orientation manquent parfois

de douceur. Cette amélioration du contrôle fait partie des optimisations pouvant être apportées à cette application.

7.5.6 Résultats sur la plate-forme réelle

La figure 7.20 présente le résultat d'un suivi de trajectoire sur notre robot réel. Les données extéroceptives sont issues de notre détecteur d'amers. En utilisant le module de localisation par mise en correspondance implicite, le comportement de suivi de trajet génère des commandes permettant de corriger l'erreur de suivi. Le module d'évitement d'obstacles vérifie alors que chaque commande proposée est sûre.

Sur la figure 7.20, on peut remarquer que la réalisation du trajet est exécutée assez précisément à $2ms^{-1}$, en passant à quelques dizaines de centimètres des voitures en stationnement. Le léger écart latéral constaté au milieu de la trajectoire est lié à la proximité d'un mur : l'angle de braquage permettant de corriger l'erreur ne pourrait être appliqué qu'en diminuant la vitesse, choix qui n'est pas privilégié avec les paramètres courants de l'application.

Nous sommes conscient que cette trajectoire expérimentale peut paraître un peu courte. En fait, il s'agit plus ici d'une limitation de l'environnement expérimental (manque d'amers artificiels et d'espace de manoeuvre) que d'une limitation de notre approche. Par ailleurs, la trajectoire que nous avons présenté est déjà trop longue pour être complètement visible dans le champ de notre caméra fixe, il aurait donc été difficile de visualiser une trajectoire plus longue.

7.6 Conclusions/Discussions

A travers de ces expériences, nous avons voulu démontrer plusieurs aspects des développement présentés dans les chapitres précédents :

- tout d'abord, la modularité de la programmation bayésienne. Modularité déjà envisagée dans les travaux de Lebeltel *[Lebeltel 1999]*, mais formalisées ici de façon plus efficace grâce d'une part à la PBOO et d'autre part à la notion d'abstraction ;
- ensuite, la complémentarité et la réutilisabilité de la plupart des techniques présentées dans ce document. A partir de cet ensemble d'algorithmes et de modèles, plusieurs applications peuvent être développées selon les classes bayésiennes utilisées ;
- finalement, l'efficacité et la pertinence de notre travail par rapport à la contrainte "R^3", leitmotiv du Robot System Lab de Canberra : un robot **R**éel travaillant dans un environnement **R**éel en temps **R**éel.

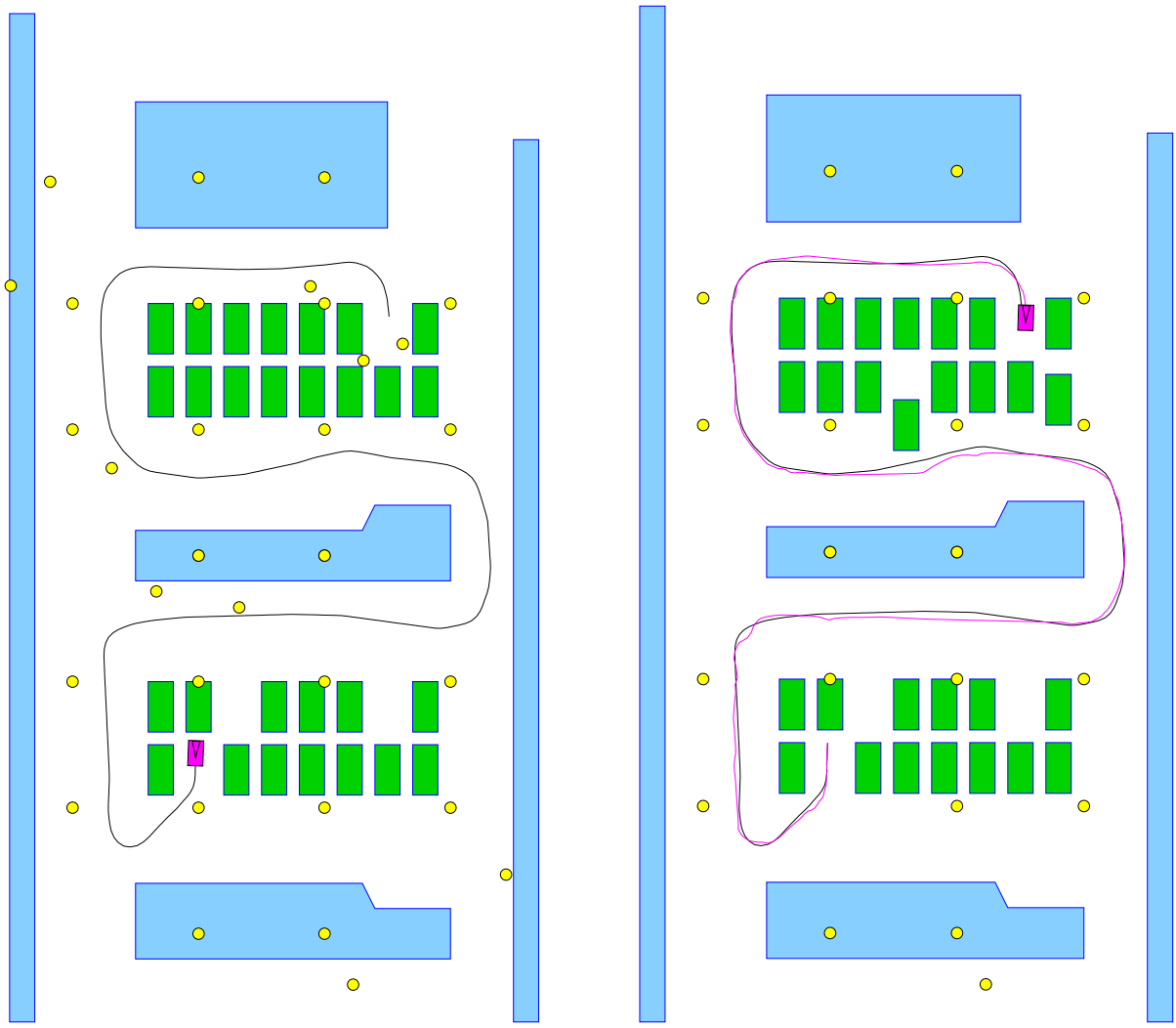


FIG. 7.19: Suivi d'une trajectoire sensorimotrice avec évitement d'obstacles. L'environnement est constitué d'obstacles polygonaux de façon à reproduire une géométrie de parking. Les cercles jaunes correspondent aux amers. A gauche, la trajectoire représentée est la trajectoire d'apprentissage. Le rectangle rose correspond à la position initiale du véhicule. A droite, la trajectoire en rose correspond à la trajectoire de suivi. Plusieurs amers ont été supprimés de l'environnement et quelques obstacles déplacés de façon à gêner le suivi de trajectoire.

