# Carte bayésienne et apprentissage

Eva Simonin




**École Doctorale**
**"Mathématique, sciences et technologie de l'information, informatique"**

**Master 2ème année de Mathématiques, Informatique**
Spécialité *Intelligence, Interaction, et Information*

Titre

# Carte bayésienne et apprentissage

**Éva SIMONIN**

le 9 Juin 2004

Laboratoire GRAVIR, équipe-projet e-Motion
Encadrants : Pierre Bessière et Julien Diard

**Jury**

P. Bessière
J. Caelen
Y. Demazeau
J. Euzenat
J.-L. Schwartz

# Table des matières

# Chapitre 1

# Introduction

Ce projet de Master 2ème année d'Informatique s'inscrit dans le domaine de la navigation en robotique mobile. Plus précisément, nous nous intéressons au problème de la représentation de l'espace dans lequel un robot navigue. Cette représentation, qui prend souvent la forme d'une carte métrique, doit lui permettre de résoudre des tâches de navigation.

Dans la littérature, cette carte est soit donnée au robot, soit apprise expérimentalement. Des méthodes de planification lui permettent ensuite de traduire un but dans l'environnement en une séquence d'actions devant lui faire atteindre ce but. Pour appliquer avec succès cette séquence d'actions dans l'environnement et effectivement atteindre le but, le robot doit enfin résoudre des problèmes de localisation, afin de connaître sa position dans la carte.

Ces deux processus, de planification et de localisation, sont cependant supposés indépendants par l'essentiel des travaux en robotique mobile. Par exemple, la construction de la carte est supposée indépendante de son utilisation ultérieure.

Nous nous intéressons au cas où cette indépendance supposée est levée. À notre connaissance, il n'existe pas de travaux traitant du problème de l'apprentissage d'une carte *en vue de la génération de plusieurs comportements*. Plus généralement, il n'existe pas de théorie communément admise qui décrive la relation entre des représentations de l'espace de type carte et des comportements. Nous étudions cette relation dans un scénario d'apprentissage de carte.

Nous décrivons dans ce travail une expérience robotique illustrant ce type d'apprentissage. L'objectif de cette expérience est de montrer que l'on peut apprendre une représentation de l'espace dans laquelle sont corrélées les connaissances sur les perceptions et sur les actions du robot. Nous montrerons aussi que l'apprentissage de cette représentation n'est pas plus difficile à mettre en œuvre qu'un apprentissage permettant l'identification d'un comportement unique. De cette représentation, nous pourrons tirer directement des comportements permettant de résoudre des tâches de navigation variées. En effet, elle contient les connaissances nécessaires pour déduire les conséquences probables d'une action du ro-

bot, au vu de ses données sensorielles. Étant capable d'estimer ce type de conséquence, le robot peut évaluer quelle action le rapprochera du but correspondant à la tâche de navigation à résoudre.

De plus, cette expérience illustre un autre aspect des relations entre carte et comportement. En effet, la carte est apprise grâce à l'observation des effets de l'application d'un comportement dans l'environnement. Ainsi les connaissances sur la corrélation entre les actions et les données sensorielles du robot sont acquises à partir d'un simple comportement initial. Pourtant ces connaissances permettent de construire une carte d'où seront tirés divers comportements, inconnus du robot jusqu'alors.

Le formalisme utilisé pour cette expérience robotique est le formalisme bayésien, c'est-à-dire la modélisation et l'inférence probabiliste. Nous utilisons la méthodologie de Programmation Bayésienne des Robots, dans laquelle s'exprime le modèle de représentation de l'espace appelé Carte Bayésienne. Dans notre expérience robotique, nous identifions expérimentalement une telle carte bayésienne, par application d'un comportement d'exploration aléatoire de l'espace sensori-moteur. La carte bayésienne apprise permet la résolution de différentes questions probabilistes générant des comportements variés.

Le reste de ce document est structuré comme suit. Le Chapitre 2 présente notre étude bibliographique et la replace dans une analyse des modèles du cycle perception – action. Le Chapitre 3 introduit notre expérience robotique en en présentant les principes généraux. Les détails liés à l'implantation de cette expérience sur notre plate-forme robotique Koala, ainsi que les résultats obtenus sont développés Chapitre 4. Enfin, le Chapitre 5 propose une synthèse du travail réalisé ainsi que les perspectives de recherche que nous avons dégagées.

# Chapitre 2

# Étude bibliographique de la représentation de l'espace pour la navigation en robotique mobile

## 2.1 Le cycle perception – action

Dans l'analyse bibliographique proposée dans sa thèse [6, 9], J. Diard discute des approches robotiques classiques de navigation. Il note que celles-ci se placent implicitement dans un même modèle de contrôle robotique, le cycle perception – action (voir Figure 2.1). Dans ce modèle, on suppose que l'algorithme de contrôle d'un robot est essentiellement composé de deux phases.

- La première phase consiste, dans un premier temps, à lire les informations fournies par les capteurs du robot. Ces capteurs fournissent des informations sur la localisation du robot dans son environnement. De celui-ci, le robot possède une représentation interne, typiquement sous la forme d'une carte, laquelle peut être préexistante à la navigation ou peut au contraire se construire au fur et à mesure de celle-ci. Le but de la première phase est de traiter les informations données par les capteurs pour en tirer une localisation ou un état du robot dans cette représentation interne.
- Dans la seconde phase, le robot utilise sa représentation interne de l'environnement pour en déduire, en fonction de ses buts, une action ou séquence d'actions à envoyer à ses actionneurs (moteurs). Dans le cadre de la navigation, cette phase est parfois séparée en deux : une première étape dans laquelle la représentation interne est utilisée pour générer des actions exprimées mathématiquement (planification) et une seconde étape dans laquelle le robot essaie d'appliquer ces actions dans l'environnement physique (suivi de trajectoire).

Dans son analyse, J. Diard remarque que ce modèle du cycle perception – action est effectivement couramment utilisé dans la littérature concernant la localisation et la construction de carte. Ainsi, les résultats des processus de localisation et de construction de carte sont destinés à être intégrés dans un algorithme de planification. On fait alors une hypothèse

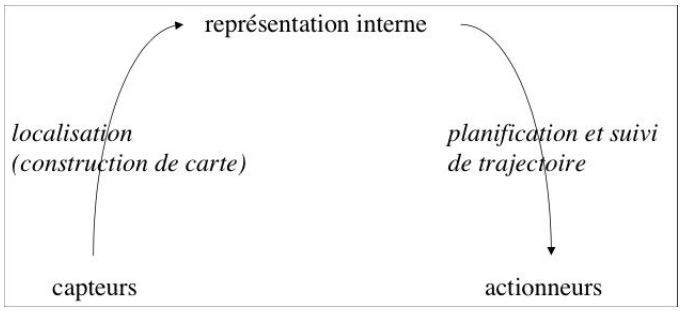



FIG. 2.1 – Modèle du cycle perception – action.

simplificatrice forte, en supposant que les deux phases peuvent être résolues indépendamment. Cette hypothèse est très intéressante, car elle permet de dissocier deux domaines de recherche : celui traitant des problèmes de localisation et construction de carte, d'une part, et celui traitant des problèmes de planification d'autre part. Étant donné cette séparation, une représentation interne (typiquement, une carte) la plus précise possible permet aux travaux en localisation de faciliter le processus de planification.

L'étude bibliographique que nous proposons ici est complémentaire de l'analyse de J. Diard qui montre que les techniques de localisation et de construction de carte s'intègrent généralement à ce modèle du cycle perception – action. En effet, nous souhaitons montrer à présent que l'essentiel des approches dans le domaine de la planification se placent elles aussi implicitement dans ce modèle. Ces travaux, comme ceux du domaine de la localisation, font l'hyphothèse d'indépendance des deux phases du cycle. Les algorithmes de localisation fournissant une position la plus précise possible, les approches en planification peuvent alors se permettre de supposer avoir la carte la plus précise possible dans laquelle calculer une trajectoire pour le robot.

Nous allons donc, dans la Section 2.2, passer en revue différentes approches proposées dans la littérature pour la planification, afin de montrer qu'elles s'intègrent effectivement dans ce modèle du cycle perception – action. Nous introduirons ensuite le formalisme de la programmation bayésienne des robots, qui est à la base de l'approche des cartes bayésiennes pour la représentation de l'espace et la navigation des robots mobiles (Section 2.3). Nous détaillerons enfin la perspective nouvelle offerte par cette approche sur le modèle du cycle perception – action (Section 2.4).

## 2.2 État de l'art du domaine de la planification

Nous proposons dans cette section un état de l'art du domaine de la planification qui, sans être exhaustif, fait le tour des principales approches de ce domaine. Chacune suppose qu'est donnée une représentation géométrique de l'environnement, dans laquelle les positions des obstacles sont précisément connues. Sont d'abord passées en revue les approches déterministes, puis les roadmaps probabilistes et enfin les Processus de Décision Markoviens Partiellement Observables (PDMPO).

### 2.2.1 Approches géométriques et déterministes

#### 2.2.1.1 Roadmap

Le but d'une roadmap est de représenter la connectivité de l'espace libre par des courbes à une dimension. Dans cette section, les méthodes déterministes pour obtenir une roadmap sont passées en revue. Il s'agit pricipalement de la méthode du graphe de visibilité et de la méthode du graphe de Voronoi.

**Graphe de visibilité** La méthode du graphe de visibilité est l'une des plus anciennes du domaine de la planification [19]. Elle s'applique dans une représentation de l'espace où les obstacles sont des polygones. Dans un graphe de visibilité, la connectivité de l'espace libre est représentée par des segments reliant les sommets des polygones. Cependant ces segments doivent appartenir à l'espace libre, c'est-à-dire qu'ils ne doivent pas couper un des obstacles.

Construire un graphe de visibilité consiste donc à construire un graphe dont l'ensemble des nœuds est l'ensemble des sommets des polygones représentant les obstacles, plus le point de départ et le but à atteindre. Les arcs de ce graphe sont tous les segments dans l'espace libre reliant deux des points correspondant à ces nœuds, que ce soit un sommet de polygone, le point de départ ou le but à atteindre (voir Figure 2.2). Ainsi, pour trouver un chemin entre départ et but, il suffit d'appliquer un algorithme classique de recherche dans un graphe.

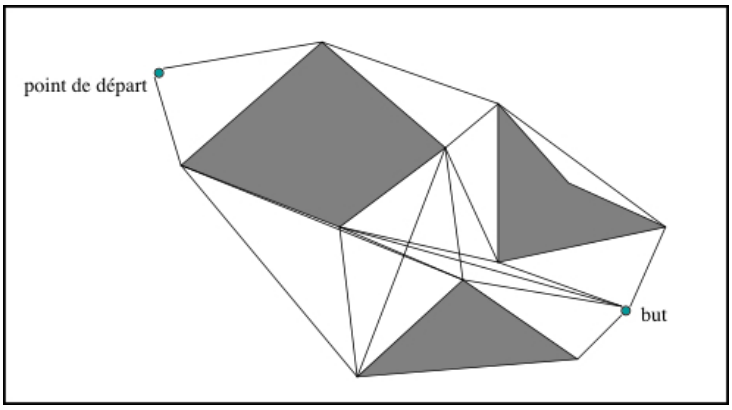



FIG. 2.2 – Exemple d'un graphe de visibilité. Tous les sommets des polygones, le point de départ et le but sont les nœuds du graphe. Tous les segments, notamment les arêtes des polygones, représentent les arcs du graphe.

**Diagramme de Voronoi** La méthode du diagramme de Voronoi [20] est principalement appliquée dans des espaces à 2 dimensions.

Le diagramme de Voronoi consiste en un ensemble de points de l'espace libre. Un point de l'espace libre appartient au diagramme s'il est le point le plus proche d'au moins deux

points qui, eux, appartiennent au contour d'un obstacle. Dans l'espace à deux dimensions, ce diagramme représente donc un ensemble de segments, droits ou courbes, qui passent exactement à équidistance des obstacles proches, comme le montre la Figure 2.3.

Pour chercher un chemin, il faut d'abord *rétracter* le point de départ et le but sur le diagramme. Par exemple, pour rétracter le départ, on cherche d'abord le point appartenant au contour d'un obstacle qui lui est le plus proche, puis on prend le point qui correspond à ce dernier dans le diagramme. Ce point, qui est le point de départ rétracté, et le point de départ sont lié par un segment droit. Le chemin cherché sera donc composé des deux segments droits reliant départ et but au diagramme, ainsi que du chemin dans le diagramme allant du point de départ rétracté au but rétracté.

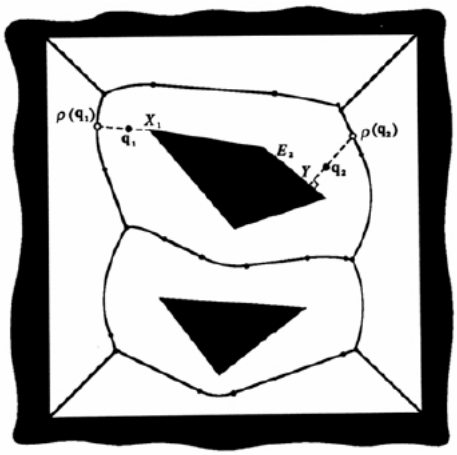



FIG. 2.3 – Exemple d'un diagramme de Voronoi. $q_1$ est le point de départ, $\rho(q_1)$ le point de départ rétracté, $q_2$ le but et $\rho(q_2)$ le but rétracté. Bien qu'ici les obstacles soient des polygones, ce n'est pas une condition nécessaire pour obtenir ce diagramme : il suffit de connaître les contours des obstacles. Figure reprise de [16].

**Autres méthodes** Il existe d'autres méthodes pour obtenir des roadmaps par approche déterministe et géométrique. Nous ne les détaillerons pas ici : ces méthodes consistent également à chercher un ensemble de courbes unidimensionnelles dans l'espace libre, afin de représenter la connectivité de ce dernier. Cet ensemble, moins riche et plus facilement manipulable que l'espace libre, se prête plus facilement à la recherche d'un chemin.

#### 2.2.1.2 Décomposition en cellules (*cell decomposition*)

**Décomposition simple** Cette décomposition consiste à découper l'espace libre en formes géométriques simples. Ces formes peuvent se situer toutes en espace libre, auquel cas la décomposition est exacte, ou certaines peuvent chevaucher un obstacle. Pour ce dernier cas, on considère que les formes chevauchant un obstacle ne doivent pas faire partie du chemin cherché.

Les décompositions exactes ne s'appliquent que sur des espaces où les obstacles sont des polygones. Les formes simples peuvent être des triangles, obtenus en reliant des sommets de polygones, ou des trapèzes, obtenus en faisant passer des droites parallèles par les sommets des polygones. Pour les décompositions non exactes, il suffit de superposer une grille à l'espace.

Une fois la décomposition faite, un graphe en est tiré représentant la connectivité des cellules qui sont dans l'espace libre. Le point de départ et le but sont liés aux cellules dans lesquelles ils se trouvent, puis un chemin entre ces deux cellules est cherché dans le graphe.

**Décomposition hiérarchique** La décomposition en quadtree, pour les espaces à deux dimensions, et la décomposition en octree, pour ceux à trois dimensions, sont toutes deux des décompositions hiérarchiques. Dans le premier cas, les cellules sont des carrés et dans le second, des cubes. Mise à part la différence de dimension, les deux approches sont fondamentalement les mêmes. Pour les quadtree, l'espace est divisé en 4 carrés égaux. Parmi ceux-ci, ceux contenant à la fois de l'espace libre et de l'espace occupé (présence d'obstacle) sont eux-mêmes divisés en 4 carrés égaux. On continue à diviser récursivement les carrés mixtes, jusqu'à atteindre le nombre de divisions maximal fixé. Les carrés qui se trouvent entièrement en espace libre ou entièrement à l'intérieur d'un obstacle ne sont pas divisés récursivement. En parallèle est construit un arbre dont chaque nœud correspond à un carré et pour lequel on sait s'il est en espace libre, dans un obstacle ou mixte. Les fils d'un nœud sont les 4 carrés donnés par la division du carré correspondant à ce nœud. La Figure 2.4 illustre la construction d'un quadtree. Pour les octree, le principe est le même, avec des divisions de cube en 8 cubes égaux.

Pour trouver un chemin une fois cette décomposition faite, il suffit de trouver une séquence de cellules libres dans l'arbre, qui relie la cellule contenant le point de départ et celle contenant le but.

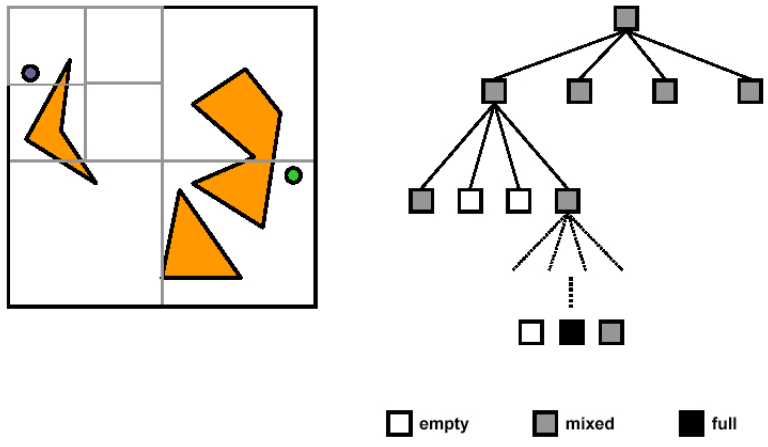



FIG. 2.4 – Exemple des premières étapes d'une décomposition en quadtree. Figure reprise d'un cours de David Hsu (*National University of Singapore*, 2003–2004).

#### 2.2.1.3 Champ de potentiel

Un champ de potentiel est une fonction scalaire qui couvre tout l'espace. Pour être utilisée en navigation, cette fonction doit généralement avoir un minimum global au point d'arrivée, un maximum global au point de départ, et tendre vers l'infini à l'abord des obstacles (voir Figure 2.5). La navigation du robot se fait alors par application d'une force proportionnelle à l'opposé du gradient du champ de potentiel. Idéalement, pour que cela marche bien, il faudrait que la fonction correspondante n'ait pas de minima locaux. Ce n'est généralement pas le cas, raison pour laquelle ont été proposées diverses méthodes pour permettre au robot d'échapper à un minimum local quand il s'y retrouve.

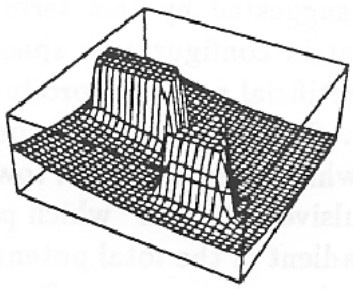


FIG. 2.5 – Exemple d'un champ de potentiel pour un espace à deux dimensions. Le point de départ se trouve dans le coin le plus à gauche, le but dans le coin le plus à droite, et il y a deux obstacles, l'un est un rectangle, l'autre un triangle. Figure reprise de [16].

