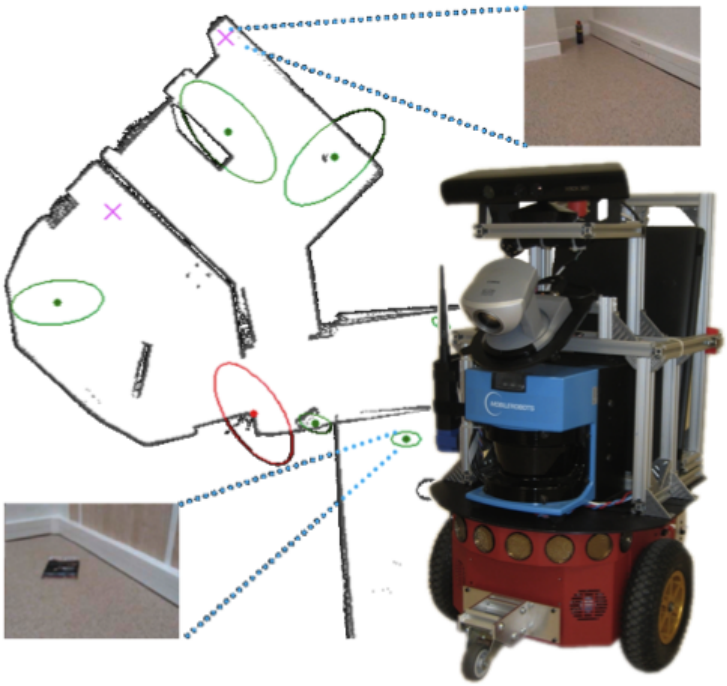
David FILLIAT
École Nationale Supérieure de Techniques Avancées Paris

Robotique Mobile



Année 2022

Ce document évolue régulièrement. La dernière version se trouve sur ma page personnelle : http://www.ensta-paris.fr/~filliat/

Si vous avez des remarques ou des suggestions pour m'aider à le faire progresser, ou simplement si il vous a été utile, n'hésitez pas à m'écrire : david.filliat@ensta-paris.fr

Dernière mise à jour : January 14, 2022

Contents

Chapter 1

Introduction

1.1 Robot Mobile

La robotique est un très bon exemple de domaine pluri-disciplinaire qui implique de nombreuses thématiques telles que la mécanique, la mécatronique, l'électronique, l'automatique, l'informatique ou l'intelligence artificielle. En fonction du domaine d'origine des auteurs, il existe donc diverses définitions du terme robot, mais elles tournent en général autour de celle-ci :

Un robot est une machine équipée de capacités de perception, de décision et d'action qui lui permettent d'agir de manière autonome dans son environnement en fonction de la perception qu'il en a.

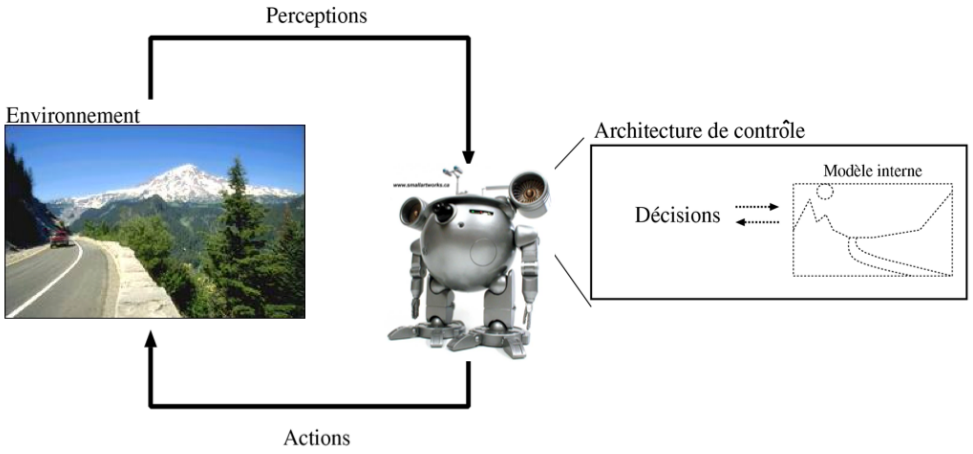


Figure 1.1: Schéma des interactions d'un robot avec son environnement. Selon les approches, un modèle interne de l'environnement peut être utilisé ou non.