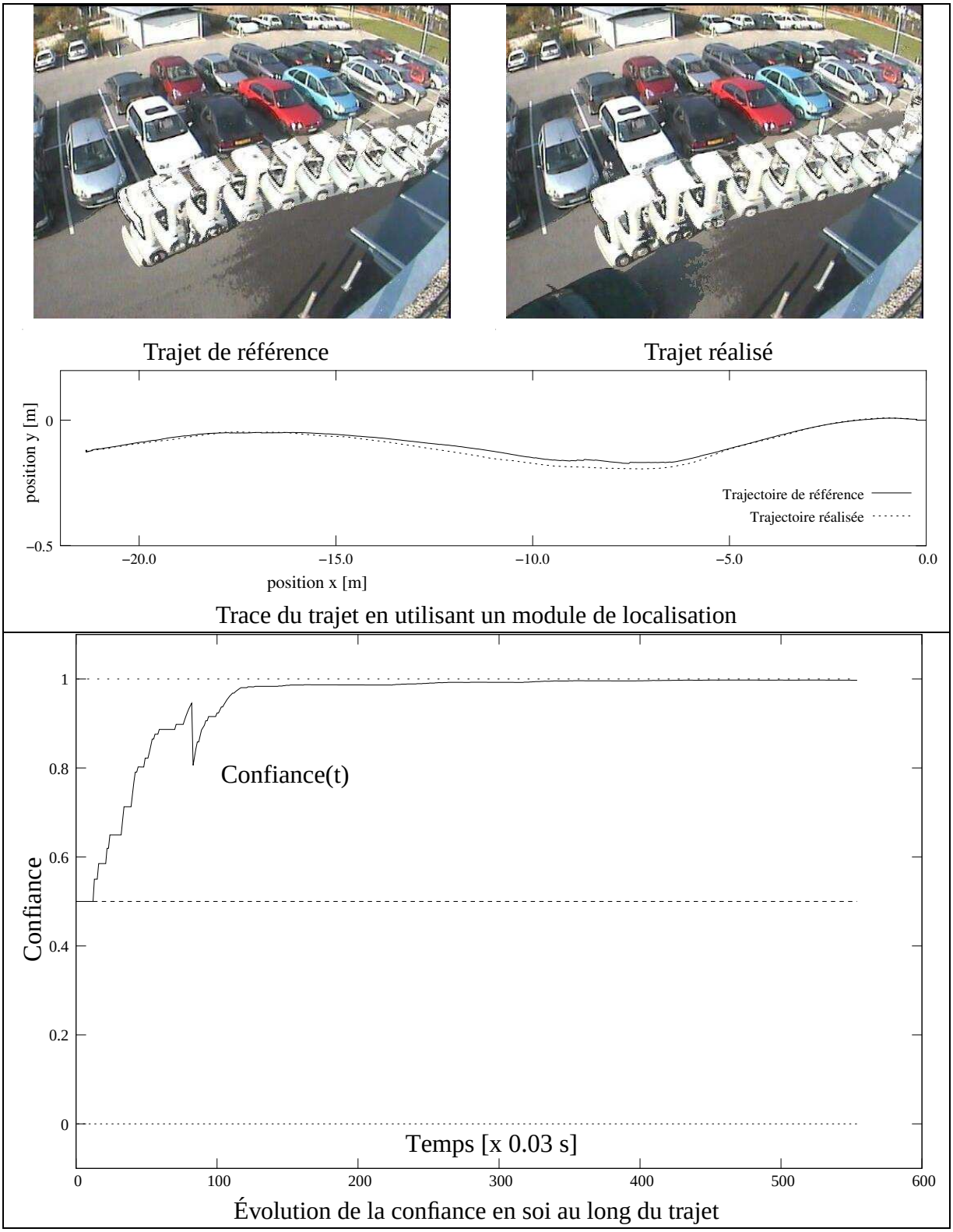


FIG. 7.20: Suivi d'un trajet sensorimoteur sur un véhicule réel.

Chapitre 8

Conclusions et perspectives

8.1 Conclusions

> Une tâche de navigation intentionnelle (TNI) est constituée d'un objectif et d'un moyen de rejoindre cet objectif, défini soit par un comportement, soit par un trajet.

A travers ce document, nous avons voulu étudier les mécanismes du mouvement et de la perception d'un système artificiel. Cette étude a été menée en cherchant à répondre à trois questions essentielles : comment définir une TNI ? Comment réaliser une TNI ? Comment implanter une TNI ?

8.1.1 Contributions

En traitant la question de la réalisation d'une TNI, nous avons été amenés à étudier trois compétences fondamentales pour l'autonomie d'un robot mobile : la localisation par rapport à un modèle de l'environnement, la modélisation autonome de l'environnement et la génération effective de commandes motrices. Cette étude nous a permis de développer un ensemble de contributions qui s'organisent comme suit :

a) Localisation

*1.***La localisation par rapport à une carte** a été notre premier centre d'intérêt :

- Nous avons d'abord étudié la localisation par triangulation. Pour cela, nous avons montré comment améliorer les performances d'un algorithme de mise en correspondance par graphes de correspondances en identifiant les triangles formés par des triplés d'amers.
- Ensuite, cette méthode n'étant pas capable de représenter de façon satisfaisante les incertitudes liées à la mise en correspondance, nous avons utilisé le formalisme bayésien pour définir une localisation avec mise en correspondance implicite.
- Finalement, nous avons montré comment améliorer encore la robustesse de notre localisation bayésienne en utilisant des modèles de diagnostic bayésien.

*2.***Localisation sans carte** En définissant un algorithme de localisation bayésienne par rapport à une trajectoire sensorimotrice, nous avons démontré l'utilisabilité de cette méthode alternative de représentation de l'environnement. Cette méthode est particulièrement intéressante car d'une part, elle ne demande pas de construire un modèle complet de l'environnement, et d'autre part, elle peut être vue comme une hypothèse sur la manière dont certains organismes modélisent leur environnement et s'y localisent.

*3.***Confiance** Quel que soit le type de localisation, avec ou sans carte, il nous a semblé indispensable que notre système soit capable de prendre du recul par rapport à l'exécution de sa tâche et évalue sa confiance en lui. Nous avons défini la confiance comme la capacité à utiliser son modèle de l'environnement et ses hypothèses de localisation pour prévoir les observations extéroceptives. Avec cette définition, nous avons montré comment estimer la confiance au cours du mouvement, toujours dans le formalisme bayésien.

b) Modélisation de l'environnement

Pour parvenir à construire, au cours du mouvement et de façon autonome, une représentation fiable et précise de l'environnement, nous avons réuni un algorithme efficace de construction de carte – le filtre à projection géométrique – et un algorithme robuste de mise en correspondance – les graphes de correspondances. Nous avons ainsi développé un algorithme unifié, le FPGMCI – filtre à projection géométrique avec mise en correspondance intégrée –, rassemblant modélisation de l'environnement, identification des observations par rapport à ce modèle et localisation, en mettant l'accent sur l'efficacité calculatoire, la robustesse et la précision des résultats.

c) Génération du mouvement

Afin de contrôler le mouvement effectif du robot, nous avons choisi d'utiliser le formalisme bayésien pour exprimer les comportements suivants :

- Un comportement de suivi de trajectoire exprimé sous la forme d'un problème d'inférence bayésienne.
- Un comportement d'évitement d'obstacles exprimé, lui aussi, sous la forme d'un problème d'inférence bayésienne, en mettant l'accent sur la sémantique de la spécification et en particulier sur la distinction entre spécification prescriptive et spécification proscriptive.

d) Conception d'une application

Dans le cadre de la conception d'applications complexes (chapitre 7), nous avons par ailleurs réfléchi sur les difficultés liées à l'intégration, sur un robot mobile, d'un ensemble de fonctionnalités indépendantes.

Pour traiter ces difficultés dans le cadre de la programmation bayésienne, nous avons défini les notions de classe bayésienne et de programmation bayésienne orientée objet (PBOO). Nous avons ensuite utilisé cette méthodologie pour spécifier et implanter nos applications.

8.1.2 Validation expérimentale

Ce travail a finalement été validé par de nombreuses expérimentations qui illustrent ce document et montrent notre attachement à traiter la réalité de la robotique : un robot réel, un environnement réel, des algorithmes en temps réel.

8.2 Perspectives

Parmi les perspectives envisageables pour prolonger notre travail, nous pouvons distinguer deux catégories : les extensions techniques à court terme et les problématiques à long terme.

Pour le court terme, nous envisageons les extensions suivantes :

Amers naturels : L'ensemble des résultats de localisation que nous avons présentés utilisent un détecteur d'amers artificiels. La nécessité d'ajouter ces amers est révélatrice d'un manque dans notre capacité à traiter les informations de nos capteurs.