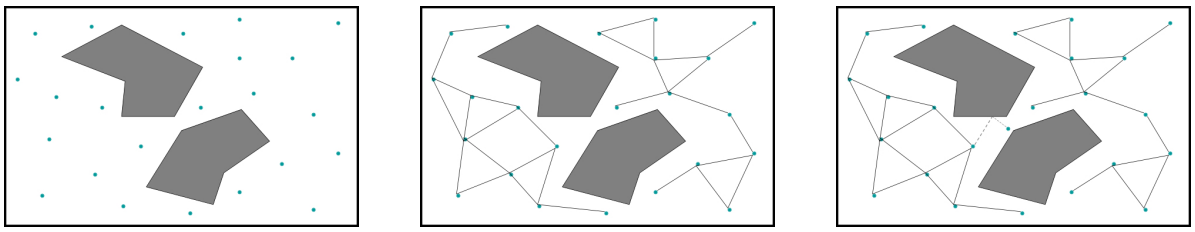
### 2.2.2 Les roadmaps probabilistes

Les roadmaps probabilistes ont été introduites par L. Kavraki et al. [15]. Comme pour les roadmaps déterministes, le but est de capturer la connectivité de l'espace libre par un ensemble de segments. Plus précisément, c'est un graphe qui est construit par cette méthode, dont les arcs correspondent à des segments dans l'espace libre.

La méthode pour construire une roadmap probabiliste est composée de trois phases. La première phase consiste à tirer, de manière uniforme, des points dans l'espace qui constitueront les nœuds du graphe. Sur les points tirés, seuls ceux en espace libre sont gardés. Le tirage continue jusqu'à ce que le nombre de points gardés atteigne la valeur voulue. À la fin de cette étape, un ensemble de nœuds a été construit pour le graphe (voir Figure 2.6(a)). Reste à construire les arcs, ce qui se fait pendant la seconde phase. Le but de celle-ci est de déterminer quels sont les points voisins qui peuvent être reliés entre eux dans l'espace libre. Pour savoir si un point peut être relié à un de ses voisins, un planificateur local est utilisé. Lorsque cela est possible, l'arc correspondant est rajouté au graphe (voir Figure 2.6(b)).

Le graphe ainsi obtenu peut ne pas bien représenter la connectivité de l'espace libre si celui-ci comporte des passages étroits. Il y a en effet un risque d'obtenir plusieurs sous-graphes non connexes si trop peu de points ont été tirés près des passages étroits. Pour éviter

cela, une troisième phase, dite d'expansion, peut être appliquée. Elle permet de rajouter des nœuds au graphe au voisinage de ces passages étroits. L'idée est de sélectionner un certain nombre de nœuds du graphe qui sont les plus susceptibles d'être dans ces régions et d'essayer de rajouter de nouveaux points de l'espace libre dans leur voisinage. Typiquement, ces nouveaux points peuvent être cherchés à partir du point déjà existant, en partant dans une direction aléatoire et en rebondissant sur les obstacles éventuellement rencontrés (voir Figure 2.6(c)).



(a) Tirage uniforme de points dans l'espace libre.

(b) Liaison des points voisins par un planificateur local.

(c) Phase d'expansion.

FIG. 2.6 – Exemple de construction d'une roadmap probabiliste.

Une fois construite, la roadmap peut être utilisée pour plusieurs requêtes. Pour une requête, on commence par relier le point de départ à un des points du graphe dans son voisinage, à l'aide du planificateur local ; on procède de même pour relier le but au graphe. Un chemin est ensuite cherché dans la roadmap, entre le point auquel on a relié le point de départ et celui auquel on a relié le but. On obtient alors un chemin dans le graphe qui est une séquence d'arcs. Ces arcs correspondent à des chemins trouvés par le planificateur local. Ainsi, le chemin global donné en réponse à la requête est la concaténation de chemins trouvés par le planificateur local. Au besoin, ce chemin global peut être amélioré par des techniques de lissage.

La méthode des roadmaps probabilistes a suscité bien des intérêts, dans différents domaines où la planification est nécessaire. Elle a même été utilisée pour chercher des configurations possibles pour des protéines [2]. De nombreux travaux ont proposé des variantes pour améliorer son efficacité. Notamment, la faiblesse de cette méthode réside dans sa difficulté à représenter la connectivité de l'espace en présence de passages étroits. Ainsi de nouvelles méthodes ont été proposées pour améliorer la première phase de tirage des nœuds, que ce soit en tirant ceux-ci sur le diagramme de Voronoi [23] ou le plus près possible du contour des obstacles [1, 5], ou encore en permettant une légère pénétration dans les obstacles pour ensuite repositionner les nœuds ne se trouvant pas en espace libre [11]. La construction de la roadmap définie à la base est assez coûteuse pour constituer une phase de pré-calcul et se destine à être utilisée pour plusieurs requêtes de recherche de chemin. Certains travaux ont donc proposé des méthodes « allégées », conçues pour être mises en œuvre pour une seule requête [4, 12].

### 2.2.3 Localisation Markovienne

La localisation markovienne est une méthode probabiliste de représentation de l'espace [22]. Elle est généralement utilisée pour construire une grille d'occupation qui représente l'environnement. Une carte de ce type est donc un découpage régulier de l'espace en cases. Leur taille est généralement assez petite, de l'ordre de 50 cm de côté. Pendant la construction de la carte, est assignée à chaque case la probabilité qu'elle soit occupée par un obstacle (voir Figure 2.7). Nous ne détaillerons pas ici cette construction, ni les détails de l'écriture probabiliste du modèle qui, eux, sont présentés Section 2.3.1.2.

En revanche, nous exposons la façon dont est utilisée cette grille d'occupation pour contrôler le robot. En effet, bien qu'il s'agisse d'une représentation discrète de l'environnement, elle n'en occupe pas moins un espace mémoire quadratique en fonction de la longueur de l'environnement : pour un espace carré de 50 m de côté et avec des cellules de 50 cm de côté, il faut $100^2 = 10^4$ cellules. Il faut donc résumer cette carte trop précise pour l'utiliser dans le cadre de la planification. Nous présentons ci-dessous deux classes de méthodes qui s'attachent à ce problème. Il s'agit de l'extraction d'une carte topologique, applicable pour une planification déterministe, et des PDMPO, applicables pour une planification probabiliste.

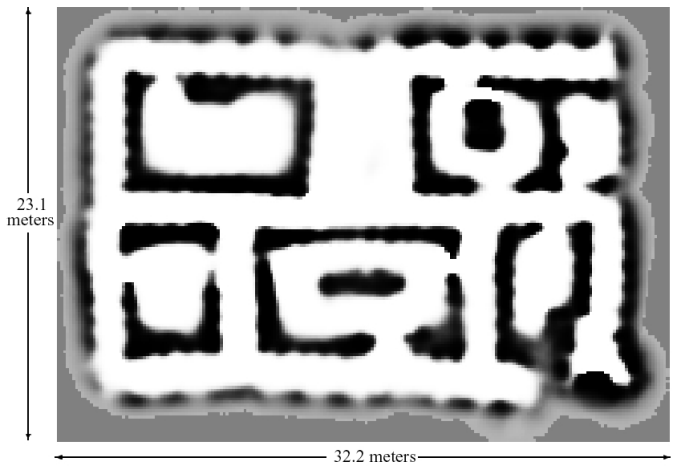


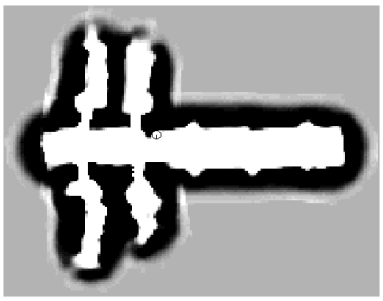

FIG. 2.7 – Exemple d'une carte de type grille d'occupation générée par la localisation markovienne. Les différentes cases ne sont pas nettement distinguables car il s'agit d'une grille très fine. Le niveau de gris des cases représente leur probabilité d'être occupées : les cases blanches sont très probablement libres, les cases noires sont très probablement occupées. Figure reprise de [21].

#### 2.2.3.1 Extraction d'une carte topologique

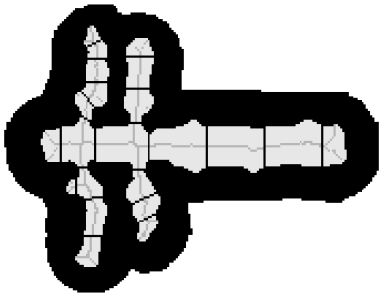
Pour utiliser une grille d'occupation construite par localisation markovienne, dans le cadre de la planification, une solution consiste à extraire un graphe topologique de cette grille. Ce graphe peut être tiré de la grille de la manière suivante [21] :

– calcul du diagramme de Voronoi pour la grille ;
– utilisation de certaines propriétés de ce diagramme (points et lignes critiques) pour définir des régions ;
– construction du graphe : un nœud est associé à chaque région et des arcs relient les nœuds de régions adjacentes.

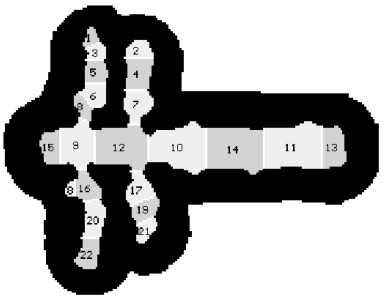
Ces étapes sont représentées Figure 2.8. Le graphe résultant encode les relations topologiques entre les zones de l'environnement, donc capture la connectivité de l'espace. Il peut ainsi être utilisé pour répondre à des requêtes de navigation, avec un des algorithmes classiques de recherche de chemin dans un graphe.



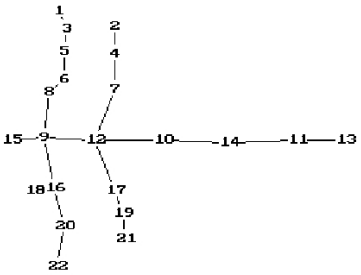
(a) Grille d'occupation initiale.



(b) Diagramme de Voronoi.



(c) Régions extraites du diagramme de Voronoi.



(d) Graphe topologique.

FIG. 2.8 – Extraction d'un graphe topologique à partir d'une grille d'occupation construite par la méthode de localisation markovienne. Figure reprise de [21].

#### 2.2.3.2 Processus de Décision Markoviens Partiellement Observables (PDMPO)

Une grille d'occupation basée sur la localisation markovienne peut aussi servir à définir un processus de planification probabiliste, nommé Processus de Décision Markovien

Partiellement Observable (PDMPO, [14]). En effet, la localisation markovienne est un formalisme probabiliste pour représenter l'environnement. Pour obtenir un PDMPO, il faut ajouter à ce formalisme une description du but à atteindre. Pour cela, on définit une fonction de coût ou de récompense. Cette fonction associe à chaque lieu de l'environnement une récompense ou un coût représentés par un nombre réel. Par exemple, il est classique d'associer une récompense de 0 au but et de -1 aux autres lieux. Il s'agit ensuite de calculer une séquence d'actions permettant de maximiser la récompense (respectivement, de minimiser le coût) accumulée pendant le déplacement. Cette maximisation se mène avec des algorithmes comme *policy iteration* ou *value iteration*.

Le problème de ces algorithmes est qu'ils impliquent une explosion combinatoire due à la prise en compte de tous les états de connaissance possibles du robot (représentés par toutes les distributions de probabilité possibles sur la position du robot dans l'environnement). Cela limite énormément la taille des espaces pour lesquels ces méthodes sont applicables. Par exemple, un PDMPO classique ne peut être utilisé pour un environnement de plus d'une centaine d'états (à comparer aux $10^4$ états couramment utilisés dans la localisation markovienne). Il existe donc un grand nombre de techniques pour rendre ces méthodes utilisables en pratique.

**Processus de Décision Markoviens complètement observables (PDM)** Une première méthode consiste à approximer le PDMPO initial, en supposant que le robot sait exactement sa position dans l'environnement. On fait alors l'hypothèse qu'il dispose d'un capteur résolvant le problème de localisation. La position du robot étant ainsi directement accessible, le processus de décision markovien est dit complètement observable. Un algorithme de planification tel que value iteration devient alors quadratique, ce qui le rend abordable pour des environnements de taille moyenne (quelques dizaines de mètres de largeur ou longueur). Au delà, des méthodes approximées sont nécessaires.

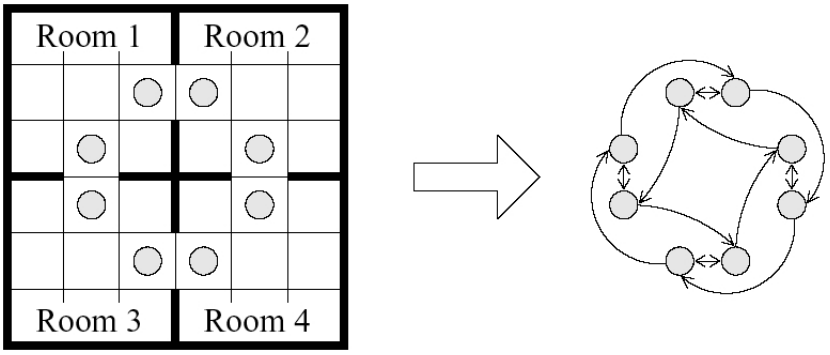



FIG. 2.9 – Extraction de macro-états dans un PDM. Chaque pièce contient initialement 9 états qui sont abstraits par deux macro-états (les ronds grisés). Ces macro-états sont reliés par des macro-actions (flèches du graphe de droite). Figure reprise de [10].

**PDMPO hiérarchiques** Une seconde méthode pour appliquer en pratique un PDMPO est d'en extraire un PDMPO de plus haut niveau hiérarchique [10]. L'objectif est que ce PDMPO abstrait contient moins d'états que le PDMPO initial. On peut ainsi extraire des PDMPO dont les états représentent soit plusieurs lieux du PDMPO initial (macro-états, voir Figure 2.9), soit une séquence d'actions du PDMPO initial (macro-actions). Le calcul d'un chemin vers le but voit ainsi sa complexité réduite, ce qui permet d'utiliser le PDMPO abstrait en pratique. En revanche, l'automatisation de l'extraction des macro-états ou macro-actions reste un problème très difficile.

## 2.3 Modélisation et inférence bayésienne pour la navigation

### 2.3.1 La programmation bayésienne des robots (PBR)

Dans cette section sont présentées diverses approches probabilistes qui s'appuient sur la modélisation et l'inférence bayésiennes et qui s'appliquent notamment en robotique mobile. Tout d'abord est exposée la structure globale qu'adoptent ces méthodes, dans le cadre de la programmation des robots (étant donné que cette étude concerne la navigation des robots mobiles). Dans une seconde partie, les diverses approches sont exposées.

#### 2.3.1.1 Structure globale

La structure globale de programmation bayésienne des robots présentée ci-dessous est issue des travaux de O. Lebeltel durant sa thèse [17]. En effet, il définit dans celle-ci une méthodologie de programmation qui lui sert à exécuter diverses expériences en robotique. Un travail de bibliographie [7, 3] a ensuite permis de constater que cette méthodologie donnait une structure dans laquelle se réécrivent l'essentiel des outils de modélisation probabiliste.

**Notions préliminaires** Voici d'abord quelques notions sur les probabilités et l'inférence bayésienne. Nous ne reviendrons pas ici sur les fondements de l'utilisation des probabilités pour représenter un état de connaissance. Le physicien E. T. Jaynes est à l'origine de la théorie du raisonnement plausible qui montre comment passer de la notion intuitive de plausibilité à la notion mathématique de probabilité, au moyen de quelques désidérata de base [13]. Nous renvoyons donc le lecteur à ces travaux pour de plus amples informations sur le sujet. Sont simplement exposées ici les briques de bases permettant de comprendre la modélisation et l'inférence bayésiennes.

**Variables :** Soit $V$ une variable, $\mathcal{D}_V = \{v_1, \ldots, v_k\}$ un domaine de valeurs. $V$ est associée aux $k$ propositions logiques $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k$ :

$$\begin{aligned} \mathcal{V}_1 &\equiv [V = v_1] \\ \mathcal{V}_2 &\equiv [V = v_2] \\ &\vdots \\ \mathcal{V}_k &\equiv [V = v_k]. \end{aligned}$$

Chacune de ces propositions indique la valeur prise par la variable ; elles sont donc exhaustives ($\mathcal{V}_1 + \cdots + \mathcal{V}_k = vrai$) et mutuellement exclusives ($\forall i, j, i \neq j, \mathcal{V}_i \mathcal{V}_j = faux$). Ainsi, nous définirons les variables en donnant leur domaine $\mathcal{D}_V$ et le cardinal de ce domaine $k_V$. Prenons l'exemple d'une variable Température. Les propositions logiques sont :

$$\begin{aligned} \mathcal{V}_1 = \text{« la température est de -20 degrés »} &\equiv [\text{Température} = -20] \\ \mathcal{V}_2 = \text{« la température est de -19 degrés »} &\equiv [\text{Température} = -19] \\ &\vdots \\ \mathcal{V}_{61} = \text{« la température est de 40 degrés »} &\equiv [\text{Température} = 40]. \end{aligned}$$

La définition de la variable est :

$$\text{Température } : \mathcal{D}_{\text{Température}} = [-20, +40], k_{\text{Température}} = 61.$$

En présence de deux variables $V_1$ et $V_2$ de domaines $\mathcal{D}_{V_1}$ et $\mathcal{D}_{V_2}$, nous pouvons construire la nouvelle variable $V = V_1 \wedge V_2$, associée à toutes les propositions logiques du type :

$$\mathcal{V}_{1_i} \mathcal{V}_{2_j} \equiv [V_1 = v_{1_i}][V_2 = v_{2_j}],$$

$i$ et $j$ variant dans les domaines respectifs des variables. Ces propositions dénotent « la variable $\mathcal{V}_1$ a pour valeur $v_{1_i}$ et la variable $\mathcal{V}_2$ a pour valeur $v_{2_j}$ » ; nous remarquons qu'elles restent exhaustives et mutuellement exclusives[1].

**Règle du produit [R1] :** La règle du produit permet d'exprimer la probabilité conjointe de deux termes (propositions ou variables) comme le produit de deux probabilités élémentaires. Dans le cas de propositions, [R1] prend la forme[2] :

$$\begin{aligned} P(\mathcal{A}\, \mathcal{B} \mid \pi) &= P(\mathcal{A} \mid \pi) P(\mathcal{B} \mid \mathcal{A}\, \pi) \\ &= P(\mathcal{B} \mid \pi) P(\mathcal{A} \mid \mathcal{B}\, \pi). \end{aligned} \tag{2.1}$$

[1] Par contre, en ce qui concerne la disjonction, les propositions ne restent pas mutuellement exclusives.