Perception active : Il serait intéressant d'ajouter des compétences de perception active, c'est à dire la capacité à agir pour mieux percevoir ou percevoir avec moins d'ambiguïtés.

Construction de carte : Nous avons présenté un algorithme de construction de carte prévu pour modéliser un environnement statique. Cela signifie qu'il n'est pas capable de réagir correctement à la suppression ou au déplacement d'un amer. La construction d'une carte d'un environnement dynamique reste un problème ouvert actuellement. Cependant, nous travaillons en ce moment au développement d'un algorithme d'acquisition d'une carte d'un parking. Un tel algorithme devra pouvoir estimer l'état des places de parking et des véhicules en stationnement, afin de déterminer si une place est disponible, si elle vient d'être libérée ou occupée, et éventuellement estimer les paramètres du mouvement des véhicules non stationnés.

Utilisation de la confiance : Nous avons décrit un estimateur de confiance mais nous ne savons pas encore comment utiliser cette information autrement que comme une condition d'arrêt du véhicule. Il nous semblerait particulièrement pertinent de coupler ce mécanisme avec un mécanisme de perception active.

Suivi de trajectoire : Nous avons montré que la modélisation du problème de suivi de trajectoire comme un problème d'inférence bayésienne permettait d'obtenir un suivi précis et près à être mis en série avec un évitement d'obstacles. Toutefois, nous sommes bien conscients de la richesse des travaux réalisés dans ce domaine par les automaticiens. Il serait donc très intéressant de voir comment concilier des lois de commande mathématiquement solides et rigoureuses avec la souplesse d'une architecture de contrôle bayésienne.

Évitement d'obstacles : Notre évitement d'obstacles, comme toute méthode réactive, peut mener le robot à une impasse. Il est nécessaire de détecter ces situations et d'agir en conséquence, en replanifiant une trajectoire par exemple. Par ailleurs, les algorithmes d'évitement d'obstacles que nous avons présentés dans ce document ne sont adaptés qu'à des environnements faiblement dynamiques. Pour traiter des situations plus dynamiques, il sera sans doute nécessaire de faire appel à des outils plus spécialisés tels que les *Velocity*

Obstacles[Large 2003] afin d'être capable de prévoir les mouvements des obstacles et de se projeter dans le futur.

Passage à l'échelle : Pour passer à l'échelle, il serait sans doute pertinent de travailler dans des environnements de plus grande taille, avec plus de capteurs, et en particulier des capteurs débarqués avec lesquels le véhicule pourrait communiquer. Ce point est actuellement étudié dans le cadre du projet ParkNav[1].

En considérant maintenant des objectifs à plus long terme, l'élément bloquant pour l'évolution de la robotique mobile nous paraît être la pauvreté des capacités de perceptions artificielles. Aucun dispositif artificiel à notre connaissance n'est capable d'apporter une interprétation sémantique des informations perceptives dans un temps compatible avec une application robotique. La sémantique est indispensable au comportement intelligent d'un robot : dans le cadre d'une application d'évitement d'obstacles par exemple, c'est la sémantique qui permet à un conducteur humain de passer à quelques centimètres d'un mur mais de doubler un vélo en laissant un mètre d'écart. Pour citer de nouveau A. Berthoz, "le cerveau humain est une machine à prédire", mais cette machine ne peut prédire le comportement d'un objet qu'en lui associant une sémantique. Cette capacité à interpréter les informations perceptives est selon nous la clef qui permettra de réaliser des applications robotiques capables de gérer des applications vraiment complexes : la conduite autonome d'un véhicule automobile en ville par exemple. Les automaticiens ont proposé de nombreuses techniques de contrôle du mouvement ; pour aller plus loin, il faut maintenant que les systèmes autonomes artificiels soient capables non seulement de percevoir leur environnement, mais aussi de le comprendre et même de l'apprendre.

[1] Projet ROBEA impliquant le laboratoire GRAVIR à Grenoble et le LAAS à Toulouse.

Bibliographie

R.C. Arkin. Reactive robotic systems, 1991.

G. Artus, P. Morin, et C. Samson. Tracking of an omnidirectional target with a nonholonomic mobile robot. In *Proc. of the IEEE Int. Conf. on Advanced Robotics*, 2003.

S. Arulampalam, S. Maskell, N. Gordon, et T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions of Signal Processing*, 2002. URL citeseer.nj.nec.com/maskell01tutorial.html

J.-C. Baccon, L. Hafemeister, et P. Gaussier. A context and task dep endent visual attention system to control a mobile robot. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, 2002.

J.A. Bagnell et J. Schneider. Autonomous helicopter control using reinforcement learning policy search methods. In *IEEE Int. Conf. Robotics Automation*, 2001. URL citeseer.ist.psu.edu/bagnell01autonomous.html

T. Bailey, E.M Nebot, J.K. Rosenblatt, et H.F. Durrant-Whyte. Data association for mobile robot navigation : a graph theoretic approach. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 2000.

P. Bessière et BIBA-INRIA Research Group. Survey : Probabilistic methodology and techniques for artefact conception and development. Technical Report RR-4730, INRIA, Grenoble, France, February 2003. http ://www.inria.fr/rrrt/rr-4730.html.

P. Bessière, E. Dedieu, O. Lebeltel, E. Mazer, et K. Mekhnacha. Interprétation vs. description I : Proposition pour une théorie probabiliste des systèmes cognitifs sensori-moteurs. *Intellectica*, 1998a.

P. Bessière, E. Dedieu, O. Lebeltel, E. Mazer, et K. Mekhnacha. Interprétation vs. description II : Fondements mathématiques. *Intellectica*, 1998b.

M. Betke et K. Gurvits. Mobile robot localization using landmarks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1994. URL citeseer.ist.psu.edu/betke95mobile.html

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.

J. Borenstein et Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 1989. URL citeseer.nj.nec.com/borenstein89realtime.html

J. Borenstein et Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 1991. URL citeseer.nj.nec.com/borenstein91vector.html

F. Bourgault, T. Furukawa, et H. Durrant-Whyte. Process model, constraints, and the coordinated search strategy. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2004.

A.J. Briggs, D. Scharstein, D. Braziunas, C. Dima, et P. Wall. Mobile robot navigation using self-similar landmarks. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2000.

O. Brock et 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.

J. Buhmann, W. Burgard, A. B. Cremers, D. Fox, T. Hofmann, F. Schneider, J. Strikos, et S. Thrun. The mobile robot Rhino. *Artificial Intelligence Magazine*, 1995.

R. Chatila et J.P. Laumond. Position referencing and consistent world modeling for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1985.

F. Chaumette. Image moments : a general and useful set of features for visual servoing. *IEEE Trans. Robotics and Automation*, 2004.

H.I. Christensen, editor. *Summerschool on "Simultaneous Localisation and Mapping"*, KTH, Stockholm, 2002. URL http://www.cas.kth.se/SLAM/toc.html

G. Cooper. The computational complexity of probabilistic inference using bayesian belief network. *Artificial Intelligence*, 1990.

C. Coué. *Modèle bayésien pour l'analyse multimodale d'environnements dynamiques et encombrés : application à l'assistance à la consuite automobile en milieu urbain.* Thèse de doctorat, Inst. Nat. Polytechnique de Grenoble, December 2003.

C. Coué et P. Bessière. Chasing an elusive target with a mobile robot. In *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems*, Hawai (HI), 2001.

C. Coué, Th. Fraichard, P. Bessière, et 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.

C. Coué, C. Pradalier, et C. Laugier. Bayesian programming for multi-target tracking : an automotive application. In *Proc. of the Int. Conference on Field and Service Robotics*, Lake Yamanaka (JP), July 2003.

R.T. Cox. Probability, frequency and reasonable expectation. *American Journal of Physics*, 1946.

R.T. Cox. *The Algebra of Probable Inference*. The John Hopkins Press, Baltimore, 1961.

J. Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Scottsdale (Az), mai 1989.

M Csorba et H Durant-Whyte. A new approach to map building using relative position estimates. *SPIE*, 1997a.

M Csorba et H Durant-Whyte. *A new approach to map building using relative position estimates*. PhD thesis, Australian Center for Field Robotics,Sidney, 1997b.

M. Deans et M. Hebert. Invariant filtering for simultaneous localization and mapping. In *ICRA*, 2000.

J. Diard. *La carte bayésienne : un modèle probabiliste hiérarchique pour la navigation en robotique mobile*. Thèse de doctorat, Inst. Nat. Polytechnique de Grenoble, January 2003a.

Julien Diard. *La carte bayséienne – Un modèle probabiliste hiérarchique pour la navigation en robotique mobile*. Thèse de doctorat, Institut National Polytechnique de Grenoble, Grenoble, France, Janvier 2003b.

G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, et M. Csorba. A solution to the simultaneous localisation and map building (slam) problem. *IEEE Trans. Robotics and Automation*, 2001.

A. Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer, Special Issue on Autonomous Intelligent Machines*, 1989.

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

R. García-Ramírez. *Programmation Bayesienne des Bras Manipulateurs*. Thèse de doctorat, Institut National Polytechnique de Grenoble (INPG), Grenoble (FR), Mai 2003.

Russell Greiner et Ramana Isukapalli. Learning to select useful landmarks. In *National Conference on Artificial Intelligence*, 1994. URL citeseer.ist.psu.edu/article/greiner94learning.html

J. Guivant, F. Masson, , et E. Nebot. Simultaneous localization and map building using natural features and absolute information. *Robotics and Autonomous Systems*, 2002.

J. Guivant et E. Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Trans. Robotics and Automation*, 2001.

J. Guivant et E. Nebot. Improving computational and memory requirements of simultaneous localization and map building algorithms. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2002.

J.S. Gutmann et K. Konolige. Incremental mapping of large cyclic environments. In *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 2000.

D. Hall et J. Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 1997.

J. Hermosillo. *Planification et exécution de mouvements pour un robot bi-guidable : une approche basée sur la platitude différentielle*. Thèse de doctorat, Inst. Nat. Polytechnique de Grenoble, Grenoble (FR), June 2003.

J. Hermosillo, C. Pradalier, et S. Sekhavat. Modelling odometry and uncertainty propagation for a bi-steerable car. In *Proc. of the IEEE Intelligent Vehicle Symp.*, Versailles (FR), June 2002. Poster session.

J. Hermosillo, C. Pradalier, S. Sekhavat, et C. Laugier. Experimental issues from map building to trajectory execution for a bi-steerable car. In *Proc. of the IEEE Int. Conf. on Advanced Robotics*, Coimbra (PT), July 2003a.

J. Hermosillo, C. Pradalier, S. Sekhavat, et C. Laugier. Autonomous navigation of a bi-steerable car : Experimental issues. *Machine Intelligence and Robotic Control Journal*, 2004. To appear.

J. Hermosillo, C. Pradalier, S. Sekhavat, Ch. Laugier, et G. Baille. Towards motion autonomy of a bi-steerable car : Experimental issues from map-building to trajectory execution. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Taipei (TW), May 2003b.

J. Hertz, A. Krogh, et R.G. Palmer. *Introduction to the theory of neural computation*. Addison-Wesley, 1991.

J. Hérault et Ch. Jutten. *Réseaux neuronaux et traitement du signal*. Hermès, Paris, 1994.

E. T. Jaynes. *Probability Theory : the Logic of Science*. Cambridge University Press, 2003.

R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 1960.

Y. Kanayama, Y. Kimura, F. Miyazaki, et T. Noguchi. A stable tracking control method for an autonomous mobile robot. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1990.

O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1985.

B. Kluge. Recursive probabilistic velocity obstacles for reflective navigation, 2003. URL citeseer.ist.psu.edu/kluge03recursive.html.

C. Koike, C. Pradalier, P. Bessière, et E. Mazer. Obstacle avoidance and proscriptive bayesian programming. In *Proc. of the Workshop on Reasoning with Uncertainty in Robotics*, Acapulco (MX), July 2003a.

C. Koike, C. Pradalier, P. Bessière, et E. Mazer. Proscriptive bayesian programming application for collision avoidance. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Las vegas, NV (US), October 2003b.

F. Lamiraux, D. Bonnafous, et C. Van Geem. Path optimization for nonholonomic systems : application to reactive obstacle avoidance and path planning. In A. Bicchi, H.I. Christensen, et D. Prattichizzo, editors, *2nd International Workshop on Control Problems in Robotics and Automation*. Springer tracts in advanced robotics, 2002.

F. Lamiraux, J.P. Laumond, et C. Van Geem. Nonholonomic path deformation : application to nonholonomic system trajectory optimization. In *3rd IARP International Workshop on Service, Assistive and Personal Robots : Technical Challenges and Real World Application Perspectives*, Madrid (Espagne), 2004.

Pierre Lamon, Illah Nourbakhsh, Björn Jensen, et Roland Siegwart. Deriving and matching image fingerprint sequences for mobile robot localization, 2000. URL citeseer.ist.psu.edu/445679.html.

F. Large. *Navigation Autonome d'un Robot Mobile en Environnement Dynamique et Incertain*. Thèse de doctorat, Université de Savoie, Chambéry (FR), November 2003.

F. Large, S. Sekhavat, Z. Shiller, et 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.

J. Latombe. *Robot Motion Planning*. Kluwer Academic Publications, Boston, USA, 1995.

O. Lebeltel. *Programmation Bayésienne des Robots*. Thèse de doctorat, Institut National Polytechnique de Grenoble, Grenoble, France, Septembre 1999.

O. Lebeltel, P. Bessière, J. Diard, et E. Mazer. Bayesian robots progamming. *Autonomous Robots*, 2003. In Press.

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

O. Lebeltel, P. Bessière, J. Diard, et E. Mazer. Bayesian robot programming. *Autonomous Robots*, 2004.

O. Lefebvre, F. Lamiraux, C. Pradalier, et Th. Fraichard. Obstacles avoidance for car-like robots. integration and experimentation on two robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, New Orleans, LA (US), April 2004.

J.J. Leonard et H.F. Durrant-Whyte. Simultaneous map building and localisation for an autonomous mobile robot. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, 1991.

Uri Lerner, Ronald Parr, Daphne Koller, et Gautam Biswas. Bayesian fault detection and diagnosis in dynamic systems. In *AAAI/IAAI*, 2000. URL citeseer.nj.nec.com/lerner00bayesian.html

F.L. Lewis. *Optimal Estimation*. John Wiley & Sons, Inc., 1986.

R. Madhavan, H. Durrant-Whyte, et G. Dissanayake. Natural landmark-based autonomous navigation using curvature scale space. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2000.

K. Mekhnacha. *Méthodes probabilistes baysiennes pour la prise en compte des incertitudes géométriques : application à la CAO-robotique*. Thèse de doctorat, Institut National Polytechnique de Grenoble (INPG), Grenoble (FR), juillet 1999a.

K. Mekhnacha. *Méthodes probabilistes baysiennes pour la prise en compte des incertitudes géométriques : application à la CAO-robotique*. Thèse de doctorat, Inst. Nat. Polytechnique de Grenoble, Grenoble (FR), July 1999b.

K. Mekhnacha, E. Mazer, et P. Bessière. The design and implementation of a bayesian CAD modeler for robotic applications. *Advanced Robotics*, 2001.

N. Mitsunaga et M. Asada. Visual attention control for a legged mobile rob ot based on information criterion. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, 2002.