[2] Dans les équations suivantes, le lecteur pourra noter que le symbole $\pi$ apparaît toujours en partie droite des probabilités. Ce symbole représente les connaissances préalables qui spécifient la distribution de ces probabilités. Par exemple, une variable $\mathcal{A}$ peut avoir deux définitions différentes pour deux ensembles de connaissances préalables différents.

Les deux formes possibles proviennent bien évidemment de la commutativité de la conjonction logique. Une dérivation élémentaire permet d'arriver à la forme de [R1] pour les variables :

$$\begin{aligned} P(X\ Y \mid \pi) &= P(X \mid \pi)P(Y \mid X\ \pi) \\ &= P(Y \mid \pi)P(X \mid Y\ \pi). \end{aligned} \tag{2.2}$$

Cette règle est également appelée le *théorème de Bayes*.

**Règle de normalisation [R2] :** La règle de normalisation est la deuxième règle fondamentale nécessaire pour l'inférence. De façon informelle, elle exprime le fait que la somme des probabilités de tous les cas possibles d'un terme vaut 1. Dans le cas des propositions, [R2] prend la forme :

$$P(\mathcal{A} \mid \pi) + P(\neg\mathcal{A} \mid \pi) = 1, \tag{2.3}$$

alors que pour les variables on écrit :

$$\forall X\ : \mathcal{D}_X, k_X, \sum_X P(X \mid \pi) = 1. \tag{2.4}$$

**Autres règles dérivables des précédentes :** Tout comme en logique où les opérateurs de conjonction et de négation permettent d'exprimer toute expression logique, les deux règles [R1] et [R2] forment la base sur laquelle tous les calculs de probabilités se dérivent. En particulier, de ces deux règles, nous pouvons dériver la *règle de la somme* :

$$\begin{gathered} \forall X\ : \mathcal{D}_X, k_X, \forall Y\ : \mathcal{D}_Y, k_Y, \forall x_i \in \mathcal{D}_X, \forall y_i \in \mathcal{D}_Y, \\ P(\mathcal{X}_i + \mathcal{Y}_i \mid \pi) = P(\mathcal{X}_i \mid \pi) + P(\mathcal{Y}_i \mid \pi) - P(\mathcal{X}_i\ \mathcal{Y}_i \mid \pi), \end{gathered} \tag{2.5}$$

où, rappelons le, $\mathcal{X}_i$ dénote la proposition $[X = x_i]$ et $\mathcal{Y}_i$ dénote $[Y = y_i]$. Nous pouvons aussi dériver la *règle de marginalisation* :

$$\forall X\ : \mathcal{D}_X, k_X, \forall Y\ : \mathcal{D}_Y, k_Y, \sum_X P(X\ Y \mid \pi) = P(Y \mid \pi). \tag{2.6}$$

**Méthode** À présent, nous passons en revue les différentes étapes de la programmation bayésienne des robots [18] qui sont schématisées Figure 2.10.

**Le programme** se décompose en une phase description et une phase question. Cette dernière peut également être appelée phase d'utilisation car la description y est exploitée pour répondre à une question.

**La description** est dénotée formellement par la distribution de probabilité conjointe $P(V_1\ \ldots\ V_n \mid \delta\ \pi)$ d'un ensemble de variables $V_1, \ldots, V_n$, déterminée au vu des connaissances préalables $\pi$ spécifiées par le programmeur et d'un ensemble de données expérimentales $\delta$. Ainsi, l'ensemble des connaissances devant êtres fournies se trouve circonscrit par les connaissances préalables $\pi$ et le jeu de données $\delta$.

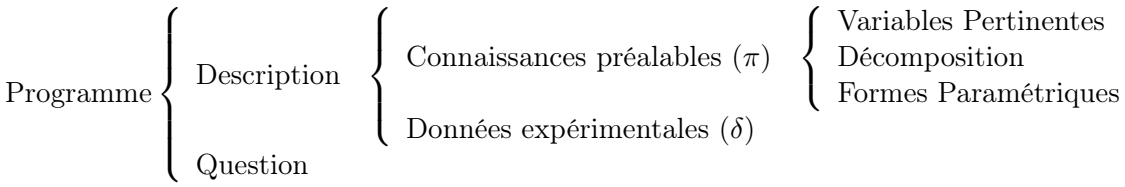



FIG. 2.10 – Structure d'un programme PBR.

**Les connaissances préalables** sont déterminées au cours la phase de spécification qui est la partie la plus délicate du travail du programmeur. Au cours de cette phase, il doit énoncer clairement et *explicitement* les connaissances dont il est à l'origine et celles qui résultent d'un processus adaptatif dépendant d'un jeu particulier de données expérimentales. Ces connaissances se subdivisent en trois : 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.

**Les variables pertinentes** correspondent aux *connaissances préalables structurelles*, qui permettent de définir l'ensemble des variables $V_1, \ldots, V_n$ pour la description et de spécifier pour chacune d'elles son domaine de variation $\mathcal{D}_{V_i}$ et son nombre $k_{V_i}$ d'états possibles. 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, les variables motrices, et enfin les variables internes, qui permettent de coder les états internes du robot.

**La décomposition** correspond aux *connaissances préalables de dépendance*. 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, qui n'est souvent pas facile à spécifier. La règle du produit [R1] nous permet de décomposer cette expression, en l'exprimant sous forme de produit de distributions.

Une fois une décomposition choisie, une seconde étape permet de simplifier davantage l'expression choisie par des *hypothèses d'indépendance conditionnelle*, ce qui réduit fortement les dimensions des termes sur lesquelles ces hypothèses portent.

Prenons l'exemple d'une distribution $P(V_1\ V_2\ V_3 \mid \delta\ \pi)$, pour laquelle a été choisie la décomposition :

$$P(V_1 \mid \delta\ \pi)P(V_2 \mid V_1\ \delta\ \pi)P(V_3 \mid V_1\ V_2\ \delta\ \pi).$$

S'il se trouve que l'on sait que $V_3$ est indépendant de $V_1$ si l'on connait $V_2$, alors le dernier terme peut être remplacé par $P(V_3 \mid V_2\ \delta\ \pi)$, selon cette hypothèse d'indépendance conditionnelle.

**Une forme paramétrique** est associée à chacun des termes apparaissant dans la décomposition choisie à l'étape précédente. Ces choix définissent des a priori sur les

valeurs des distributions de probabilité et la manière dont ces valeurs seront modifiées, éventuellement, par l'expérience. Quelques formes paramétriques simples sont la loi uniforme, la loi dirac, la loi normale ou gaussienne et la loi de succession de Laplace.

**Des données expérimentales,** représentées par le symbole $\delta$, peuvent être nécessaires pour fournir les valeurs des paramètres numériques de certaines formes paramétriques. Par exemple, dans le cas des gaussiennes, on peut tirer de ces données leurs moyennes et écarts type. Cette phase est la phase d'*identification* ou encore la période d'*apprentissage*.

**Pour la question,** les variables sont divisées en trois sous-ensembles : celles dont on cherche les valeurs ($Cherchées$), celles dont on connaît les valeurs ($Connues$) et celles dont on ne connaît pas les valeurs sans pour autant les chercher ($Inconnues$). Une question est alors définie pas la distribution :

$$P(Cherchées \mid Connues\ \delta\ \pi).$$

#### 2.3.1.2 Vue d'ensemble des différentes approches

Nous ne souhaitons pas détailler ici toutes les approches en modélisation et inférence bayésiennes qui peuvent se réécrire dans la structure PBR détaillée ci-dessus. La Figure 2.11 montre l'ensemble de ces approches et la façon dont elles sont apparentées.

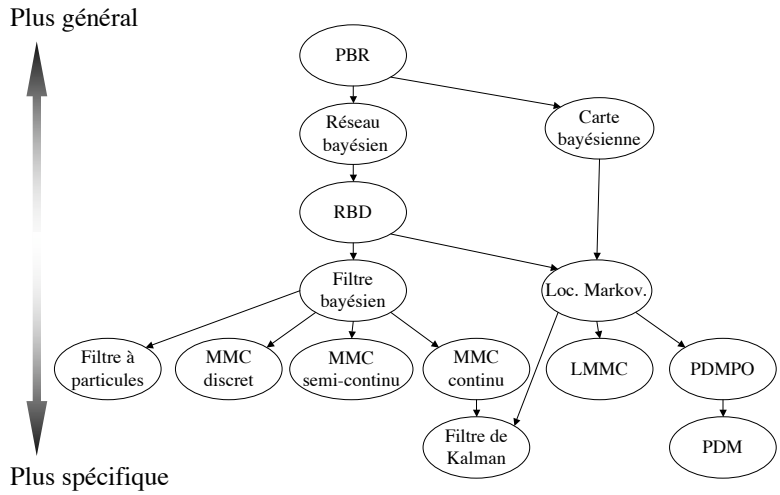



FIG. 2.11 – Ensemble des approches en modélisation et inférence bayésiennes pouvant se réécrire dans le formalisme PBR. RBD signifie Réseau Bayésien Dynamique, MMC signifie Modèle de Markov Caché, Loc. Markov. signifie Localisation Markovienne, LMMC signifie Localisation Markovienne à base de Monte Carlo, PDMPO signifie Processus de Décision Markovien Partiellement Observable et PDM signifie Processus de Décision Markovien [7].

Par contre, puisque la localisation markovienne et les POMDPs ont été présentés Sections 2.2.3 et 2.2.3.2, nous détaillons ici la réécriture de ces approches dans le formalisme PBR (voir Figure 2.12).

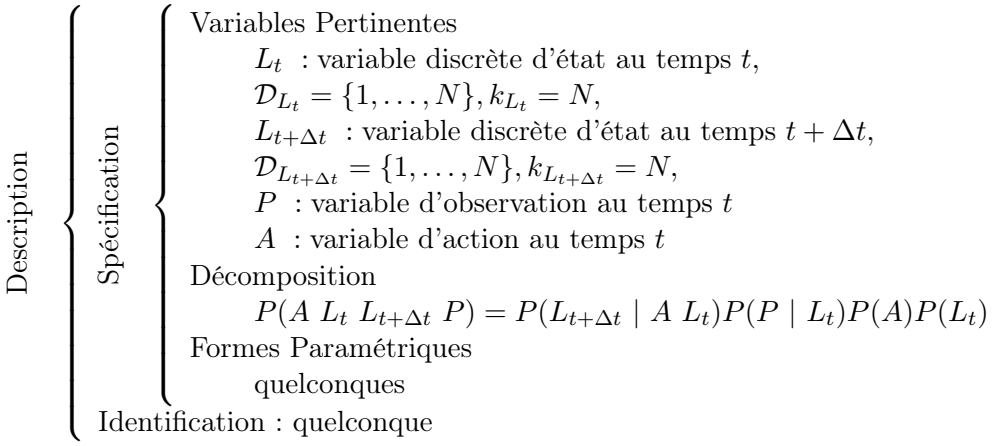



FIG. 2.12 – Le formalisme de la localisation markovienne exprimé en PBR. À ce modèle probabiliste, il faut ajouter une fonction de récompense $R : L_t \rightarrow \mathbb{R}$ pour obtenir la définition du formalisme des PDMPOs.

Un modèle de localisation markovienne est un modèle probabiliste concernant quatre variables :

- une variable d'observation $P_t$ au temps $t$ ;
- une variable d'action $A_t$ au temps $t$ ;
- une variable $L_t$ décrivant l'état du robot dans l'environnement au temps $t$. Cette variable représente en général la position $x, y$ et l'orientation du robot $\theta$, discrétisées et replacées dans la grille d'occupation ;
- enfin, une variable d'état $L_{t+\Delta t}$ au temps futur $t + \Delta t$.

Il s'agit ensuite de définir la distribution de probabilité conjointe $P(A\ L_t\ L_{t+\Delta t}\ P)$. La décomposition utilisée fait intervenir deux termes particuliers. Le premier est $P(L_{t+\Delta t} \mid A\ L_t)$ et décrit l'évolution de la position du robot dans l'environnement en fonction des ordres moteurs et de sa connaissance de sa position passée : c'est le modèle de transitions. Le second est $P(P \mid L_t)$ et décrit les observations attendues en fonction de la position du robot dans l'environnement : c'est le modèle d'observation. Cependant, le formalisme de localisation markovienne ne contraint pas la définition de ces modèles, qui peuvent donc prendre des formes paramétriques variées.

La question posée au modèle de localisation markovienne est une question de *localisation*. On cherche à calculer la distribution de probabilité sur la position du robot, sachant la dernière action et la dernière observation, tout en supposant connaître la distribution sur la position au pas de temps précédent : $P(L_{t+\Delta t} \mid A\ P)$.

Le modèle de PDMPO consiste à ajouter une fonction de récompense au modèle de la localisation markovienne. Cette fonction est utilisée pour résoudre une question de *plani-*

*fication*, c'est-à-dire pour calculer la distribution de probabilité sur la séquence d'actions futures, sachant que l'on veut maximiser la récompense acquise.

### 2.3.2 La carte bayésienne

Dans cette section est présentée en détail la carte bayésienne. Cette approche, qui a été développée par J. Diard durant sa thèse [6], a déjà été introduite dans la Figure 2.11. Comme cette figure le montre, la carte bayésienne est une généralisation de la localisation markovienne.

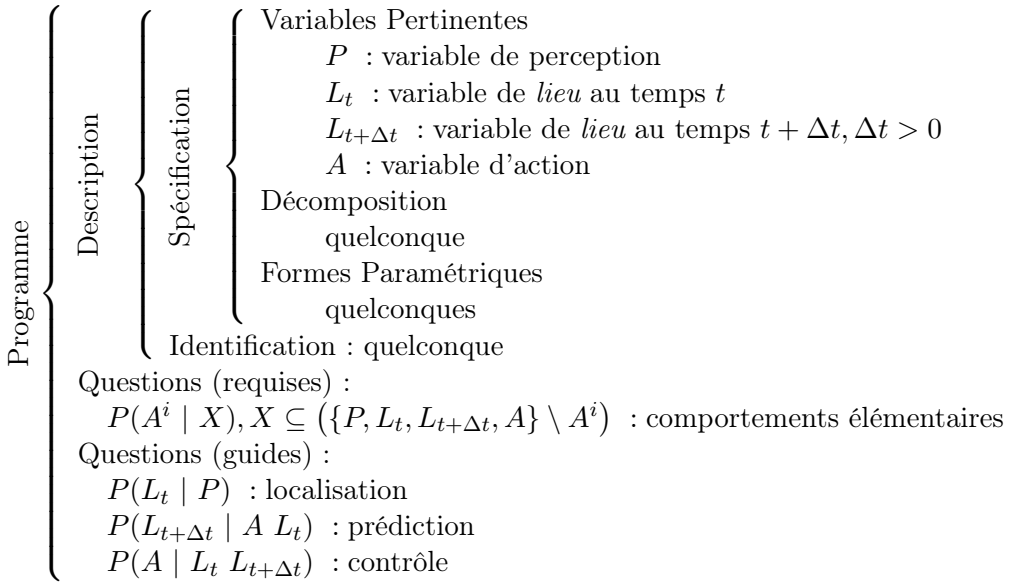



FIG. 2.13 – La définition d'une carte bayésienne dans le formalisme PBR.

En effet, comme pour la localisation markovienne, les variables pertinentes consistent en une variable $P$ de perception (ou d'observation), une variable $L_t$ de lieu au temps $t$, une variable $L_{t+\Delta t}$ de lieu au temps $t + \Delta t$ et une variable $A$ d'action (voir Figure 2.13). En revanche, le choix de la décomposition de la conjointe $P(A\ L_t\ L_{t+\Delta t}\ P)$ n'est pas fixé. De plus, une carte bayésienne ne sert pas seulement à se localiser ou à planifier : on doit pouvoir tirer des *comportements* de ce modèle. Un comportement est défini par une question probabiliste du type $P(A^i \mid X)$, où $A^i$ est un sous-ensemble des variables d'action et $X$ un sous-ensemble des variables du modèle. Cependant, dans le cadre de ce projet, nous nous restreindrons à des comportements définis par des questions de la forme $P(A \mid P\ L_{t+\Delta t})$ : sachant les perceptions, sachant le lieu que l'on souhaite atteindre, quelle est la distribution de probabilité sur les actions qui peuvent être appliquées ?

## 2.4 Conclusion sur l'étude bibliographique

### 2.4.1 Dégradation nécessaire d'une représentation trop précise en planification

L'étude bibliographique précédente montre que l'essentiel des approches dans le domaine de la planification font l'hypothèse d'indépendance entre les deux phases du modèle du cycle perception – action présenté Section 2.1. De ce fait, ces approches supposent avoir initialement une représentation interne de l'environnement très riche. Nous avons effectivement vu ci-dessus que la première étape des solutions proposées au problème de planification est souvent de *dégrader* cette représentation trop précise.

Sans cette étape, la représentation reste difficilement manipulable pour la recherche d'un chemin. Une structure discrète tel un graphe se prête plus facilement à l'application d'un algorithme de recherche classique (A* par exemple) qu'une structure continue. Ainsi le problème de dégradation prend souvent la forme d'un problème de *discrétisation* de l'espace. C'est le cas pour la décomposition en cellules et pour le graphe de visibilité (Section 2.2.1.2), ainsi que pour les roadmaps probabilistes (Section 2.2.2).

Les approches à base de localisation markovienne, quant à elles, partent déjà d'un modèle discret. Cependant cette représentation est encore trop riche pour être manipulée, comme nous l'avons vu Section 2.2.3. Par exemple, une grille d'occupation trop précise peut être résumée sous la forme d'un graphe : le problème de dégradation prend alors la forme d'un problème de *résumé* de l'information contenue dans la représentation initiale. C'est aussi le cas du diagramme de Voronoi, qui résume l'environnement en courbes unidimensionnelles (Section 2.2.1.1), et de la méthode du champ de potentiel, qui le résume en une fonction (Section 2.2.1.3).

Ainsi, le principal problème de l'essentiel des approches en planification est de **dégrader** une représentation initiale trop précise et donc difficilement manipulable. Ce fait vient, selon nous, de l'hypothèse d'indépendance dont nous avons parlé au début de cette section. Le modèle du cycle perception – action présenté Figure 2.1 sépare localisation d'une part et planification d'autre part mais demande, pour relier les deux, à ce que la représentation interne soit unique et la plus précise possible.

C'est pourquoi nous proposons, dans la section suivante, une remise en question de ce modèle, qui est inspirée par les possibilités qu'offre le formalisme de carte bayésienne.

### 2.4.2 Remise en question du cycle perception – action

Le formalisme de carte bayésienne se place dans un autre modèle du cycle perception – action. Dans ce formalisme, la représentation que le robot a de l'environnement n'est pas nécessairement unique, mais peut être composée de multiples représentations, plus simples, chacune décrivant une partie de l'interaction sensorimotrice.