M. Montemerlo, S. Thrun, D. Koller, et B. Wegbreit. Fastslam : A factored solution to the simultaneous localization and mapping problem. *Proc. of the Nat. Conf. on Artificial Intelligence*, 2002. URL citeseer.ist.psu.edu/article/montemerlo02fastslam.html

Kevin P. Murphy. Switching kalman filters, 1998. URL citeseer.nj.nec.com/murphy98switching.html

J. Neira et J.D. Tardos. Data association in stochastic mapping using the joint compatibility test. *IEEE Trans. Robotics and Automation*, 2001.

P. Newman. *On the structures and solution of simultaneous localization and mapping problem*. PhD thesis, Australian Center for Field Robotics,Sidney, 1999.

G. Oriolo, G. Ulivi, et M. Vendittelli. *Fuzzy maps : Managing uncertainty in sensor-based motion planning*. Prentice-Hall, 1997.

G. Oriolo, G. Ulivi, et M. Vendittelli. Real-time map building and navigation for autonomous robots in unknown environments. *IEEE Trans. On Systems, Man and Cybernetics - Part B : Cybernetics*, 28, 1998.

Bessière P., Ahuactzin J.M., Talbi E.G., et Mazer E. The ariadne's clew algorithm : global planning with local methods. *The Algorithmic Foundations of Robotics*, 1995.

C. Pradalier, P. Bessière, et C. Laugier. Driving on a known sensori-motor trajectory with a car-like robot. In *Proc. of the Int. Symp. on Experimental Robotics*, Singapore (SG), June 2004a.

C. Pradalier et P. Bessière. Perceptual navigation around a sensori-motor trajectory. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, New Orleans, LA (US), April 2004.

C. Pradalier, F. Colas, et P. Bessière. Expressing bayesian fusion as a product of distributions : Applications in robotics. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Las vegas, NV (US), October 2003a.

C. Pradalier, F. Colas, et P. Bessière. Expressing bayesian fusion as a product of distributions : Application to randomized hough transform. In *Proc. of the Conf. on Bayesian Methods and Maximum Entropy in Science and Engineering*, Jackson Hole, WY (US), August 2003b.

C. Pradalier, J. Hermosillo, C. Koike, C. Braillon, P. P.Bessière, et C. Laugier. Safe and autonomous navigation for a car-like robot among pedestrian. In *IARP Int. Workshop on Service, Assistive and Personal Robots*, Madrid (ES), October 2003c.

C. Pradalier, J. Hermosillo, K. Koike, C. Braillon, P. Bessière, et C. Laugier. An autonomous car-like robot navigating safely among pedestrians. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2004b.

C. Pradalier et S. Sekhavat. Concurrent matching, localization and map building using invariant features. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Lausanne (CH), September-October 2002a.

C. Pradalier et S. Sekhavat. «localization space» : a framework for localization and planning, for systems using a sensor/landmarks module. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Washington, DC (US), May 2002b.

C. Pradalier et S. Sekhavat. Simultaneous localization and mapping using the geometric projection filter and correspondence graph matching. *Advanced Robotics*, 2004. To appear.

R. Garcia Ramirez. *Programmation bayésienne des bras manipulateurs*. Thèse de doctorat, Inst. Nat. Polytechnique de Grenoble, Grenoble (FR), May 2003.

C. Samson et K. Ait-Abderrahim. Feedback stabilization of a nonholonomic wheeled mobile robot. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, 1991.

A. Scheuer et Th. Fraichard. Collision-free and continuous-curvature path planning for car-like robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, volume 1, pages 867–873, Albuquerque, NM (US), April 1997.

C. Schmidt. *Appariement d'images par invariants locaux de niveaux de gris*. Thèse de doctorat, Institut National Polytechnique de Grenoble (INPG), Grenoble (FR), 1996.

S. Sekhavat et J.-P. Laumond. Topological property for collision-free nonholonomic motion planning : the case of sinusoidal inputs for chained form systems. *IEEE Trans. Robotics and Automation*, 14(5) :671–680, October 1998.

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.

C. Soyer, I. Bozma, et Y. Istefanopulos. Apes : Actively perceiving robot. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, 2002.

R.S. Sutton et A.G. Barto. *Reinforcement Learning*. MIT Press, 1998.

S. Thrun. Finding landmarks for mobile robot navigation. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1998a. URL citeseer.ist.psu.edu/thrun98finding.html

S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 1998b.

S. Thrun. Robotic mapping : A survey. In G. Lakemeyer et B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, et D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot Minerva. *International Journal of Robotic Research*, 2000.

N. Tomatis, I. Nourbakhsh, et R. Siegwart. Hybrid simultaneous localization and map building : a natural integration of topological and metric. *Robotics and Autonomous Systems*, 2003.

I. Ulrich et J. Borenstein. VFH+ : Reliable obstacle avoidance for fast mobile robots. In *IEEE Int. Conf. Robotics Automation*, Leuven, Belgium, May 1998. URL citeseer.nj.nec.com/ulrich98vfh.html

I. Ulrich et J. Borenstein. VFH* : Local obstacle avoidance with look-ahead verification. In *IEEE Int. Conf. Robotics Automation*, 2000. URL citeseer.nj.nec.com/ulrich00vfh.html

I. Ulrich et I.R. Nourbakhsh. Appearance-based place recognition for topological localization. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2000. URL citeseer.ist.psu.edu/article/ulrich00appearancebased.html.

G. Welch et G. Bishop. An introduction to the Kalman filter, 97. URL http://www.cs.unc.edu/~welch/kalman/index.html.

S.B. Williams, H.F. Durrant-Whyte, et G. Dissanayake. Constrained initialization of the simultaneous localization and mapping algorithm. *Int. Journal of Robotics Research*, 2003.

B. Yamauchi et P. Langley. Place recognition in dynamic environments. *Journal of Robotic System*, 1997.

Annexe A

Expressions de fusion probabiliste

Le texte présenté dans cette annexe est en grande partie la traduction synthétique de deux articles publiés aux conférences IROS'03[Pradalier et al. 2003a] *et MaxEnt'03*[Pradalier et al. 2003b].

La fusion de données est un problème récurrent dans des domaines tels que la robotique mobile, du diagnostic médical assisté par ordinateur ou du contrôle comportemental de personnages simulés. De façon générale, la fusion de données peut se comprendre comme l'*estimation* d'une variable d'état par rapport à des mesures sensorielles, la *fusion* de diagnostics venant de différents experts, ou *la sélection d'actions* parmi celles proposées par différents modules.

De par son principe, la fusion de données issues de plusieurs modèles possède des avantages significatifs par rapport à l'utilisation d'une source donnée unique. Il y a, bien sûr, l'avantage statistique obtenu par la combinaison de plusieurs données de même origine (obtenant ainsi une estimation améliorée d'un phénomène physique grâce à l'accumulation d'observations redondantes). A cet avantage s'ajoute le gain de précision obtenue en observant un même phénomène de différentes manières, en utilisant donc des modèles différents. Les applications de fusion de données multi-modèles, et tout particulièrement, multi-capteurs sont nombreuses, aussi bien dans le domaine civil que militaire. Le livre *[Hall & Llinas 1997]* fournit un tour d'horizon des technologies de fusion de données multi-capteurs et de ses applications.

Par ailleurs, il existe des situations où, de même que des données sensorielles, les opinions de divers experts ou même les commandes à appliquer à un moteur ne peuvent être exprimées avec une précision arbitraire. Dans ces cas, la logique seule peut difficilement exprimer le processus de fusion de manière satisfaisante. Ce processus peut toutefois être formalisé efficacement dans le contexte de la programmation bayésienne(chapitre 2), de façon à confronter différentes sources de connaissance dans un environnement incertain. Cela a été illustré, par exemple dans les travaux de Lebeltel *[Lebeltel et al. 2003]* et Coué*[Coué et al. 2002]*.

Dans la suite de cette annexe, nous nous intéressons à l'évaluation d'une variable V, à partir de la connaissance d'un ensemble d'autres variables $V_1 \dots V_n$: $P(V \mid V_1 \dots V_n)$. Dans le cas de la fusion multi-capteurs, V peut représenter la position du robot et $V_1 \dots V_n$, les valeurs lues sur ses capteurs. Dans ce cas, le programmeur peu spécifier chaque modèle de capteur par la distribution conditionnelle $P(V_k \mid V)$. C'est ce que nous appelons un modèle *direct*, particulièrement intéressant car dans ce cas, on peut appliquer directement la règle de Bayes pour obtenir le résul-

tat de la fusion de données. De plus, nous montrerons dans la section A.1 que $P(V \mid V_1 \dots V_n)$ est alors proportionnelle au produit des opinions de chaque modèle : $P(V \mid V_k)$. C'est une propriété intéressante car elle peut mener à des calculs plus efficaces en terme de temps de calcul ou de mémoire nécessaire.

Considérons maintenant le cas de la fusion de commande. Dans ce cas, V représentera la commande à appliquer. D'autre part pour le programmeur, il est alors naturel d'exprimer l'influence de chaque mesure capteur sur la commande à appliquer : $P(V \mid V_k)$. Nous appellerons ce type de spécification un modèle inverse, car, pour calculer la fusion des commandes, il est nécessaire d'utiliser une première fois la règle de Bayes pour inverser chaque sous-modèle. Ce type de fusion et ses conséquences seront traitées dans la section A.2. On montrera en particulier que la distribution de fusion n'est plus le produit des distributions venant des sous-modèles.

Finalement, la section A.3 présentera une manière alternative de spécifier un modèle de fusion de données. Ce modèle utilisera une variable de diagnostic qui permettra d'écrire la fusion comme un produit de distributions. L'application de ce schéma de fusion ne sera pas présenté dans cette annexe car plusieurs exemples détaillés peuvent être trouvé dans ce document, en particulier aux chapitres 4 et 6.

Notations : Pour terminer cette introduction, voici les conventions qui seront utilisées dans le reste de cette annexe :

- V : une variable,
- $\vec{V}$: un ensemble de variable $V_1 \dots V_n$,
- v : une valeur particulière de la variable V.

Nous utiliserons de plus un ensemble de variables dont la sémantique est donnée ci-dessous :

- A : *avis ou opinion* d'un expert, ou fusion de différents avis venant de différents sous-modèles (la position d'un robot ou une commande moteur par exemple) ;
- D ou D_k : une donnée mesurée ;
- π_f ou π_k : connaissances *a priori*.

A.1 Fusion bayésienne à partir de "modèles directs"

Afin d'être dans de bonnes conditions pour la suite de cette annexe, il est nécessaire de comprendre le mécanisme de la fusion bayésienne classique, tel que présentée dans *[Lebeltel et al. 2000]*.

Tout d'abord, nous supposons que nous savons exprimer $P(D_k \mid A\,\pi_k)$, π_k étant l'ensemble des connaissances *a priori* utilisées par le programmeur pour décrire le modèle k liant D_k à A. On s'intéresse alors à $P(A \mid D_1 \dots D_n\,\pi_f)$. Dans le contexte de la robotique mobile, $P(D_k \mid A\,\pi_k)$ pourrait être un modèle de capteur, capable de prévoir la mesure D_k connaissant la position du robot A.

En utilisant une approche modulaire de la programmation, on commence par définir des sous-modèles qui expriment les connaissances *a priori* π_k. En pratique, pour tout k, on utilise la règle

de Bayes pour construire la distribution conjointe suivante :

$$P(A\, D_k \mid \pi_k) \;=\; P(A \mid \pi_k) P(D_k \mid A\, \pi_k) \tag{A.1}$$

Ensuite, nous supposons n'avoir aucune connaissance *a priori* sur A, ce qui nous amène à définir $P(A \mid \pi_k)$ par une loi uniforme. Dans ce cas, par application directe de la règle de Bayes, on obtient :

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

Puisque $P(A \mid \pi_k)$ est uniforme et $P([D_k = d_k] \mid \pi_k)$ ne dépend pas de a, on obtient la propriété suivante :

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

Pour simplifier les notations, on notera l'équation précédente comme suit : $P(A \mid D_k\, \pi_k) \propto P(D_k \mid A\, \pi_k)$.

En utilisant la règle de Bayes et en supposant que les données mesurées sont indépendantes, on peut maintenant exprimer la distribution conjointe complète pour le système :

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

Pour rester cohérent avec les sous modèles, on choisit de définir $P(A \mid \pi_f)$ par une loi uniforme et on pose $P(D_k \mid A\, \pi_f) \;=\; P(D_k \mid A\, \pi_k)$.

La distribution à laquelle nous nous intéressons est alors la suivante :

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

Comme $P([\vec{D} = \vec{d}] \mid \pi_f)$ ne dépend pas de a, la proportionnalité qui était vrai pour les sous-modèles reste vrai pour le modèle complet :

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

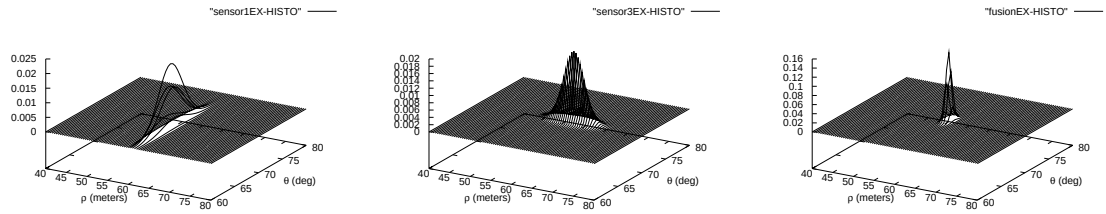
finalement, en insérant l'équation A.4, on obtient :

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

Ainsi, la distribution de probabilité sur A, résultat de l'observation de n données d_k, est proportionnelle au produit des distributions de probabilité provenant de l'observation individuelle de chaque donnée.

Ce résultat est intuitivement satisfaisant pour au moins deux raisons :

– En premier lieu, si on n'utilise qu'un seul sous-modèle, le résultat de la fusion de son unique opinion sera bien son opinion non modifiée. Ainsi, le processus de fusion n'introduit pas de connaissance supplémentaire.
– En second lieu, si la dimension de A est supérieure à 1, et si chaque sous-modèle fournit une information sur une dimension de A, la projection de la distribution de fusion sur une dimension particulière correspondra à l'opinion non modifiée exprimée par le sous-modèle correspondant. Cette propriété est bien illustrée dans *[Coué et al. 2002]* (cf. figure A.1).



$P(A \mid D_1)$ $P(A \mid D_2)$ $P(A \mid D_1\, D_2)$

FIG. A.1: Fusion de données sur une variable bidimensionnelle. Figure tirée de *[Coué et al. 2002]*.

A.2 Fusion bayésienne à partir de modèle à inverser

Dans un contexte de localisation, il est souvent possible de prédire les données capteur à partir de la position du robot. La distribution conjointe peut alors être écrite en utilisant la règle de Bayes :

$$P(\text{Pose Sensor1 Sensor2}) = P(\text{Pose})P(\text{Sensor1} \mid \text{Pose})P(\text{Sensor2} \mid \text{Pose}) \tag{A.8}$$

C'est ce que nous appelons un modèle directe car il peut être construit directement à partir de ce que nous savons exprimer.