En effet, le formalisme de carte bayésienne inclut la définition d'opérateurs d'assemblage de cartes bayésiennes. Ces opérateurs, la Superposition et l'Abstraction de cartes

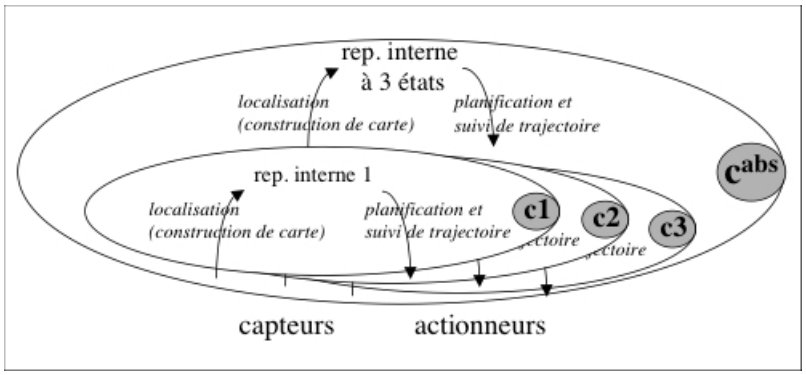



FIG. 2.14 – Modèle du cycle perception – action dans le cadre de l'abstraction de cartes bayésiennes. Les trois cartes bayésiennes $c1$, $c2$ et $c3$ sont utilisées comme ressources pour la construction d'une carte bayésienne $c^{abs}$.

bayésiennes, permettent la définition de nouvelles cartes obtenues à partir d'un ensemble de cartes. On obtient ainsi des hiérarchies de cartes bayésiennes.

Nous montrons Figure 2.14 un exemple de hiérarchie de cartes bayésiennes. Dans cette hiérarchie, les trois cartes $c1$, $c2$ et $c3$ sont utilisées pour la création de la carte $c^{abs}$, via l'opérateur d'abstraction. Chacune des cartes de cette hiérarchie est un modèle d'une partie de l'interaction sensorimotrice. Pour permettre la définition de comportements, chaque carte doit résoudre les problèmes de localisation et de planification. Ces problèmes sont donc traités par toutes les représentations de la hiérarchie, qui décrivent chacune une petite partie de l'environnement. La carte bayésienne abstraite de notre exemple doit se localiser et planifier dans un espace à trois états seulement [3]. La localisation et la planification dans un espace à trois états discrets sont deux problèmes très simples.

### 2.4.3 Synthèse

L'étude bibliographique précédente nous a permis de montrer que l'essentiel des approches en planification se confrontent à un problème de dégradation d'une représentation initiale trop précise. Nous avons positionné ce problème par rapport au modèle du cycle perception – action classique : dans celui-ci, la représentation interne est la charnière entre localisation et planification, qui sont deux étapes distinctes. Pour assurer un lien correct entre ces deux étapes, la représentation doit être la plus fine possible. En effet, bien que les approches en planification doivent dégrader cette représentation, il leur est nécessaire de connaître précisément la position des obstacles (se référer, par exemple, au problème des passages étroits) ; et de fait, les chemins trouvés peuvent passer très près de leurs contours.

Cette étude est complémentaire de celle de J. Diard, qui montre que l'essentiel des problèmes en localisation sont dus à l'objectif de créer une représentation très fine de l'environnement. Nous avons donc introduit le formalisme de carte bayésienne développé

[3] Un état pour chaque carte bayésienne sous-jacente. Le lecteur intéressé pourra se référer à [6, 8] pour plus de détails.

par ce dernier, et brièvement décrit en quoi ce formalisme remet en cause le modèle classique du cycle perception – action.

Le projet que nous avons mis au point et réalisé se propose d'illustrer plus avant la remise en question de ce modèle. En effet, nous y proposons une expérience robotique, dans laquelle nous étudierons la relation entre un schéma sensori-moteur et des comportements de navigation. Nous montrerons que, dans le cadre du formalisme des cartes bayésiennes, il n'est pas nécessaire de construire une représentation métrique fine d'un environnement pour y réaliser certaines tâches de navigation. Nous axerons également notre travail sur l'apprentissage expérimental d'une telle carte bayésienne.

# Chapitre 3

# Présentation de l'expérience robotique

Nous souhaitons à présent décrire l'expérience robotique que nous avons réalisé. Ce chapitre n'en donne que les principes généraux. Les détails liés à sa mise en œuvre sur notre plate-forme robotique sont exposés dans le chapitre suivant. Nous donnons d'abord une description de notre expérience sous la forme d'un petit scénario, les détails mathématiques étant donnés par la suite.

## 3.1 Description

L'objectif de ce projet est de montrer qu'il est possible, dans le cadre de la navigation mobile des robots, d'apprendre une carte à partir d'un comportement connu, puis d'exploiter cette carte pour en tirer de nouveaux comportements. Plus précisément, nous avons mis en œuvre le scénario suivant.

Un robot sait appliquer un comportement. On demande donc au robot d'appliquer celui-ci, ce qui le fait évoluer dans son environnement. Au cours de ses mouvements, on enregistre ce qu'observe le robot, à travers ses capteurs. Ces observations comportent des informations sur les effets d'une action sur la position du robot dans l'environnement. Nous pouvons les exploiter pour en construire une représentation interne, sous la forme d'une carte bayésienne. Une telle carte comporte les connaissances corrélées sur les actions et les perceptions du robot. En d'autres termes, cette carte permet d'estimer les conséquences d'une action sur la position du robot au vu des données sensorielles. Elle rend donc également possible le choix d'une action pour se rapprocher d'un but dans l'environnement.

Ainsi, la carte bayésienne apprise est un modèle dont peuvent être tirés des comportements. En effet, puisqu'une telle carte permet de sélectionner une action pour se rapprocher d'un but, elle doit nous permettre de réaliser différentes tâches de navigations, qui prennent ici la forme de comportements. Nous nous attacherons donc à chercher quels nouveaux comportements nous pourrons en tirer.

En résumé, l'application d'un comportement initial connu permet au robot d'explorer son environnement ; cette exploration fournit les informations nécessaires à la construction d'une représentation interne de cet environnement, où sont contenues des connaissances

sur les corrélations entre actions et perceptions ; la représentation interne ainsi obtenue permet de faire appliquer au robot des comportements qui lui étaient jusqu'alors inconnus.

## 3.2 Construction d'une carte bayésienne à partir d'observations

Dans cette section, nous détaillons comment une carte bayésienne peut être construite à partir de données d'observations enregistrées par le robot.

Il convient d'abord de préciser quelle va être la décomposition choisie. En effet, comme nous l'avons vu Section 2.3.2, le formalisme des cartes bayésienne laisse ce choix à l'utilisateur. Dans le cadre de notre projet, la décomposition choisie est la suivante :

$$P(P\ L_t\ L_{t+\Delta t}\ A) = P(L_t)P(A)P(P \mid L_t)P(L_{t+\Delta t} \mid A\ L_t),$$

où, nous le rappelons, $P$ est la variable de perception au temps $t$, $A$ la variable d'action au temps $t$, $L_t$ la variable de lieu au temps $t$ et $L_{t+\Delta t}$ la variable de lieu au temps $t + \Delta t$. Cette décomposition correspond à un modèle de type localisation markovienne (voir Figure 2.12). Nous voyons donc apparaître les termes $P(P \mid L_t)$ et $P(L_{t+\Delta t} \mid A\ L_t)$ qui sont respectivement appelés le modèle capteur et le terme de prédiction. Le premier donne une distribution de probabilité sur ce qui doit être perçu, sachant le lieu où l'on se trouve. Le second donne une distribution de probabilité sur le lieu dans lequel on se trouvera au pas de temps suivant, sachant le lieu où l'on se trouve et l'action accomplie.

Passons à présent à l'assignation de formes paramétriques aux différents termes de la décomposition choisie. Aux termes $P(L_t)$ et $P(A)$ sont associées des distributions uniformes, car nous n'avons pas d'a priori sur le lieu où se trouve le robot au temps $t$, ni sur l'action à appliquer :

$$P(L_t) = \mathbf{U}_{k_{L_t}}(L_t) \text{ et } P(A) = \mathbf{U}_{k_A}(A).$$

Le modèle capteur $P(P \mid L_t)$ et le terme de prédiction $P(L_{t+\Delta t} \mid A\ L_t)$ ont chacun pour forme paramétrique une famille de gaussiennes :

$$\begin{aligned} P(P \mid L_t) &= \mathbf{G}_{\mu(L_t),\ \sigma(L_t)}(P), \\ P(L_{t+\Delta t} \mid A\ L_t) &= \mathbf{G}_{\mu(A,L_t),\ \sigma(A,L_t)}(L_{t+\Delta t}), \end{aligned}$$

où $\mu(L_t)$ et $\sigma(L_t)$ sont respectivement une moyenne et un écart-type déterminés par la valeur de $L_t$, et où $\mu(A, L_t)$ et $\sigma(A, L_t)$ sont respectivement la moyenne et l'écart-type déterminés par les valeurs de $L_t$ et $A$. La façon dont sont obtenus les fonctions $\mu(L_t)$ et $\sigma(L_t)$ pour le modèle capteur sera détaillée Section 4.2. Pour que la carte bayésienne soit complète, il ne manque alors plus que les fonctions $\mu(A, L_t)$ et $\sigma(A, L_t)$.

Ces fonctions peuvent être obtenues à partir des observations faites par le robot au cours de l'application du comportement initial. En effet, on enregistre, à chaque pas de temps, le lieu où il se trouve et l'action qu'il accomplit. L'historique de ces observations nous permet donc de calculer, pour chaque couple $\langle A, L_t \rangle$, quel est en moyenne le lieu dans

lequel il se retrouve au pas de temps suivant, et avec quel écart-type. On peut également les calculer pour le lieu dans lequel il se retrouve deux pas de temps après. Dans ce cas, on obtient $L_{t+\Delta t}$ avec $\Delta t$ égal à deux pas de temps. On peut ainsi calculer moyennes et écart-types pour différents $\Delta t$, à condition qu'ils soient multiples du pas de temps pour lequel ont été enregistrées les observations.

## 3.3 Exploitation de la carte pour accomplir de nouveaux comportements

La carte bayésienne obtenue nous permet de résoudre des tâches de navigation. Ces tâches consistent, pour le robot, à essayer d'atteindre un but choisi par le programmeur. Pour résoudre une telle tâche de navigation, avec la carte bayésienne, on pose la question suivante :

$$P(A \mid [L_t = l_1]\ [L_{t+\Delta t} = l_2]),$$

c'est-à-dire, sachant le lieu $L_t = l_1$ [1] où se trouve le robot et le lieu $L_{t+\Delta t} = l_2$ qu'il doit atteindre, quelle est l'action qu'il doit accomplir ? Donc $l_2$ est le but choisi et en demandant, à chaque pas de temps, quelle action doit accomplir le robot pour s'en approcher, celui-ci finira par l'atteindre. Dans le cadre de la carte bayésienne, la question $P(A \mid [L_t = l_1]\ [L_{t+\Delta t} = l_2])$ est appelée comportement.

L'inférence bayésienne permet de répondre à la question posée à partir de la décomposition choisie :

$$\begin{aligned}
&P(A \mid [L_t = l_1]\ [L_{t+\Delta t} = l_2]) \\
&\overset{[R1]}{=} \frac{P(A\ [L_t = l_1]\ [L_{t+\Delta t} = l_2])}{P([L_t = l_1]\ [L_{t+\Delta t} = l_2])} && (3.1)\\
&= \frac{1}{Z_1}\ P(A\ [L_t = l_1]\ [L_{t+\Delta t} = l_2]) && (3.2)\\
&\overset{[R1]}{=} \frac{1}{Z_1}\ P(A)\ P([L_t = l_1])\ P([L_{t+\Delta t} = l_2] \mid A\ [L_t = l_1]) && (3.3)\\
&= \frac{1}{Z_2}\ P([L_{t+\Delta t} = l_2] \mid A\ [L_t = l_1]). && (3.4)
\end{aligned}$$

Dans la dérivation ci-dessus, les égalités (3.1) et (3.3) sont obtenues par application de la règle [R1] (voir Section 2.3.1). Le passage de (3.1) à (3.2) est justifié par le fait que le dénominateur, ayant une valeur fixe, peut être noté en une constante de normalisation $1/Z_1$. Le passage de (3.3) à (3.4) est obtenu par l'inclusion des termes $P(A)$ et $P([L_t = l_1])$ dans la constante de normalisation : en effet, $P([L_t = l_1])$ est une valeur constante, et la distribution de probabilité $P(A)$ est définie comme étant uniforme, donc sa valeur ne varie pas.

[1] En pratique, ce lieu n'est pas connu directement, mais est calculé à chaque pas de temps, grâce au modèle capteur (voir Section 4.2.3).

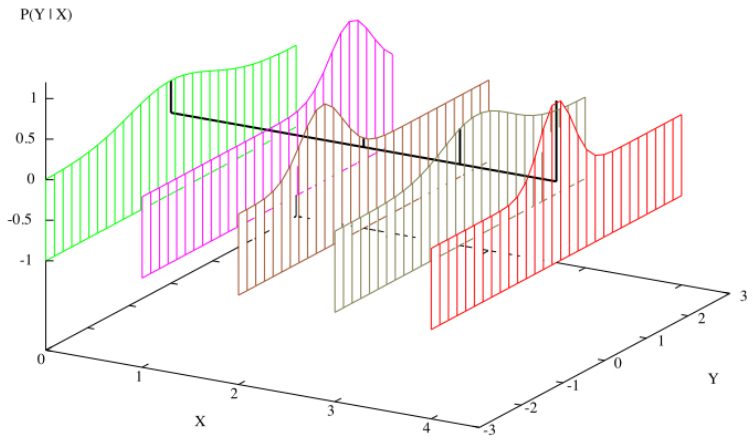



FIG. 3.1 – Exemple d'une inversion sur un ensemble de cinq gaussiennes. Ces gaussiennes sont à une dimension, pour une variable $Y$, dépendant d'une variable $X$ : $P(Y \mid X) = \mathbf{G}_{\mu(X),\ \sigma(X)}(Y)$. Cette figure illustre le mécanisme d'inversion : la distribution de probabilité $P(X \mid [Y = 0])$ est obtenue par « coupe » dans les gaussiennes. En effet, les traits noirs montrent les valeurs de $P([Y = 0] \mid X)$ pour chaque gaussienne. On peut d'ailleurs remarquer que, dans cette distribution, la valeur de $X$ la plus probable est 4 (voir la dernière gaussienne).

La distribution de probabilité correspondant à la question est donc calculée par « inversion » du terme de prédiction. Ce terme est la distribution de probabilité sur $L_{t+\Delta t}$ sachant $A$ et $L_t$. Or ici on connaît les valeurs des variables $L_t$ et $L_{t+\Delta t}$ et on veut obtenir une distribution de probabilité sur les valeurs de la variable $A$. Le terme de prédiction a pour forme paramétrique un ensemble de gaussiennes. À chaque valeur $a_i$ possible de $A$ correspond la gaussienne $\mathbf{G}_{\mu(a_i,l_1),\sigma(a_i,l_1)}$. La valeur de cette gaussienne au point $L_{t+\Delta t} = l_2$ donne la probabilité non normalisée de $A = a_i$. Une fois obtenues les probabilités pour toutes les valeurs de $A$ possibles, elles peuvent être normalisées. Un exemple d'inversion de gaussienne à une dimension est illustré dans la Figure 3.1.

Dans la distribution de probabilité sur $A$ ainsi crée, on peut alors tirer aléatoirement une action à accomplir pour se rapprocher du but. Cette action se traduit en ordres moteurs que le robot exécute pour se déplacer dans l'environnement. En calculant, comme nous venons de le décrire, une distribution de probabilité sur $A$ puis en tirant aléatoirement une action à accomplir à chaque pas de temps, le robot applique ainsi un comportement.

# Chapitre 4

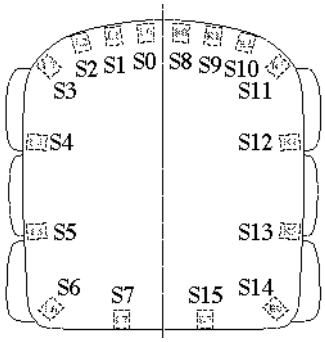
# Réalisation et résultats

## 4.1 Plate-forme expérimentale

### 4.1.1 Robot

Les différentes expériences présentées ont été réalisées avec un robot mobile développé à l'École Polytechnique Fédérale de Lausanne (EPFL) : le Koala (voir Figure 4.1(a)) [1].

(a) Le robot mobile Koala. Les points verts et rouges sur le haut du robot ne sont pas utilisés dans le cadre de notre expérience.




(b) L'emplacement des capteurs sur le Koala.

FIG. 4.1 – Photo et schéma du robot Koala.

Le Koala (longueur 32 cm, largeur 32 cm, hauteur 20 cm, poids 3 kg dans sa configuration de base) dispose d'un processeur embarqué performant (M68331, 32 bits, cadencé a 16 MHz), associé à une EEPROM de 128 Koctets et une RAM statique de 1 Moctets. Le BIOS est le système bas niveau embarqué dans le robot. Il offre des capacités multi-tâches

[1] Ce descriptif du robot Koala est repris de la thèse de J. Diard [6].

et permet la gestion de plusieurs modules logiciels : acquisition et conversion des données sensorielles, asservissement des moteurs, contrôle de la communication entre différents modules du robot et l'extérieur.

La partie motrice est constituée de deux blocs de trois roues latérales commandés indépendamment. Ils offrent en particulier au Koala la possibilité de tourner sur place. Le Koala est donc quasiment holonome. Chaque bloc de roues est entraîné par un moteur continu ($M_g$ et $M_d$), associé à un réducteur à vis sans fin et à un codeur incrémental. Ces capteurs ($PosG$ et $PosD$) fournissent une information sur l'ego-mouvement du robot. À chaque moteur sont associés deux asservissements classiques de type PID permettant le contrôle du robot en vitesse ou en position.

Les ressources systèmes fournissent des ordres permettant de spécifier les valeurs des vitesses de rotation des moteurs gauche et droit ($V_g$ et $V_d$). Nous avons préféré travailler avec des variables homogènes aux vitesses de rotation ($Vr$) et de translation ($Vt$) du robot. Ces 4 variables sont reliées les unes aux autres par les relations :

$$\begin{aligned} V_g &= Vt + Vr, \\ V_d &= Vt - Vr. \end{aligned}$$

Le Koala dispose d'un anneau de 16 capteurs infrarouges. Chaque capteur est composé d'un émetteur de lumière infrarouge et d'un récepteur. Ces capteurs permettent deux types de mesure : une de la luminosité ambiante en mode récepteur (mode lumimètre, variables $Lm_i$), et l'autre de la réflexion de la lumière par les obstacles en utilisant le mode émetteur et récepteur (mode proximètre, variables $Px_i$). Cette dernière mesure permet d'obtenir une information relative à la distance des obstacles pouvant varier entre 1 et 25 cm, dépendant fortement de leur nature (orientation, couleur, matière, etc.) et des conditions de luminosité ambiante. Le positionnement des capteurs sur le Koala est montré Figure 4.1(b). Cependant, dans le cadre de ce projet, seul le mode proximètre est utilisé.

### 4.1.2 Environnement

Toutes nos expériences ont été menées dans la halle robotique de l'INRIA. Le robot n'a donc roulé qu'en intérieur, sur un sol lisse et sous éclairage artificiel. Cet éclairage est d'ailleurs nécessaire au bon fonctionnement des proximètres qui sont trop sensibles à la lumière du soleil.

Les environnements utilisés ont été construits à l'aide de planches blanches, dressées pour en faire des murs, d'environ 30 cm de hauteur et à la longueur variant de 50 cm à 1 m. Un ballon de volley à également été utilisé pour certaines expériences.

### 4.1.3 Logiciels

La programmation du robot s'est faite en C++, avec l'aide de deux librairies. La première est une librairie de communication avec le robot. Elle permet d'ouvrir ou de fermer

la communication, d'envoyer des ordres aux moteurs et de récupérer les valeurs des proximètres. La seconde, nommée ProBT $^{(R)}$, est une librairie permettant de faire de l'inférence et du calcul bayésien.

## 4.2 Mise en œuvre du projet

### 4.2.1 Les variables

Les quatre variables utilisées dans une carte bayésienne sont une variable de perception, une variable d'action et deux variables de lieux, l'une au temps $t$ et l'autre au temps $t+\Delta t$. Dans cette section, nous montrons à quoi correspondent ces variables dans le cadre de notre expérience robotique.

La variable de perception est un ensemble de 16 variables $Px_0, \ldots, Px_{15}$, qui font référence aux 16 proximètres du robot. Chacune de ces variables peut prendre une valeur entière entre 0 et 1023, puisque les proximètres du robot renvoient chacun une valeur entre 0 et 1023. Cette valeur est d'autant plus élevée que l'obstacle est proche.

La variable d'action représente les différentes vitesses de rotation que peut appliquer le robot. En effet, dans nos expériences, la vitesse de translation est fixe et les décisions à prendre pour l'action ne portent que sur la vitesse de rotation à appliquer. La variable d'action est donc la variable $Vrot$, introduite Section 4.1.1, et peut prendre trois valeurs. La valeur -1, qui fait appliquer au robot une rotation vers la gauche ($Vr = -25$), la valeur 0 pour une vitesse de rotation nulle ($Vr = 0$) et la valeur 25, pour une rotation vers la droite ($Vr = 25$). Techniquement, la vitesse de rotation $Vr$ applicable par le robot peut prendre n'importe quelle valeur entière entre -50 (vitesse de rotation vers la gauche la plus grande) et 50 (vitesse de rotation vers la droite la plus grande). Parmi ces 101 valeurs possibles, nous avons donc choisi de restreindre le domaine de $Vrot$ à 3 valeurs.

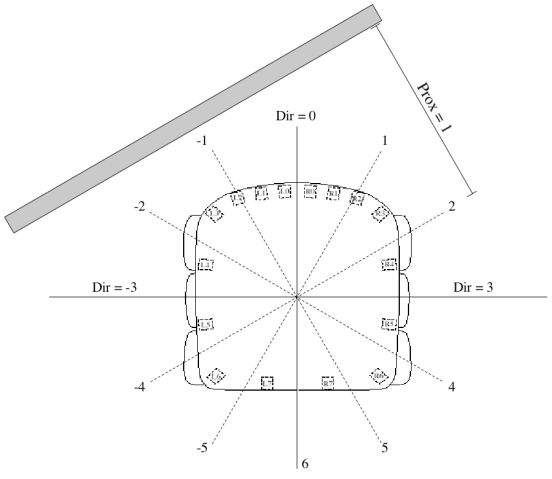



FIG. 4.2 – Répartition des différentes valeurs pour $Dir$. Sur ce schéma, $Dir = -1$ et $Prox = 1$.

Les deux variables de lieux ont le même domaine. Dans la représentation interne choisie, on fait l'hypothèse que le robot ne voit jamais qu'un seul mur à la fois. Un lieu est ici défini par la proximité et l'angle du robot à ce mur. Ainsi, les variables de lieu sont un couple de deux variables : $Dir$, pour la direction de l'obstacle et $Prox$ pour sa proximité. La variable $Dir$ peut prendre 12 valeurs entières, de -5 à 6. La Figure 4.2 montre la répartition des valeurs que peut prendre cette variable. Pour $Prox$ ont été choisies 3 valeurs que nous avons définies ainsi :

- $Prox = 0$ quand le centre du robot est situé à 44 cm du mur ;
- $Prox = 1$ quand le centre du robot est situé à 35 cm du mur ;
- $Prox = 2$ quand le centre du robot est situé à 24 cm du mur.

La façon dont sont déterminées les valeurs de $Dir$ et $Prox$ à partir des données des proximètres est décrite dans la Section 4.2.3.

**Variables Pertinentes**

$$
\begin{array}{lcl}
P & = & Px_0 \wedge \ldots \wedge Px_{15} \; ; \quad \mathcal{D}_P = \{0, \ldots, 1023\}^{16} \\
L_t & = & Dir_t \wedge Prox_t \; ; \quad \mathcal{D}_{L_t} = \{-5, \ldots, 6\} \times \{0, 1, 2\} \\
L_{t+\Delta t} & = & Dir_{t+\Delta t} \wedge Prox_{t+\Delta t} \; ; \quad \mathcal{D}_{L_{t+\Delta t}} = \{-5, \ldots, 6\} \times \{0, 1, 2\} \\
A & = & Vrot \; ; \quad \mathcal{D}_A = \{-1, 0, 1\}
\end{array}
$$

**Fig. 4.3:** Description des variables pertinentes. La notation $\mathcal{D}$ est utilisée pour définir le domaine de valeurs d'une variable.

### 4.2.2 Décomposition et formes paramétriques

Avec la définition des variables données ci-dessus, la décomposition de la distribution de probabilité conjointe prend la forme suivante :

$$
\begin{aligned}
& P(Dir_t \; Prox_t \; Vrot \; Px_0 \ldots Px_{15} \; Dir_{t+\Delta t} \; Prox_{t+\Delta t}) \\
& \quad = \; P(Dir_t \; Prox_t) P(Vrot) P(Px_0 \ldots Px_{15} \mid Dir_t \; Prox_t) \\
& \qquad P(Dir_{t+\Delta t} \; Prox_{t+\Delta t} \mid Dir_t \; Prox_t \; Vrot) \qquad (4.1) \\
& \quad = \; P(Dir_t \; Prox_t) P(Vrot) \prod_i P(Px_i \mid Dir_t \; Prox_t) \\
& \qquad P(Dir_{t+\Delta t} \; Prox_{t+\Delta t} \mid Dir_t \; Prox_t \; Vrot) \qquad (4.2)
\end{aligned}
$$

L'égalité (4.2) est obtenue par une hypothèse d'indépendance conditionnelle : on fait l'hypothèse que la valeur d'un capteur est indépendante de la valeur des autres capteurs, à condition de connaître le lieu où se trouve le robot.

Les formes paramétriques associées aux différents termes sont exposées Figure 4.4.

### 4.2.3 Le modèle capteur

Aux termes $\prod_i P(Px_i \mid Dir_t \; Prox_t)$, qui correspondent au modèle capteur, sont associées des distributions de probabilité gaussiennes (voir Figure 4.4). Ces gaussiennes ont été

**Formes Paramétriques**

$$
\begin{aligned}
P(Dir_t\ Prox_t) &= \mathbf{U}(Dir_t, Prox_t)\\
P(Vrot) &= \mathbf{U}(Vrot)\\
P(Px_i \mid Dir_t\ Prox_t) &= \mathbf{G}_{\mu(Dir_t,Prox_t),\ \sigma(Dir_t,Prox_t)}(Px_i)\\
P(Dir_{t+\Delta t}\ Prox_{t+\Delta t} &\mid Dir_t\ Prox_t\ Vrot)\\
&= \mathbf{G}_{\mu(Dir_t,Prox_t,Vrot),\ \sigma(Dir_t,Prox_t,Vrot)}(Dir_{t+\Delta t}, Prox_{t+\Delta t})
\end{aligned}
$$

**FIG. 4.4:** Formes paramétriques associées aux différents termes apparaissant dans la décomposition.

identifiées expérimentalement. En pratique, il s'agit de poser le robot dans l'environnement à un angle et une distance donnée, d'indiquer ces valeurs au robot et de lui faire enregistrer un ensemble de données capteurs. Dans notre expérience, nous enregistrons 50 données par capteur. Les triplets enregistrés sont de la forme $\langle px_i, dir_t, prox_t\rangle$ pour chaque capteur, ce qui permet le calcul des moyennes et écart-types des gaussiennes.

Le modèle capteur nous permet de calculer, à chaque pas de temps, quel est le lieu $\langle dir_t, prox_t\rangle$ dans lequel se trouve le robot. Pour le déterminer, on pose donc la question :

$$
\begin{aligned}
&P(Dir_t\ Prox_t \mid [Px_0 = px_0] \ldots [Px_{15} = px_{15}])\\
&= \frac{1}{Z_2}\ P(Px_0 = px_0] \ldots [Px_{15} = px_{15}] \mid Dir_t\ Prox_t).
\end{aligned}
$$

La distribution de probabilité correspondant à la question se calcule donc par « inversion » du terme du modèle capteur, selon la même méthode que celle décrite Section 3.3.

### 4.2.4 Le comportement initial et l'apprentissage de la carte bayésienne

Initialement, le comportement connu par le robot est un comportement « aléatoire ». Appliquer ce comportement consiste à tirer selon une distribution uniforme une valeur de $Vrot$ et à maintenir l'ordre moteur correspondant pendant une seconde, avant de faire un nouveau tirage. L'ordre moteur pourrait être changé toutes les 100 ms, mais cela ne laisserait pas suffisamment de temps au robot pour observer les variations de $Dir$ et de $Prox$ engendrées par ses actions. En revanche, en une seconde, les valeurs de ces variables peuvent changer plusieurs fois au cours de l'application d'un ordre moteur.

Pour l'apprentissage de la carte bayésienne, ce comportement initial a été appliqué avec une vitesse de translation nulle. L'apprentissage a donc été fait en trois phases, afin d'avoir des données pour les trois valeurs possibles de $Prox$. Une première phase d'apprentissage a consisté à faire appliquer son comportement au robot à 44 cm d'un mur, soit à $Prox = 0$ (voir Figure 4.5 [2]). Pour la seconde phase, le robot a appliqué son comportement à 35 cm

[2]Ces images sont tirées d'un film disponible à l'adresse `http://julien.diard.free.fr/research.html`

du mur ($Prox = 1$) et pour la troisième à 24 cm du mur ($Prox = 2$). Chaque phase a duré cinq minutes.

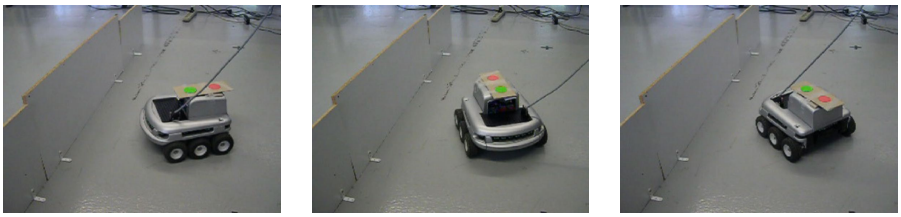


FIG. 4.5: Le robot appliquant le comportement initial pour une phase d'apprentissage à $Prox = 0$.

Pendant ces cinq minutes, toutes les 100 ms, les valeurs de $Dir$ et de $Prox$, ainsi que la valeur de $Vrot$ courante, ont été enregistrées. Cela donne une liste de 3000 triplets. Dans cet historique, on peut donc savoir, pour un lieu donné et une action appliquée, quel est le lieu auquel est arrivé le robot au pas de temps suivant. En effet, $Dir_t$, $Prox_t$ et $Vrot$ sont alors donnés par un triplet, tandis que $Dir_{t+\Delta t}$ et $Prox_{t+\Delta t}$ sont donnés par le triplet suivant dans l'historique. Ainsi, on peut obtenir un ensemble $L_t$, $A$ et $L_{t+\Delta t}$.

Mais ce $\Delta t$ n'a pas à correspondre forcément à 100 ms. L'historique nous permet en effet de tirer des ensembles de $L_t$, $A$ et $L_{t+\Delta t}$ pour $\Delta t = 200\ ms$, $\Delta t = 300\ ms$, $\Delta t = 400\ ms$, etc. Par exemple, pour $\Delta t = 200\ ms$, il faut prendre, dans l'historique, pour $L_{t+\Delta t}$, les valeur de $Dir$ et $Prox$ du deuxième triplet suivant celui donnant $L_t$ et $A$. Il s'est effectivement passé 200 ms entre l'observation de ce triplet et celle du triplet d'où l'on tire la valeur de $L_{t+\Delta t}$.

On peut ainsi générer, à partir des historiques donnés par les trois phases d'apprentissage, des ensembles de valeurs pour $L_t$, $A$ et $L_{t+\Delta t}$, avec différents $\Delta t$, à condition que la même action ait bien été appliquée durant ce $\Delta t$. Comme les observations ont été enregistrées avec un ordre moteur variant possiblement toutes les secondes, on a au plus $\Delta t = 1s$.

Un $\Delta t$ étant choisi, on peut recenser les valeurs de $L_{t+\Delta t}$ trouvées pour un couple $L_t$, $A$ donné. Pour chacun de ces couples, on peut donc calculer une moyenne et un écart-type sur la répartition des $L_{t+\Delta t}$. Cette variable étant constituée des deux variables $Dir$ et $Prox$, on obtient une gaussienne à deux dimensions pour chaque couple $L_t$ et $A$. C'est ainsi qu'est apprise la forme paramétrique associée au terme de prédiction :

$$\begin{aligned} & P(Dir_{t+\Delta t}\ Prox_{t+\Delta t} \mid Dir_t\ Prox_t\ Vrot) \\ & \quad = \quad \mathbf{G}_{\mu(Dir_t, Prox_t, Vrot),\ \sigma(Dir_t, Prox_t, Vrot)}(Dir_{t+\Delta t}, Prox_{t+\Delta t}), \end{aligned}$$

$\mu(Dir_t, Prox_t, Vrot)$ et $\sigma(Dir_t, Prox_t, Vrot)$, la moyenne et l'écart-type selon $Dir_t, Prox_t$ et $Vrot$, venant d'être calculés à partir des trois historiques d'observations.

### 4.2.5 Question et résumé

Au vu des variables utilisées, les questions que nous poserons à la carte bayésienne seront de la forme :

$$\begin{aligned} & P(Vrot \mid [Dir_t = d_1]\ [Prox_t = p_1]\ [Dir_{t+\Delta t} = d_2]\ [Prox_{t+\Delta t} = p_2]) \\ = \ & \frac{1}{Z_2}\ P([Dir_{t+\Delta t} = d_2]\ [Prox_{t+\Delta t} = p_2] \mid Vrot\ [Dir_t = d_1]\ [Prox_t = p_1]). \end{aligned}$$

Ce que nous avons obtenu en exploitant ce type de question est décrit dans la Section 4.3. La Figure 4.6 résume les informations que nous avons données ci-dessus sur la carte bayésienne utilisée dans notre expérience.

Programme

- Description
  - Spécification
    - Variables Pertinentes
      - $P = Px_0 \wedge \ldots \wedge Px_{15}$ ; $\mathcal{D}_P = \{0, \ldots, 1023\}^{16}$
      - $L_t = Dir_t \wedge Prox_t$ ; $\mathcal{D}_{L_t} = \{-5, \ldots, 6\} \times \{0, 1, 2\}$
      - $L_{t+\Delta t} = Dir_{t+\Delta t} \wedge Prox_{t+\Delta t}$ ; $\mathcal{D}_{L_{t+\Delta t}} = \{-5, \ldots, 6\} \times \{0, 1, 2\}$
      - $A = Vrot$ ; $\mathcal{D}_A = \{-1, 0, 1\}$
    - Décomposition
      - $P(Dir_t\ Prox_t\ Vrot\ Px_0 \ldots Px_{15}\ Dir_{t+\Delta t}\ Prox_{t+\Delta t})$
      - $= P(Dir_t\ Prox_t) P(Vrot) \prod_i P(Px_i \mid Dir_t\ Prox_t)$
      - $P(Dir_{t+\Delta t}\ Prox_{t+\Delta t} \mid Dir_t\ Prox_t\ Vrot)$
    - Formes Paramétriques
      - $P(Dir_t\ Prox_t) = \mathbf{U}(Dir_t, Prox_t)$
      - $P(Vrot) = \mathbf{U}(Vrot)$
      - $P(Px_i \mid Dir_t\ Prox_t) = \mathbf{G}_{\mu(Dir_t, Prox_t),\ \sigma(Dir_t, Prox_t)}(Px_i)$
      - $P(Dir_{t+\Delta t}\ Prox_{t+\Delta t} \mid Dir_t\ Prox_t\ Vrot) = \mathbf{G}_{\substack{\mu(Dir_t, Prox_t, Vrot), \\ \sigma(Dir_t, Prox_t, Vrot)}}(Dir_{t+\Delta t}, Prox_{t+\Delta t})$
  - Identification :
    - Apprentissage de $P(Px_i \mid Dir_t\ Prox_t)$
    - et de $P(Dir_{t+\Delta t}\ Prox_{t+\Delta t} \mid Dir_t\ Prox_t\ Vrot)$
- Question : définition de *comportements*
  - $P(Vrot \mid [Dir_t = d_1]\ [Prox_t = p_1]\ [Dir_{t+\Delta t} = d_2]\ [Prox_{t+\Delta t} = p_2])$
  - $= \frac{1}{Z_2}\ P([Dir_{t+\Delta t} = d_2]\ [Prox_{t+\Delta t} = p_2] \mid Vrot\ [Dir_t = d_1]\ [Prox_t = p_1])$

FIG. 4.6: Résumé de notre carte bayésienne.

## 4.3 Résultats

La carte bayésienne étant apprise, nous l'avons exploitée avec diverses questions, en vue de générer divers comportements. Les résultats que nous présentons dans cette section ont tous été obtenus avec une carte apprise pour $\Delta t = 900$ ms (voir Section 4.2.4).

### 4.3.1 Comportements à vitesse de translation nulle

Dans un premier temps, nous avons gardé nulle la vitesse de translation. Le robot étant situé à proximité d'un mur, nous lui avons spécifié comme but une valeur de $Dir$ à atteindre, la valeur de $Prox$ devant rester telle quelle. Nous lui avons demandé ainsi de s'aligner sur différentes valeur de $Dir$, afin de tester ses limites. La plupart du temps, le robot atteint effectivement le but fixé et s'y maintient. Les cas qui lui posent un problème et donc pour lesquels il ne peut atteindre le but seront explicités dans la Section 4.4. Néanmoins, nous avons considérés satisfaisants ces premiers comportements obtenus, du type « s'orienter dans une direction, par rapport à un obstacle ».