Au contraire, dans un contexte de fusion de commande, on sait en général exprimer la distribution des commandes connaissant une mesure capteur $P(\text{Command} \mid \text{Sensor1})$, mais l'inverse est souvent plus difficile. Nous nous intéressons alors à la distribution de probabilité sur les commandes : $P(\text{Command} \mid \text{Sensor1 Sensor2})$. Malheureusement, il n'est pas possible de construire une distribution conjointe commandes : $P(\text{Command Sensor1 Sensor2})$ en utilisant une seule fois la règle de Bayes. C'est pourquoi, il va nous falloir passer par plusieurs sous-modèles (autant que de capteurs) et les inverser. D'où le nom de "modèle à inverser".

Formellement, supposons que nous savons exprimer $P(A \mid D_k \pi_k)$ au lieu de $P(D_k \mid A \pi_k)$, et que nous nous intéressons toujours à l'évaluation de $P(A \mid \vec{D} \pi_f)$.

Comme dans la section précédente, nous allons utiliser une approche de programmation modulaire et commencer par définir des sous-modèles exprimant les connaissances π_k :

$$P(A\, D_k \mid \pi_k) \;=\; P(D_k \mid \pi_k) P(A \mid D_k\, \pi_k) \tag{A.9}$$

où $P(D_k \mid \pi_k)$ suit une loi uniforme. A partir de ces sous-modèles, en utilisant la règle de Bayes, on peut exprimer $P(D_k \mid A\, \pi_k)$. On peut alors utiliser cette expression dans un modèle global :

$$P(A\, \vec{D} \mid \pi_f) \;=\; P(A \mid \pi_f) \prod_k P(D_k \mid A\, \pi_f) \tag{A.10}$$

en posant $P(D_k \mid A\, \pi_f) \;=\; P(D_k \mid A\, \pi_k)$.

Dans ce cas, quelle que soit la forme de $P(A \mid \pi_f)$, on obtient

$$\begin{aligned} & P(A \mid \vec{D}\, \pi_f) \\ & \propto \;\; P(A \mid \pi_f) \prod_k P(D_k \mid A\, \pi_k) \\ & \propto \;\; P(A \mid \pi_f) \prod_k \frac{P(D_k \mid \pi_k) P(A \mid D_k\, \pi_k)}{P(A \mid \pi_k)} \end{aligned} \tag{A.11}$$

Dans le cas général ($P(A \mid \pi_f)$ non spécifiée, uniforme...), ceci amène à

$$P(A \mid \vec{D}\, \pi_f) \not\propto \prod_k P(A \mid D_k\, \pi_k) \tag{A.12}$$

Or ce résultat va à l'encontre de la compréhension intuitive du processus de fusion bayésienne que nous avions obtenue dans la section A.1.

Il faut noter toutefois qu'il est possible d'obtenir cette proportionnalité : il suffit de définir $P(A \mid \pi_f)$ telle que

$$\frac{P(A \mid \pi_f)}{\prod_k P(A \mid \pi_k)} \;=\; \text{constante} \tag{A.13}$$

Ce qui correspond, en pratique à

$$\begin{aligned} P(A \mid \pi_f) \;\; & \propto \;\; \prod P(A \mid \pi_k) \\ & \propto \;\; \prod_k \sum_{D_k} P(A \mid D_k\, \pi_k) P(D_k \mid \pi_k) \end{aligned} \tag{A.14}$$

En utilisant cette distribution de probabilité, nous obtenons effectivement un processus de fusion intuitif, mais la compréhension de la "signification physique" de $P(A \mid \pi_f)$ devient plutôt problématique.

A.3 Fusion bayésienne par diagnostic

A.3.1 Définitions

Dans cette section, nous introduisons une nouvelle variable :
– $\mathbb{I}$ ou $\mathbb{I}_k$: variable booléenne qui **I**ndique si l'opinion A est *cohérente* avec la mesure D_k.
On exprime ensuite les sous-modèles suivants :

$$P(A\, D_k\, \mathbb{I}_k \mid \pi_k) \;=\; P(A \mid \pi_k) P(D_k \mid \pi_k) P(\mathbb{I}_k \mid A\, D_k\, \pi_k) \tag{A.15}$$

Où A et D_k sont indépendants et suivent une loi uniforme. La distribution conditionnelle sur $\mathbb{I}_k$ doit être spécifié par le programmeur. Il pourrait choisir par exemple :

$$P([\mathbb{I}_k \;=\; 1] \mid A\, D_k\, \pi_k) \;=\; \exp\left(-\frac{1}{2}\left[\frac{A - D_k}{\sigma}\right]^2\right) \tag{A.16}$$

L'indépendance de A et D_k peut paraître choquante de prime abord. Il faut toutefois se rappeler que $\mathbb{I}_k$ est définit de façon à exprimer la cohérence entre ces variables, et donc leur dépendance.

L'avantage principal d'un tel modèle provient de sa capacité à exprimer

$$P(A \mid \vec{D}\vec{\mathbb{I}}\pi_f) \propto \prod_k P(A \mid D_k\, \mathbb{I}_k\, \pi_k) \tag{A.17}$$

Cet avantage est illustré par la figure A.2. Celle-ci permet de comparer les résultat d'une fusion par inversion et d'une fusion par diagnostic. Tout en étant parfaitement correct d'un point de vue mathématique, la fusion par inversion amène parfois à des résultats contre-intuitifs, alors qu'une fusion par diagnostic produit bien le résultat attendu.

A.3.2 Preuve de l'équation A.17

Tout d'abord, en utilisant la règle de Bayes, on peut écrire

$$\begin{aligned} &P([A \;=\; a] \mid [\vec{D} \;=\; \vec{d}][\vec{\mathbb{I}} \;=\; \vec{i}]\, \pi_f) \\ &= \frac{P([A \;=\; a]\, [\vec{D} \;=\; \vec{d}] \mid \pi_f)}{P([\vec{D} \;=\; \vec{d}]\, [\vec{\mathbb{I}} \;=\; \vec{i}] \mid \pi_f)} \times P([\vec{\mathbb{I}} \;=\; \vec{i}] \mid [A \;=\; a]\, [\vec{D} \;=\; \vec{d}]\, \pi_f) \end{aligned} \tag{A.18}$$

Or, grâce aux hypothèses contenues dans les π_k, A et D_k sont indépendants et suivent une loi uniforme. On en déduit que $P([A \;=\; a]\, [\vec{D} \;=\; \vec{d}] \mid \pi_f)$ ne dépend pas de a. On peut alors écrire :

$$P([A \;=\; a] \mid [\vec{D} \;=\; \vec{d}][\vec{\mathbb{I}} \;=\; \vec{i}]\, \pi_f) \propto P([\vec{\mathbb{I}} \;=\; \vec{i}] \mid [A \;=\; a]\, [\vec{D} \;=\; \vec{d}]\, \pi_f) \tag{A.19}$$

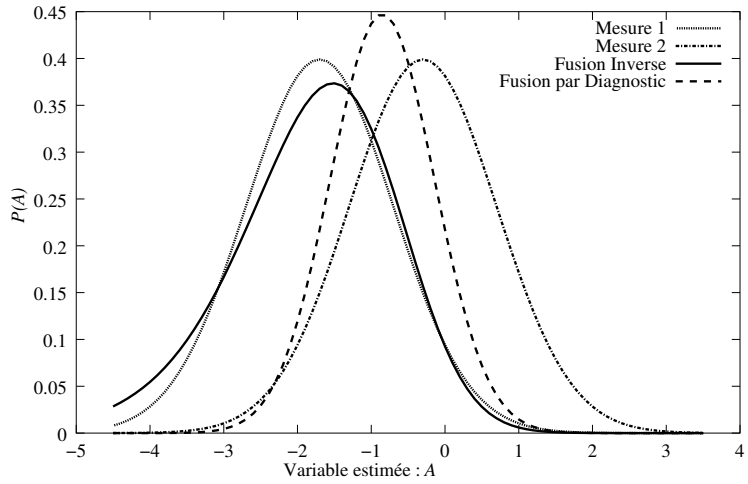


FIG. A.2: Comparaison des processus de fusion.