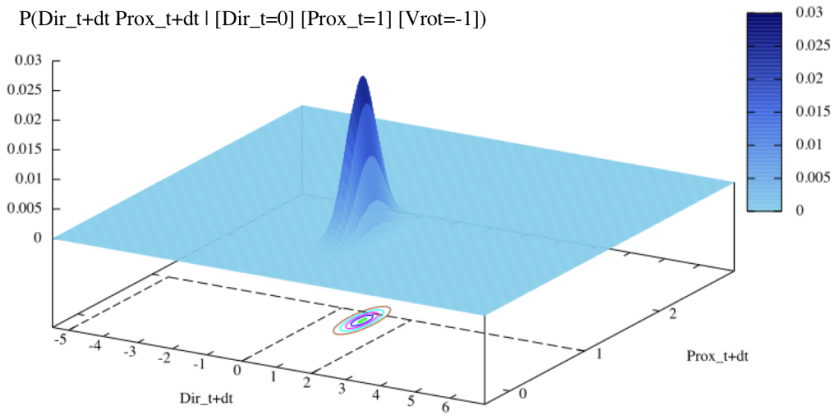
Les Figures 4.7 et 4.8 illustrent la façon dont une distribution sur $Vrot$ est calculée, pour répondre à une question donnée. Les sous-figures 4.7(a), 4.7(b) et 4.7(c) montrent respectivement les gaussiennes correspondant à $P(Dir_{t+\Delta t}\ Prox_{t+\Delta t} \mid [Dir_t = 0]\ [Prox_t = 1]\ [Vrot = -1])$, à $P(Dir_{t+\Delta t}\ Prox_{t+\Delta t} \mid [Dir_t = 0]\ [Prox_t = 1]\ [Vrot = 0])$ et à $P(Dir_{t+\Delta t}\ Prox_{t+\Delta t} \mid [Dir_t = 0]\ [Prox_t = 1]\ [Vrot = 1])$. Sur ces figures, sont représentées, en pointillés, les coupes que l'ont fait dans ces gaussiennes pour répondre à trois questions différentes. Par exemple, le point de coupe le plus à gauche correspond à la question où le but est $Dir_{t+\Delta t} = -5$ et $Prox_{t+\Delta t} = 1$. La résolution de cette question suit un principe « d'inversion » similaire à celui décrit précédemment dans le cas des gaussiennes à une dimension (voir Section 3.3). Les sous-figures 4.8(a), 4.8(b) et 4.8(c) montrent les distributions sur $Vrot$ obtenues en réponse à ces questions :

$$P(Vrot \mid [Dir_t = 0]\ [Prox_t = 1]\ [Dir_{t+\Delta t} = -5]\ [Prox_{t+\Delta t} = 1]),$$
$$P(Vrot \mid [Dir_t = 0]\ [Prox_t = 1]\ [Dir_{t+\Delta t} = 0]\ [Prox_{t+\Delta t} = 1]),$$
$$P(Vrot \mid [Dir_t = 0]\ [Prox_t = 1]\ [Dir_{t+\Delta t} = 2]\ [Prox_{t+\Delta t} = 1]).$$

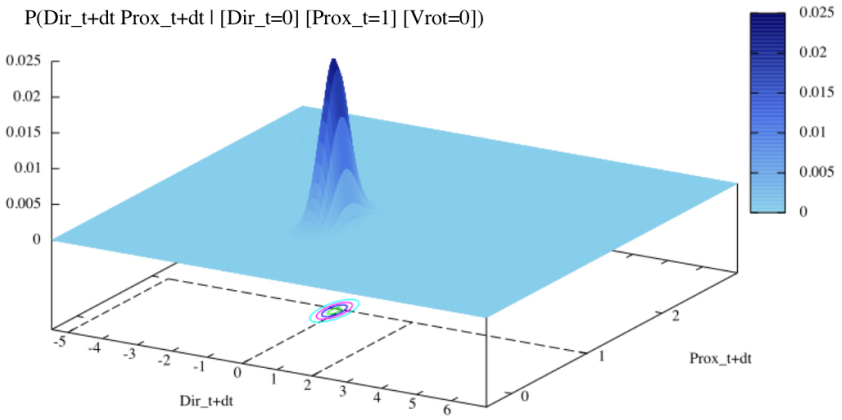
Nous venons d'illustrer le mécanisme de l'inférence nécessaire à la génération d'un comportement, dans le cas où la vitesse de translation est nulle. Ce mécanisme est similaire pour les comportements suivants et ne sera donc pas redétaillé.

### 4.3.2 Comportements en présence d'un mur

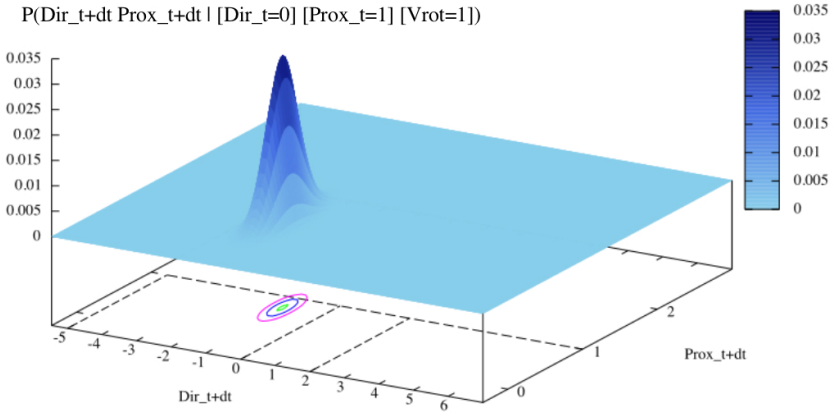
Les comportements suivants (ceux de cette section et de la suivante) ont tous été réalisés avec une vitesse de translation égale à 20. Dans chacun des cas présentés ci-dessous, le but fixé permet d'obtenir du robot un comportement reconnaissable par un observateur extérieur. Ceci nous permet de valider la réussite du robot à appliquer ces nouveaux comportements.




(a) Gaussienne pour $Dir_t = 0$, $Prox_t = 1$ et $Vrot = -1$.

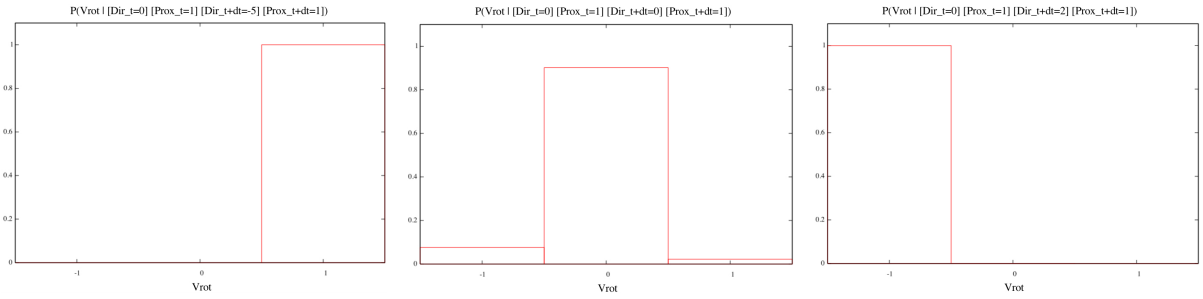



(b) Gaussienne pour $Dir_t = 0$, $Prox_t = 1$ et $Vrot = 0$.




(c) Gaussienne pour $Dir_t = 0$, $Prox_t = 1$ et $Vrot = 1$.

FIG. 4.7: Exemples de distributions comprises dans le terme de prédiction $P(Dir_{t+\Delta t}\ Prox_{t+\Delta t} \mid Dir_t\ Prox_t\ Vrot)$, avec $\Delta t = 900$ ms.




(a) Distribution sur $Vrot$ obtenue pour $Dir_{t+\Delta t} = -5$ et $Prox_{t+\Delta t} = 1$.

(b) Distribution sur $Vrot$ obtenue pour $Dir_{t+\Delta t} = 0$ et $Prox_{t+\Delta t} = 1$.

(c) Distribution sur $Vrot$ obtenue pour $Dir_{t+\Delta t} = 2$ et $Prox_{t+\Delta t} = 1$.

FIG. 4.8: Exemples de distributions sur $Vrot$ obtenues en réponse à une question avec $\Delta t = 900$ ms.

Tout d'abord, nous avons demandé au robot d'appliquer un comportement d'évitement d'obstacles. Pour cela, deux buts ont été fixés, en fonction de la valeur de $Dir_t$. Le premier but permet au robot d'éviter les obstacles par la droite. Il s'applique donc dans le cas où $Dir_t \leq 0$, c'est-à-dire quand l'obstacle se situe soit en face, soit sur son côté gauche. On souhaite alors que le robot tourne sur sa droite, pour amener l'obstacle sur son côté arrière-gauche (soit $Dir = -5$). Le deuxième but permet au robot d'éviter les obstacles par la gauche et s'applique dans le cas où $Dir_t \geq 1$. On souhaite alors que le robot tourne sur sa gauche, pour amener l'obstacle derrière lui (soit $Dir = 6$). Ces buts sont atteints en posant les questions suivantes :

$$P(Vrot \mid [Dir_t = d]\ [Prox_t = p]\ [Dir_{t+\Delta t} = -5]\ [Prox_{t+\Delta t} = 0]), \quad \text{avec} \quad -5 \leq d \leq 0,$$
$$P(Vrot \mid [Dir_t = d]\ [Prox_t = p]\ [Dir_{t+\Delta t} = 6]\ [Prox_{t+\Delta t} = 0]), \quad \text{avec} \quad 1 \leq d \leq 6.$$

La Figure 4.9 [3] montre le robot appliquant le comportement d'évitement d'obstacles dans une arène.

Ensuite, nous avons fait appliquer au robot des comportements de suivi de mur, soit par la droite, soit par la gauche. Pour obtenir un suivi de mur, nous avons fixé trois buts différents en fonction de la valeur de $Prox_t$. En effet, nous voulons que le robot suive le mur sans trop s'en approcher, ni trop s'en écarter. Comme il a fait son apprentissage à des distances fixes du mur, il n'a pas appris de schéma sensori-moteur lui permettant, grâce à une rotation, de passer d'une valeur de $Prox$ à une autre. Pourtant, en suivant le mur il risque de s'en écarter ou de s'en éloigner un peu par moment, en particulier lorsqu'il doit passer des coins. Pour lui permettre de se recaler à la bonne distance, nous avons donc posé trois questions différentes, selon la valeur de $Prox_t$. Ces questions, pour le suivi droit,

[3] Ces images, et celles des figures suivantes, sont tirées de films disponibles à l'adresse `http://julien.diard.free.fr/research.html`.

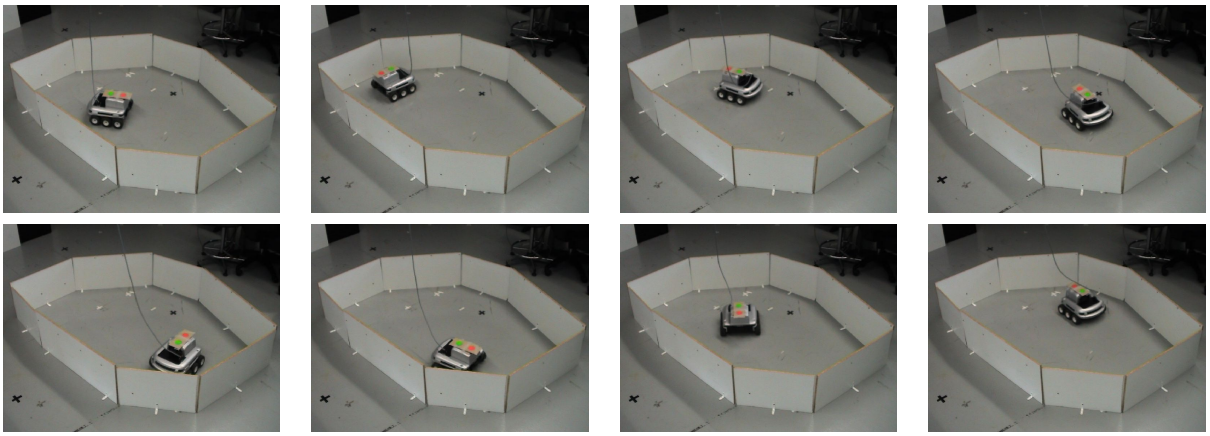


FIG. 4.9: Le robot applique un comportement d'évitement d'obstacles. Il circule donc dans l'arène en s'écartant des murs.

s'énoncent ainsi :

$$
\begin{aligned}
&P(Vrot \mid [Dir_t = d]\ [Prox_t = 0]\ [Dir_{t+\Delta t} = 2]\ [Prox_{t+\Delta t} = 1]) \quad \text{si} \quad Prox_t = 0,\\
&P(Vrot \mid [Dir_t = d]\ [Prox_t = 1]\ [Dir_{t+\Delta t} = 3]\ [Prox_{t+\Delta t} = 1]) \quad \text{si} \quad Prox_t = 1,\\
&P(Vrot \mid [Dir_t = d]\ [Prox_t = 2]\ [Dir_{t+\Delta t} = 4]\ [Prox_{t+\Delta t} = 1]) \quad \text{si} \quad Prox_t = 2.
\end{aligned}
$$

Nous avons ainsi obtenu un comportement de suivi droit de mur qui permet au robot de longer son arène aussi bien par l'intérieur que par l'extérieur (voir Figures 4.10 et 4.11).

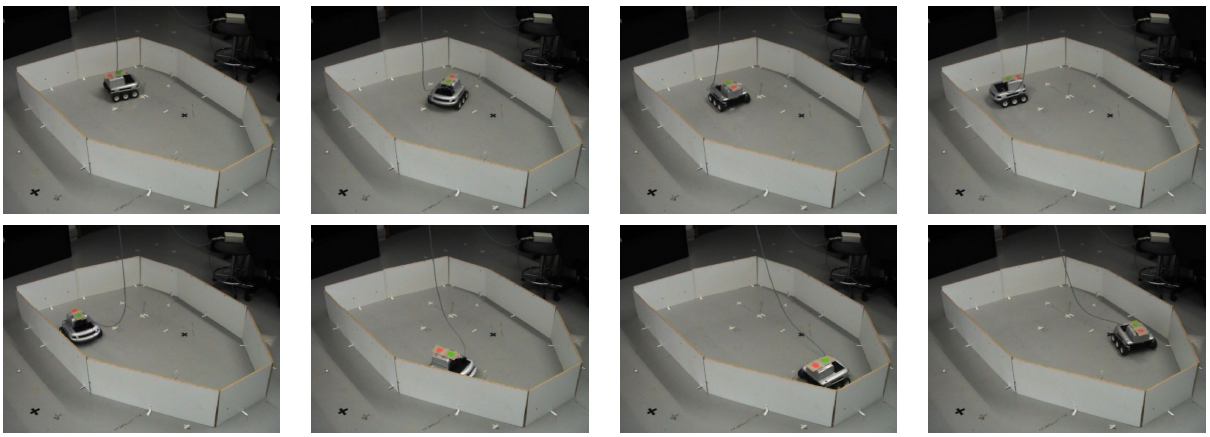


FIG. 4.10: Application du comportement de suivi droit de mur : le robot longe l'intérieur de l'arène par la droite. On peut remarquer que le robot commence son comportement avec le mur à sa gauche. Il commence donc par effectuer un demi-tour d'avoir celui-ci sur sa droite, avant de débuter le suivi.

Pour le suivi de mur gauche (voir Figures 4.12 et 4.13), les questions posées sont de la

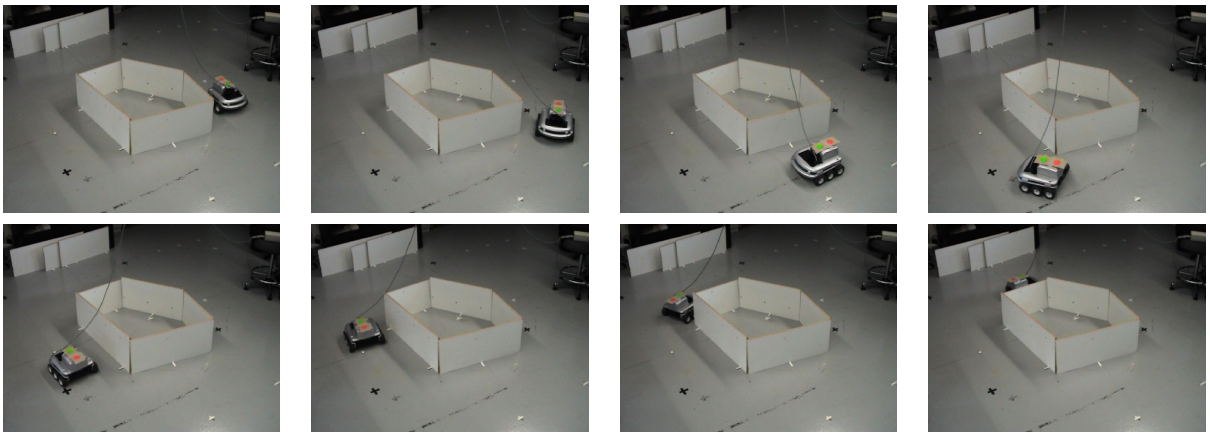


FIG. 4.11: Application du comportement de suivi droit de mur : le robot longe l'extérieur de l'arène par la droite.

même forme :

$$
\begin{aligned}
&P(Vrot \mid [Dir_t = d]\ [Prox_t = 0]\ [Dir_{t+\Delta t} = -2]\ [Prox_{t+\Delta t} = 1]) \quad \text{si} \quad Prox_t = 0,\\
&P(Vrot \mid [Dir_t = d]\ [Prox_t = 1]\ [Dir_{t+\Delta t} = -3]\ [Prox_{t+\Delta t} = 1]) \quad \text{si} \quad Prox_t = 1,\\
&P(Vrot \mid [Dir_t = d]\ [Prox_t = 2]\ [Dir_{t+\Delta t} = -4]\ [Prox_{t+\Delta t} = 1]) \quad \text{si} \quad Prox_t = 2.
\end{aligned}
$$

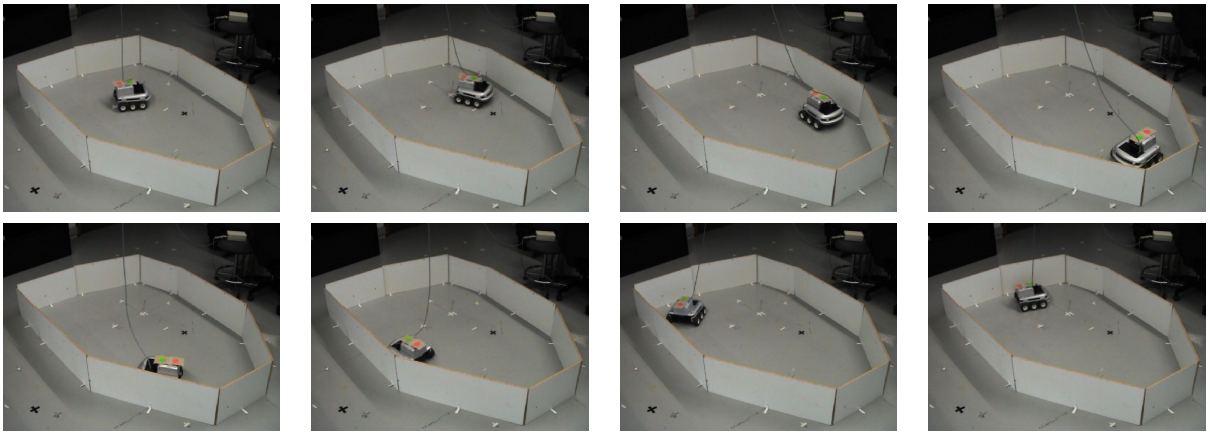


FIG. 4.12: Application du comportement de suivi gauche de mur : le robot longe l'intérieur de l'arène par la gauche.

### 4.3.3 Comportements en présence d'un ballon

Enfin, nous avons vérifié que, bien que le robot ait fait son apprentissage face à un mur constitué d'une planche blanche, il est possible de généraliser ces comportements à d'autres types d'environnement. Par exemple, nous l'avons mis en présence d'un ballon. Nous lui

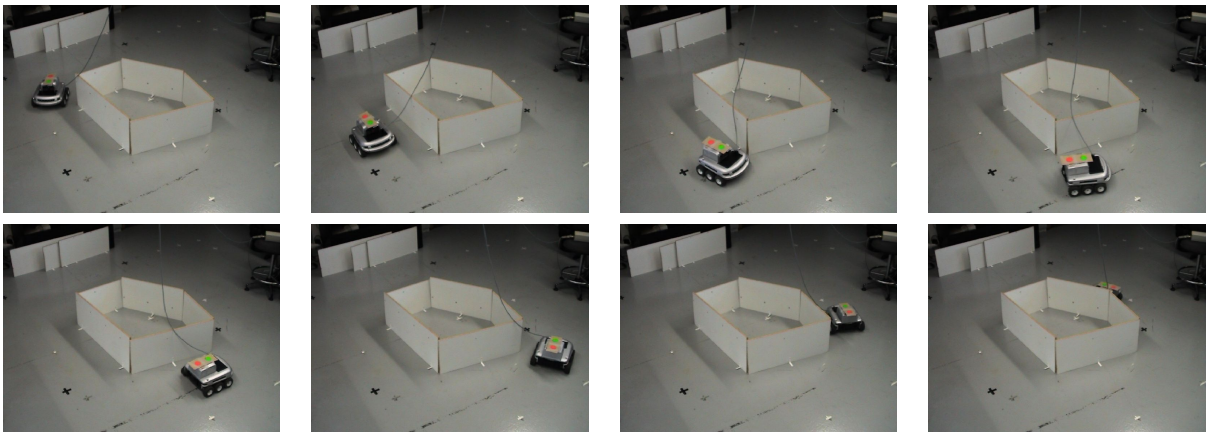


FIG. 4.13: Application du comportement de suivi gauche de mur : le robot longe l'extérieur de l'arène par la gauche.

avons donc fait appliquer un comportement de poussé de ballon, en posant la question suivante :

$$P(Vrot \mid [Dir_t = d]\ [Prox_t = p]\ [Dir_{t+\Delta t} = 0]\ [Prox_{t+\Delta t} = 2]).$$

Ainsi, le robot s'approche du ballon lorsque celui-ci est dans son entourage. Une fois en contact, il le pousse et, si on intervient pour écarter le ballon de côté, le robot tourne pour être à nouveau face à celui-ci. La Figure 4.14 illustre l'interaction d'un intervenant extérieur avec le robot, par manipulation du ballon, lorsque ce comportement est appliqué.

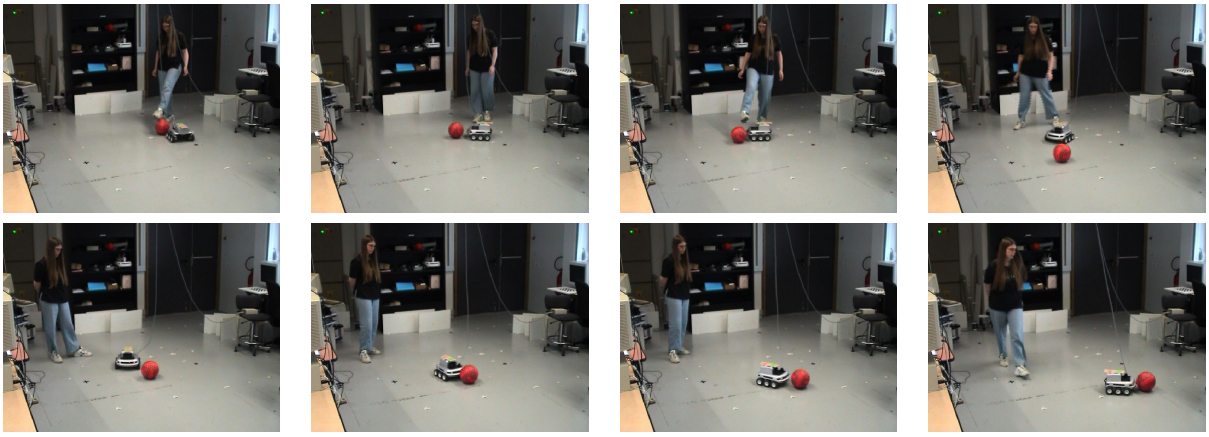


FIG. 4.14: Application interactive du comportement de poussé de ballon.

Le robot peut en fait appliquer les premiers comportements décrits aussi bien avec le ballon qu'avec les planches. Ainsi, faire appliquer le comportement d'évitement d'obstacles avec le ballon nous donne un robot qui « ne veut plus jouer » (par comparaison avec le comportement de poussé de ballon). En effet, celui-ci s'écarte désormais du ballon lorsqu'on le pose à proximité, comme le montre la Figure 4.15. Nous lui avons également fait appliquer

les comportements de suivi de mur : le robot se met à tourner autour du ballon (voir Figure 4.16)

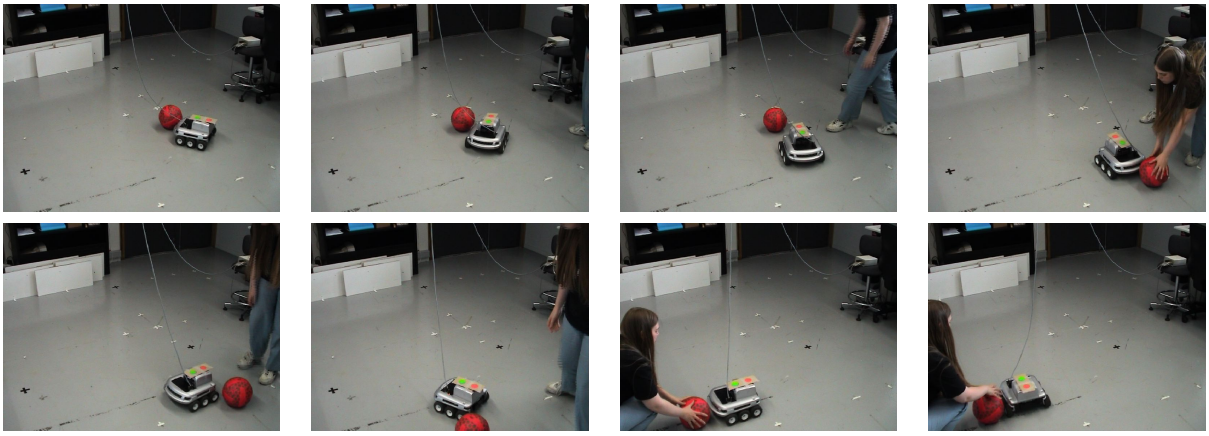


FIG. 4.15: Application du comportement d'évitement d'obstacle : le robot s'écarte du ballon.

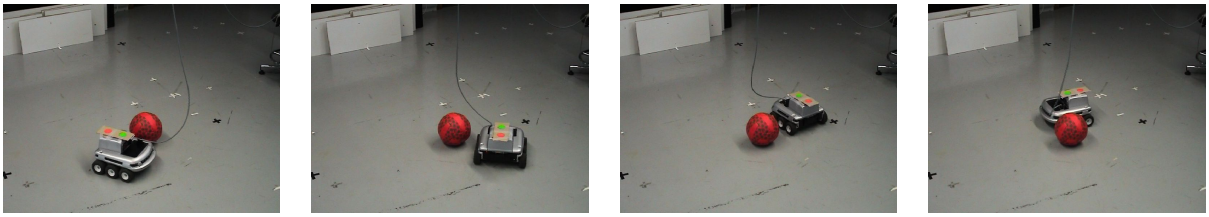


FIG. 4.16: Application du comportement de suivi de mur gauche : le robot tourne autour du ballon.

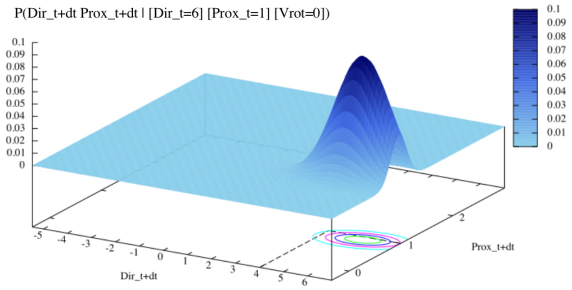
## 4.4 Analyse et discussion

Nous présentons maintenant certains problèmes techniques rencontrés pendant notre expérience robotique. Nous discutons également de leur portée, et des solutions envisagées.

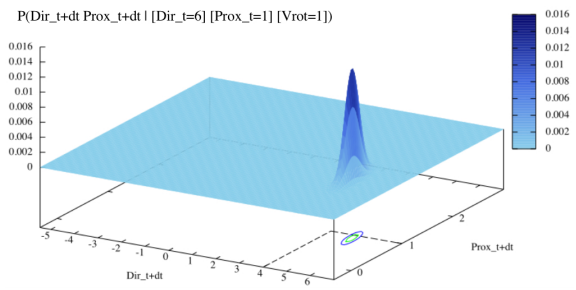
### 4.4.1 Problèmes dus à l'implantation des gaussiennes

Nous analysons ici le problème soulevé dans la Section 4.3. En effet, nous y avions remarqué que parfois le robot n'arrive pas à atteindre le but fixé, lors de nos divers essais à vitesse de translation nulle. Notre analyse des gaussiennes obtenues par apprentissage, pour le terme de prédiction, nous a permis de cerner l'origine du problème.

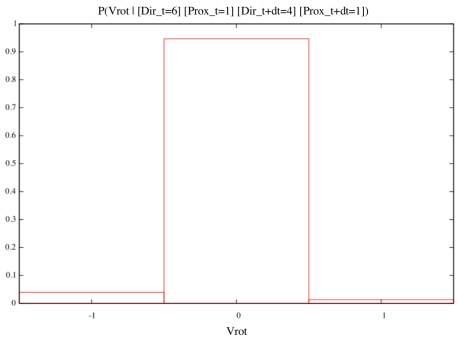
La source du problème tient à l'implantation des gaussiennes dans la librairie ProBT $^{(R)}$. En effet, les gaussiennes du terme de prédiction portent sur les variables $Dir$ et $Prox$. Or, la variable $Dir$ peut être assimilée à un angle. Ainsi, lorsque le robot est à $Dir = 6$ et qu'il




(a) Gaussienne pour $Dir_t = 6$, $Prox_t = 1$ et $Vrot = 0$.




(b) Gaussienne pour $Dir_t = 6$, $Prox_t = 1$ et $Vrot = 1$.




(c) Distribution obtenue pour $Dir_{t+\Delta t} = 4$ et $Prox_{t+\Delta t} = 1$.

FIG. 4.17: Illustration du problème que pose l'implantation des gaussiennes dans ProBT $^{(R)}$, par rapport aux caractéristiques de la variable $Dir$.

tourne à gauche, il passe à $Dir = -5$. Ce passage possible d'une borne de $Dir$ à l'autre n'est pas correctement représenté dans nos gaussiennes.

Par exemple, la gaussienne correspondant à $P(Dir_{t+\Delta t}\ Prox_{t+\Delta t} \mid [Dir_t = 6]\ [Prox_t = 1]\ [Vrot = 0])$ ne représente pas correctement ce qui se passe lorsque le mur est derrière le robot et que celui-ci reste sur place. En effet, du fait d'une certaine tendance des capteurs à fluctuer légèrement, lors de l'apprentissage, les valeurs observées pour $Dir_{t+\Delta t}$ sont principalement 6, mais parfois également 5 et -5 (alors que le robot ne bouge pas). La valeur -5 est proche d'un point de vue physique de la valeur 6, mais ce n'est pas le cas dans l'implantation. Idéalement, nous devrions donc obtenir une gaussienne centrée sur 6 avec un écart-type assez faible. Mais, de la façon dont sont implantées les gaussiennes, l'axe de la variable $Dir$ comporte la valeur -5 à une extrémité du domaine, et la valeur 6 à l'autre extrémité. Les quelques valeurs expérimentales $Dir_{t+\Delta t} = -5$ perturbent l'estimation de l'écart-type : la gaussienne correspondant à $P(Dir_{t+\Delta t}\ Prox_{t+\Delta t} \mid [Dir_t = 6]\ [Prox_t = 1]\ [Vrot = 0])$ a un très grand écart-type selon $Dir_{t+\Delta t}$, comme le montre la Figure 4.17(a). Elle ne représente donc pas correctement l'information sur la partie du schéma sensori-moteur qu'elle

concerne.

Or, cette gaussienne peut être nécessaire au calcul d'une distribution de probabilité sur $Vrot$, pour répondre à une question. Par exemple, dans la question $P(Vrot \mid [Dir_t = 6]\ [Prox_t = 1]\ [Dir_{t+\Delta t} = 4]\ [Prox_{t+\Delta t} = 1])$. Cette question correspond à la situation où le robot a le mur derrière lui et cherche à l'avoir sur son côté droit. Il devrait donc tirer $Vrot = 1$, ce qui le ferait tourner sur sa droite. Cependant, du fait de son écart-type élevé, la gaussienne correspondant à $P(Dir_{t+\Delta t}\ Prox_{t+\Delta t} \mid [Dir_t = 6]\ [Prox_t = 1]\ [Vrot = 0])$, que nous avons décrite dans le paragraphe ci-dessus, a une plus grand probabilité pour la coupe en $[Dir_{t+\Delta t} = 4]\ [Prox_{t+\Delta t} = 1]$ que la gaussienne correspondant au bon ordre moteur. La Figure 4.17 montre ces deux gaussiennes et la distribution obtenue sur $Vrot$. On y voit que la valeur de $Vrot$ qui sera très probablement tirée est 0 et que le robot n'atteindra donc pas son but.

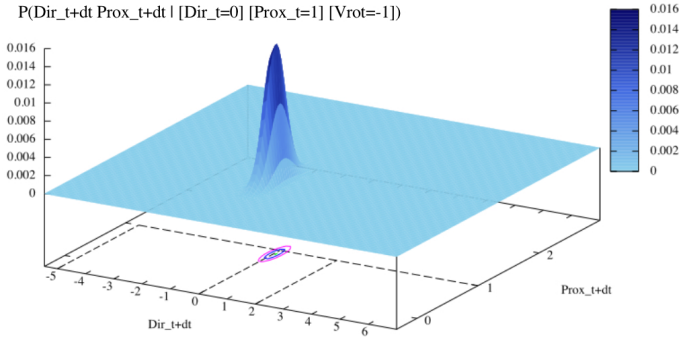
Le problème que nous avons décrit dans cette section est, comme nous l'avons souligné, un problème d'implantation. Ce problème a une solution technique qui consiste à implanter des variables probabilistes « angulaires ». Dans une variable de ce type, les bornes du domaine de valeur sont supposées se rejoindre. C'est le cas sur notre variable $Dir$, où les valeurs 6 et -5 sont en réalités contiguës dans l'espace physique.

### 4.4.2 Choix de $\Delta t$

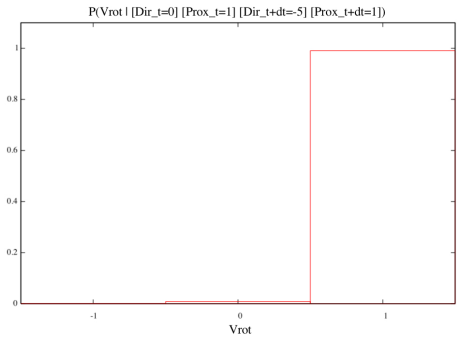
Les résultats que nous avons montrés Section 4.3 ont été obtenus avec $\Delta t = 900$ ms pour l'apprentissage des gaussiennes du terme de prédiction. Nous n'avons pas eu des résultats aussi réussis avec des valeurs de $\Delta t$ trop petites. En effet, $\Delta t$ est la taille de la fenêtre d'observation des variations de $Dir$. La plus petite valeur possible est $\Delta t = 100$ ms : dans ce laps de temps, le robot n'a le temps d'effectuer qu'une très petite rotation, qui n'est souvent pas suffisante pour faire varier la valeur de $Dir$. Au cours de l'apprentissage, les valeurs de $Dir_{t+\Delta t}$ observées sont donc souvent les mêmes que celles de $Dir_t$.

De ce fait, les gaussiennes apprises pour $\Delta t = 100$ ms ont, pour un couple $\langle Dir_t,\ Prox_t \rangle$ donné, des moyennes très semblables pour les différents $Vrot$, car chacune est centrée sur $Dir_{t+\Delta t} = Dir_t$. Pour certaines questions, ces gaussiennes obtenues pour $\Delta t = 100$ ms ne permettent donc pas de déterminer correctement la valeur de $Vrot$ intéressante pour atteindre un but. Ainsi, la Figure 4.18 illustre les trois gaussiennes correspondant à $Dir_t = 0$ et $Prox_t = 1$, obtenues avec $\Delta t = 100$ ms. Le lecteur pourra constater que leur moyenne se situe près de $Dir_{t+\Delta t} = 0$ et pourra les comparer aux gaussiennes obtenues pour $\Delta t = 900$ ms illustrées dans la Figure 4.7, page 39.