En supposant tous les modèles capteurs indépendants, on obtient :

$$\begin{aligned} & P([A = a] \mid [\vec{D} = \vec{d}]\,[\vec{\mathit{I\!I}} = \vec{\imath}]\,\pi_f) \\ & \quad \propto \; P([\vec{\mathit{I\!I}} = \vec{\imath}] \mid [A = a]\,[\vec{D} = \vec{d}]\,\pi_f) \\ & \quad \propto \; \prod_k P([\mathit{I\!I}_k = i_k] \mid [A = a]\,[D_k = d_k]\,\pi_k) \end{aligned} \tag{A.20}$$

Une autre application de la règle de Bayes donne :

$$\begin{aligned} & P([A = a] \mid [\vec{D} = \vec{d}]\,[\vec{\mathit{I\!I}} = \vec{\imath}]\,\pi_f) \\ & \quad \propto \; \prod_k \left[\frac{P([D_k = d_k]\,[\mathit{I\!I}_k = i_k] \mid \pi_k)}{P([A = a]\,[D_k = d_k] \mid \pi_k)} \times P([A = a] \mid [D_k = d_k]\,[\mathit{I\!I}_k = i_k]\,\pi_k) \right] \end{aligned} \tag{A.21}$$

En utilisant de nouveau les hypothèses contenus dans les π_k, A et D_k sont indépendants et suivent une loi uniforme. On en déduit que $P([A = a]\,[D_k = d_k] \mid \pi_k)$ ne dépend pas de a. Ce qui prouve l'équation A.17 :

$$P([A = a] \mid [\vec{D} = \vec{d}][\vec{\mathit{I\!I}} = \vec{\imath}]\,\pi_f) \propto \prod_k P([A = a] \mid [D_k = d_k]\,[\mathit{I\!I}_k = i_k]\,\pi_k) \tag{A.22}$$

A.3.3 Expression de $P(\mathit{I\!I}_k \mid A\, D_k\, \pi_k)$

Il est parfois plus facile d'exprimer quelque chose qui ressemble à $P(A \mid D_k\, \pi_k)$ ou $P(D_k \mid A\, \pi_k)$ plutôt que d'exprimer directement $P(\mathit{I\!I}_k \mid A\, D_k\, \pi_k)$. Comment construire cette dernière distribution à partir de ce que nous savons exprimer ?

En fait, la variable $\mathit{I\!I}_k$ nous apporte plus que la simple connaissance de $P(A \mid D_k\, \pi_k)$ (ou $P(D_k\, \pi_k \mid A)$). Elle nous permet aussi de décrire les données d'un capteur qui ne fonctionne pas de la manière attendue, ou un comportement incohérent au vue de certaines mesures. Pour

pouvoir calculer $P(\mathbb{I}_k \mid A\ D_k\ \pi_k)$, nous devons donc savoir exprimer non seulement $P(A \mid D_k\ [\mathbb{I}_k = 1]\ \pi_k)$ (habituellement noté $P(A \mid D)$), mais aussi $P(A \mid D_k\ [\mathbb{I}_k = 0]\ \pi_k)$.

Dans ce cas, grâce à l'équation A.21, il existe c_0 et c_1 tels que

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

D'où

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

En définissant

$$\begin{aligned}
P_0 &= P([A = a] \mid [\mathbb{I}_k = 0]\ D_k\ \pi_k) \\
P_1 &= P([A = a] \mid [\mathbb{I}_k = 1]\ D_k\ \pi_k) \\
U_A &= P([A = a] \mid \pi_k) \leftarrow \text{Valeur de l'uniforme sur } A,
\end{aligned}$$

on obtient un système linéaire en c_0 et c_1, à partir du quel on calcule :

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

On peut faire deux remarques au sujet de ce résultat :

- Tout d'abord, ce résultat est indéterminé lorsque $P_0(a) = P_1(a)$. Un moyen de lever cette indétermination est de poser $P_0(a) = P_1(a) = U_A$ en ces points. Ceci permet bien d'obtenir une valeur pour $P(\mathbb{I} \mid [A = a]\ D_k)$, mais cette dernière ne dépend plus des valeurs relatives de $P_0(a)$ et $P_1(a)$, mais de leurs dérivées. La sémantique de ces valeurs devient alors assez floue.
- Connaissant P_1, fonction continue de a, comment définir P_0 pour que $P(\mathbb{I}_k \mid A\ D_k)$ soit une fonction continue de a ? Une condition suffisante (sans doute non nécessaire) est de choisir $C > \max(P_1)$, puis de poser $P_0 \propto C - P_1$. Si C est inférieur à $\max(P_1)$, $C - P_1$ est parfois négative, ce qui n'est pas compatible avec une distribution de probabilité. Avec un tel C, $P(\mathbb{I}_k \mid A\ D_k)$ est prolongeable par continuité aux points où $P_0(a) = P_1(a)$.

On peut donc exprimer *facilement* $P(\mathbb{I}_k \mid A\ D_k\ \pi_k)$ à partir de la connaissance de $P(A \mid [\mathbb{I}_k = 1]\ D_k\ \pi_k)$. Puisque ces expressions sont symétriques en A et D_k, la connaissance de $P(D_k \mid A\ [\mathbb{I}_k = 1]\ \pi_k)$ serait elle aussi suffisante.

A.3.4 Propriétés

En général, nous nous intéressons aux données qui rentrent dans le cadre de nos modèles, ce qui se traduit par $\vec{\mathbb{I}} = \vec{1}$. Ainsi, lorsqu'on calcule $P(A \mid \vec{D}) \propto \prod_k P(A \mid D_k)$, on peut considérer qu'on se place implicitement dans le contexte de l'équation A.17, avec $\vec{\mathbb{I}} = \vec{1}$.

Un autre aspect intéressant de cette forme de fusion apparaît lorsqu'on utilise qu'un seul expert dans un schéma général destiné à recueillir l'avis de n experts. Le résultat de la fusion de cet avis unique est l'avis initial. Ainsi, comme dans une fusion de modèle directs, la fusion par diagnostic n'ajoute pas de connaissance. Dans le cas général, nous avons vu que cela ne peut être garanti par la fusion par inversion de modèle.

A.3.5 Conclusion

Cette annexe a permis de présenter nos réflexions sur la fusion de données bayésienne. Nous nous sommes principalement intéressé à l'expression symbolique du résultat d'une fusion de données probabiliste, et en particulier à son expression sous forme de produit. Il existe des situations, telles que la fusion de commandes, ou le résultat de la fusion doit, sémantiquement, être un produit. Dans ces cas, à défaut de pouvoir utiliser une fusion par modèle direct, il est nécessaire d'utiliser un modèle de diagnostic, caractérisé par la présence de variables booléennes définissant la cohérence entre données mesurées et variables estimées.

Les avantages de la fusion par diagnostic sont multiples. Tout d'abord, comme nous nous plaçons dans le contexte de la programmation bayésienne, nous en conservons les avantages :

- Un formalisme mathématique clair et bien défini,
- Une formulation générique et uniforme des problèmes.

La fusion par diagnostic ajoute ensuite les avantages suivant :

- Une spécification du modèle qui est symétrique en termes de variables d'entrée et de sortie,
- Une schéma de fusion qui peut toujours s'exprimer sous la forme d'un produit de distributions de probabilité.
- En bonus : une validité mathématique pour des expériences qui marchaient bien sans que l'on puisse leur donner une formulation mathématique correcte.