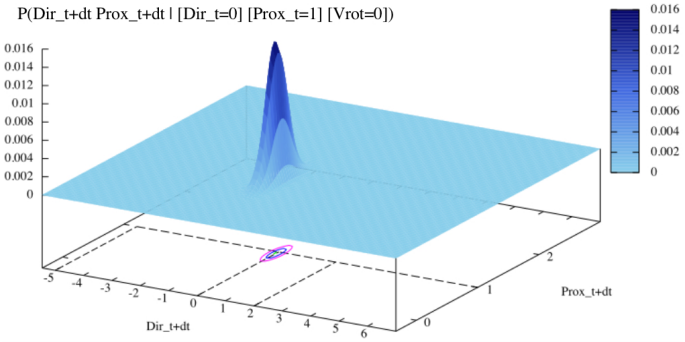
La Figure 4.18 illustre également les distributions sur $Vrot$ obtenues en réponse à trois questions. Ces distributions sont nettement différentes de celles obtenues pour les mêmes questions avec $\Delta t = 900$ ms (Figure 4.8). En effet, seule la distribution Figure 4.18(b) est qualitativement bonne, c'est-à-dire qu'elle permet de tirer aléatoirement la bonne valeur de $Vrot$ avec une grande probabilité. En revanche, la Figure 4.18(d) montre une distribution uniforme, correspondant à la question $P(Vrot \mid [Dir_t = 0]\ [Prox_t = 1]\ [Dir_{t+\Delta t} = 0]\ [Prox_{t+\Delta t} = 1])$. Le robot cherche à rester sur place, mais le terme de prédiction lui indique qu'il va rester sur place quelque soit l'ordre moteur. Il n'y a donc pas de vitesse



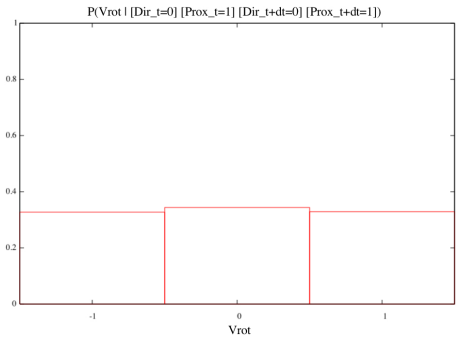

(a) Gaussienne pour $Dir_t = 0$, $Prox_t = 1$ et $Vrot = -1$.



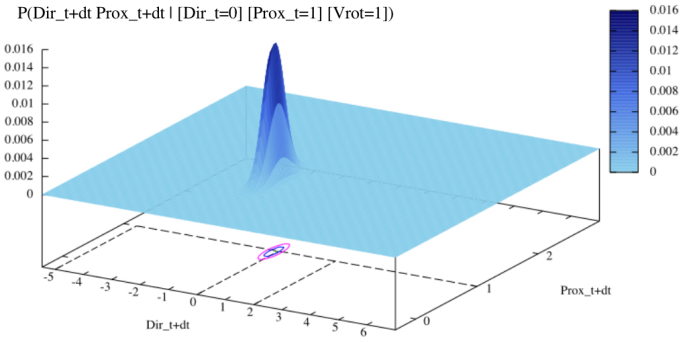

(b) Distribution obtenue pour $Dir_{t+\Delta t} = -5$ et $Prox_{t+\Delta t} = 1$.



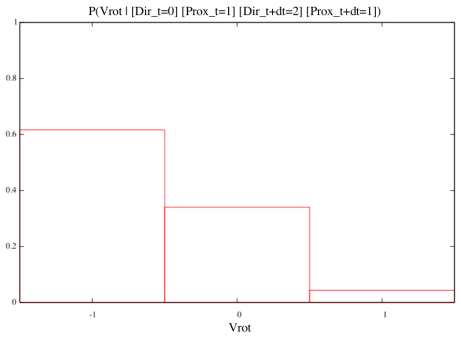

(c) Gaussienne pour $Dir_t = 0$, $Prox_t = 1$ et $Vrot = 0$.




(d) Distribution obtenue pour $Dir_{t+\Delta t} = 0$ et $Prox_{t+\Delta t} = 1$.




(e) Gaussienne pour $Dir_t = 0$, $Prox_t = 1$ et $Vrot = 1$.




(f) Distribution obtenue pour $Dir_{t+\Delta t} = 2$ et $Prox_{t+\Delta t} = 1$.

FIG. 4.18: Exemples de distributions sur $Vrot$ obtenues en réponse à une question, avec $\Delta t = 100$ ms.

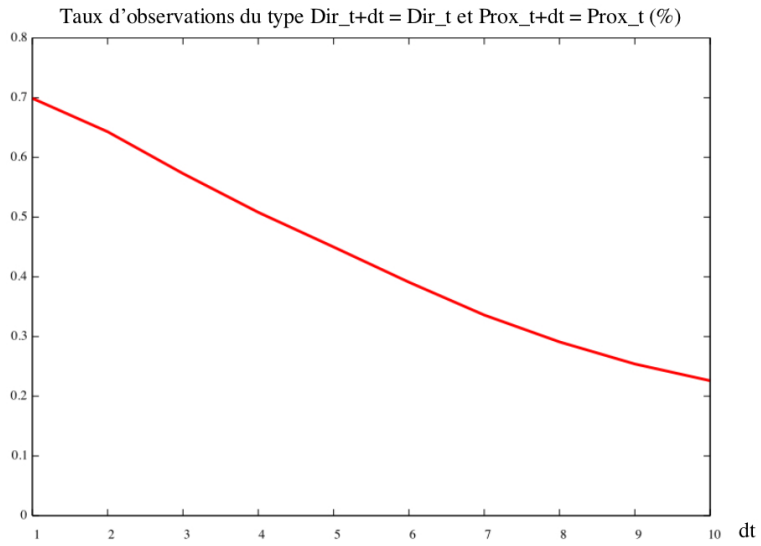



FIG. 4.19: Evolution du taux d'observations de type $Dir_{t+\Delta t} = Dir_t$ et $Prox_{t+\Delta t} = Prox_t$ en fonction de la valeur de $\Delta t$.

de rotation plus pertinente qu'une autre, et on obtient une distribution uniforme. Enfin, la Figure 4.18(f) montre une distribution pour laquelle il est probable de tirer aléatoirement une mauvaise valeur de $Vrot$ (obtenir $Vrot = 0$ alors que le robot cherche à tourner à gauche pour rejoindre $Dir = 2$).

Ce problème vient donc du fait que, plus le $\Delta t$ est faible, plus souvent on observe $Dir_{t+\Delta t} = Dir_t$, au cours de l'apprentissage. La Figure 4.19 montre l'évolution selon $\Delta t$ du nombre d'observations $Dir_{t+\Delta t} = Dir_t$ et $Prox_{t+\Delta t} = Prox_t$ par rapport au nombre d'observations total. On y voit que pour $\Delta t = 100$ ms, environ 70% des observations sont du type $Dir_{t+\Delta t} = Dir_t$. Il est donc logique d'obtenir des gaussiennes telles que celles montrées Figure 4.18.

Dans notre expérience robotique, nous avons résolu ce problème de manière *ad hoc*, c'est-à-dire que nous avons choisi empiriquement un $\Delta t$ approprié (900 ms). Les travaux actuels portent sur la systématisation de cette solution, en proposant un modèle bayésien permettant la comparaison des modèles appris pour différents $\Delta t$. En effet, nous pouvons apprendre expérimentalement une carte bayésienne pour chaque valeur de $\Delta t$, comprise par exemple entre 100 ms et 1 s. On peut ensuite définir un modèle regroupant toutes ces cartes bayésiennes. L'inférence bayésienne nous permet alors de répondre à des questions comme : « étant donné un but, étant donné la situation actuelle, quelle est le $\Delta t$ le plus pertinent ? » D'un point de vue technique, ce modèle regroupant les cartes bayésiennes pourrait être inspiré du modèle d'abstraction de cartes bayésiennes.

### 4.4.3 Le choix du comportement initial

Nous avons montré des résultats obtenus par l'application d'un comportement aléatoire pendant la phase d'apprentissage. Nous avons également testé cette expérience avec

d'autres comportements initiaux. Il s'avère que celui que nous avons présenté ici est le meilleur, c'est-à-dire qu'il permet un meilleur apprentissage de la carte bayésienne.

Par « meilleur apprentissage », nous entendons un apprentissage qui couvre une plus grande partie de l'espace sensori-moteur. Dans notre exemple, nous considérons l'ensemble des triplets $\langle Dir_t, Prox_t, Vrot \rangle$ possibles (noté *total*), et les comparons à l'ensemble de ces triplets observés durant l'apprentissage (noté *obs*). Un comportement donné peut se caractériser par le taux d'exploration de l'espace sensori-moteur $obs/total$.

Par exemple, nous avons essayé d'utiliser un comportement d'évitement d'obstacles pour apprendre la carte bayésienne. Le taux d'exploration de ce comportement est 18,8% pour 5 minutes d'apprentissage. Le robot est donc fortement contraint, lors de sa phase d'apprentissage, dans un sous-espace sensori-moteur. Or, la carte bayésienne est un modèle de l'espace sensori-moteur complet. Le comportement d'évitement d'obstacle ne permet donc pas de l'apprendre correctement. La carte ainsi obtenue peut alors difficilement être utilisée pour définir des comportements concernant d'autres sous-espaces sensori-moteurs que celui couvert par le comportement initial.

Nous avons donc résolu ce problème en définissant un comportement initial dont le but est de couvrir au maximum l'espace sensori-moteur. C'est le comportement de rotation aléatoire sur place que nous avons décrit précédemment.

# Chapitre 5

# Conclusion

## 5.1 Synthèse

Nous avons présenté dans ce document nos travaux sur l'apprentissage des cartes bayésiennes, à partir d'un comportement.

Nous nous sommes tout d'abord attachés à replacer notre contribution dans le contexte des modèles de navigation en robotique mobile. Dans ce domaine, l'essentiel des travaux partagent le même modèle du cycle perception – action. En particulier, nous avons montré dans notre étude bibliographique que ce cycle est commun aux travaux traitant de la localisation et de la planification. Nous plaçons notre contribution dans le cadre d'un modèle du cycle perception – action plus compliqué, faisant intervenir une hiérarchie de représentations de l'espace. Ces représentations sont des cartes bayésiennes.

Ensuite, nous avons décrit une expérience robotique ayant pour but l'apprentissage d'une telle carte bayésienne. Cet apprentissage se base sur un comportement initial, qui est utilisé pour accumuler des données expérimentales. Ces données permettent l'identification des paramètres du terme de prédiction de la carte bayésienne. Ainsi apprise, celle-ci est une ressource que l'on peut utiliser pour générer, par l'inférence probabiliste, de nouveaux comportements.

Enfin, nous avons décrit la mise en œuvre pratique de cette expérience sur un robot mobile Koala. Nous avons présenté des résultats très satisfaisants : le robot a acquis des comportements variés, lui permettant d'atteindre divers buts dans l'espace sensori-moteur. Ces comportements sont applicables dans des environnements variés : il peut par exemple aussi bien suivre les contours d'une arène que ceux d'un ballon. Nous avons ainsi montré qu'une représentation non-orientée de l'interaction sensori-motrice, telle que la carte bayésienne, est plus riche en connaissances qu'un comportement. La variété des comportements que nous avons issus d'une carte bayésienne unique en atteste.

## 5.2 Perspectives

Nous souhaitons maintenant passer en revue nos pistes de recherche, qui soit répondent aux difficultés discutées, soit se placent en continuation de notre contribution.

La première concerne l'implantation de variables probabilistes de type « angulaire ». Cet outil permettrait de traiter correctement des connaissances pour des variables représentant des angles. On pourrait alors résoudre les quelques cas pour lesquels le robot n'arrive pas à atteindre son but. De plus, un tel outil informatique pourrait être inclus dans la librairie probabiliste ProBT $^{(R)}$, en collaboration avec la société ProBayes.

Pour la seconde piste de recherche, nous proposons une étude systématique de l'influence de $\Delta t$ sur la qualité du modèle. En particulier, nous pensons que l'inférence bayésienne offre un cadre mathématiquement clair et rigoureux pour prendre en compte plusieurs modèles, basés chacun sur des constantes de temps différentes.

La troisième piste de recherche concerne l'enrichissement de la carte bayésienne que nous avons apprise. Jusqu'à maintenant, nous avons fait l'hypothèse que les vitesses de rotation et de translation étaient indépendantes. En effet, notre carte bayésienne ne comportait que la variable $Vrot$. Or, nous envisageons d'apprendre une seconde carte bayésienne, qui, elle, ne ferait figurer que la variable $Vtrans$ (la vitesse de translation). Nous pourrions ensuite fusionner les informations de ces deux cartes, en utilisant l'opérateur de superposition de carte bayésienne. La carte ainsi obtenue par superposition est un cadre formel dans lequel peuvent être identifiées expérimentalement les dépendances entre les variables $Vrot$ et $Vtrans$.

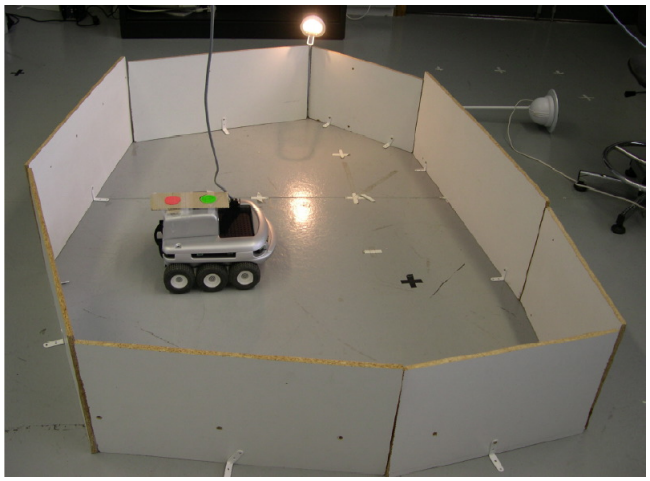


FIG. 5.1: Illustration d'une nouvelle expérience robotique. Le robot apprend une carte bayésienne basée sur la luminosité, en utilisant un comportement d'évitement d'obstacles issu d'une autre carte bayésienne.

Enfin, nous nous intéressons à l'apprentissage d'une carte bayésienne à partir d'une autre carte bayésienne. Nous avons détaillé dans ce travail l'apprentissage d'une carte à partir d'un comportement. La carte apprise permet d'obtenir de nouveaux comportements.

Dans quelle mesure ces nouveaux comportements peuvent-ils être utilisés pour apprendre une autre carte ? Nous pensons qu'une première réponse à cette question pourrait être obtenue en considérant plusieurs modalités sensorielles. Ainsi nous proposons une nouvelle expérience robotique. Dans cette expérience, on donne initialement au robot la carte bayésienne que nous avons apprise dans ce travail. Nous l'utilisons pour lui faire appliquer un comportement d'évitement d'obstacles dans une arène. Cette arène est éclairée par une source lumineuse principale. Pendant sa navigation, le robot enregistre les variations de variables liées à la luminosité. Il peut alors apprendre une nouvelle carte bayésienne basée sur ces variables sensorielles. De cette nouvelle carte, on peut tirer de nouveaux comportements, qui permettent de résoudre des tâches liées à la source lumineuse, comme par exemple de s'en rapprocher (phototaxie) ou de s'en éloigner (photophobie).

À plus long terme, nous souhaitons étudier le statut des questions probabilistes que nous avons définies pour obtenir des comportements. En effet, dans notre travail, nous avons laissé au programmeur le soin de fournir ces questions au robot. Est-il possible de trouver un mécanisme général permettant au robot de définir lui-même ces questions ? Une direction de travail consisterait à établir un parallèle avec le monde du vivant. Par exemple, il semble que, au cours de son développement, l'enfant exploite des processus d'imitation pour acquérir des connaissances. Se pourrait-il que ces imitations lui permettent d'obtenir de son entourage des buts à atteindre dans son espace sensori-moteur ? Un mécanisme similaire permettrait-il au robot de se définir des buts et les questions probabilistes correspondantes ?

# Bibliographie


[1] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. Obprm : An obstacle-based prm for 3d workspaces. In *Proceedings of the Workshop on Algorithmic Foundations of Robotics (WAFR'98)*, pages 155–168, 1998.

[2] N. M. Amato, K. A. Dill, and G. Song. Using motion planning to map protein folding landscapes and analyse folding kinetics of known native structures. In *6th International Conference on Computational Molecular Biology*, 2002.

[3] P. Bessière, J.-M. Ahuactzin, O. Aycard, D. Bellot, F. Colas, C. Coué, J. Diard, R. Garcia, C. Koike, O. Lebeltel, R. LeHy, O. Malrait, E. Mazer, K. Mekhnacha, C. Pradalier, and A. Spalanzani. Survey : Probabilistic methodology and techniques for artefact conception and development. Technical Report RR-4730, INRIA Rhône-Alpes, Montbonnot, France, 2003.

[4] R. Bohlin and L. E. Kavraki. Path planning using lazy prm. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA00)*, 2000.

[5] V. Boor, M. H. Overmars, and A. F. van der Stappen. Gaussian sampling for probabilistic roadmap planners.

[6] J. Diard. *La carte bayésienne un modèle probabiliste hiérarchique pour la navigation en robotique mobile*. PhD thesis, Institut National Polytechnique de Grenoble, 2003.

[7] J. Diard, P. Bessière, and E. Mazer. A survey of probabilistic models, using the bayesian programming methodology as a unifying framework. In *The Second International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2003)*, Singapore, December 2003.

[8] J. Diard, P. Bessière, and E. Mazer. Hierarchies of probabilistic models of navigation : the bayesian map and the abstraction operator. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA04)*, pages 3837–3842, New Orleans, LA, USA, 2004.

[9] J. Diard, P. Bessière, and E. Mazer. A theoretical comparison of probabilistic and biomimetic models of mobile robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA04)*, pages 933–938, New Orleans, LA, USA, 2004.

[10] M. Hauskrecht, N. Meuleau, L. P. Kaelbling, T. Dean, and C. Boutilier. Hierarchical solution of Markov decision processes using macro-actions. In G. F. Cooper and S. Mo-

ral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 220–229, San Francisco, July, 24–26 1998. Morgan Kaufmann.

[11] D. Hsu, L. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In *In Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 141–154, 1998.

[12] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *International Journal of Computational Geometry & Applications*, 9(4 & 5) :495–512, 1999.

[13] E. T. Jaynes. *Probability Theory : The Logic of Science*. Cambridge University Press, June 2003.

[14] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2) :99–134, 1998.

[15] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. on Robotics and Automation*, 12(4) :566–580, 1996.

[16] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[17] O. Lebeltel. *Programmation bayésienne des robots*. PhD thesis, Institut National Polytechnique de Grenoble, 1999.

[18] O. Lebeltel, P. Bessière, J. Diard, and E. Mazer. Bayesian robot programming. *Autonomous Robots (in press)*, 16(1) :49–79, 2004.

[19] N. Nilsson. A mobile automaton : An application of artificial intelligence techniques. In *1st International Conference on Artificial Intelligence*, pages 509–520, 1969.

[20] C. O'Dunlaing and C. Yap. A retraction method for planning the motion of a disc. *J. of Algorithms*, 6 :104–111, 1982.

[21] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1) :21–71, 1998.

[22] S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4) :93–109, 2000.

[23] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. Motion planning for a rigid body using random networks on the medial axis free space. In *Proceedings of the 15th Annual ACM Symposium on Computational Geometry*, pages 173–180, 1999.