Cette définition s'illustre par un schéma classique des interactions d'un robot avec son environnement (Figure 1.1). Les différentes notions que nous présenterons dans ce cours sont essentiellement issues de cette vision de la robotique, très orientée sur l'Intelligence Artificielle,

qui place au centre des préoccupations l'enchaînement de ce *cycle Perception/Décision/Action*. La manière dont un robot gère ces différents éléments est définie par son *architecture de contrôle*, qui la plupart du temps va faire appel à un *modèle interne* de l'environnement qui lui permettra de planifier ses actions à long terme. Nous verrons cependant qu'un certain nombre de méthodes de navigation peuvent ne pas utiliser de modèle interne.

1.2 Objectifs du cours

L'objectif de ce cours est de fournir un aperçu des problèmes de la robotique mobile et des solutions actuelles. Ce cours se veut proche de la recherche, en présentant des méthodes apparues dans les dernières années, mais présente également les notions de base nécessaires à leur compréhension, ainsi qu'un panorama de techniques classiques dont la portée va au delà de leur application en robotique mobile. La lecture des nombreuses références à des articles scientifiques ou à des ouvrages de référence (la plupart du temps en anglais) n'est évidement pas utile pour la compréhension du cours, mais doit permettre d'approfondir des points particuliers hors de la portée de ce cours.

La robotique mobile est un domaine dans lequel l'expérience pratique est particulièrement illustratrice et importante pour la compréhension des problèmes. Au delà des méthodes présentée dans ce texte, les travaux dirigés ou des projets pratiques associés que réalisent les étudiants apporteront également leur lot de connaissances irremplaçables.

1.3 (Très) Bref aperçu historique

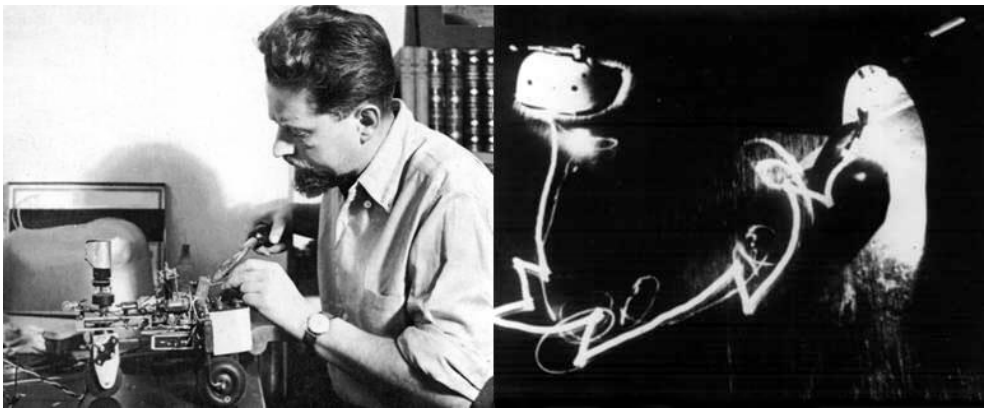


Figure 1.2: La tortue de Grey Walter (nommée "machina speculatrix" et surnommée Elsie) et une illustration de sa trajectoire pour rejoindre sa niche.

Le terme de robot apparaît pour la première fois dans une pièce de Karel Capek en 1920 : Rossum's Universal Robots. Il vient du tchèque 'robota' ($\sim$ servitude) et présente une vision des robots comme serviteurs dociles et efficaces pour réaliser les taches pénibles mais qui déjà vont se rebeller contre leurs créateurs.

La Tortue construite par Grey Walter dans les année 1950 (Figure 1.2), est l'un des premiers robots mobiles autonomes. Grey Walter n'utilise que quelques composants analogiques, dont des tubes à vide, mais son robot est capable de se diriger vers une lumière qui marque un but, de s'arrêter face à des obstacles et de recharger ses batteries lorsqu'il arrive dans sa niche. Toutes ces fonctions sont réalisées dans un environnement entièrement préparé, mais restent des fonctions de base qui sont toujours des sujets de recherche et de développement technologiques pour les rendre de plus en plus génériques et robustes.

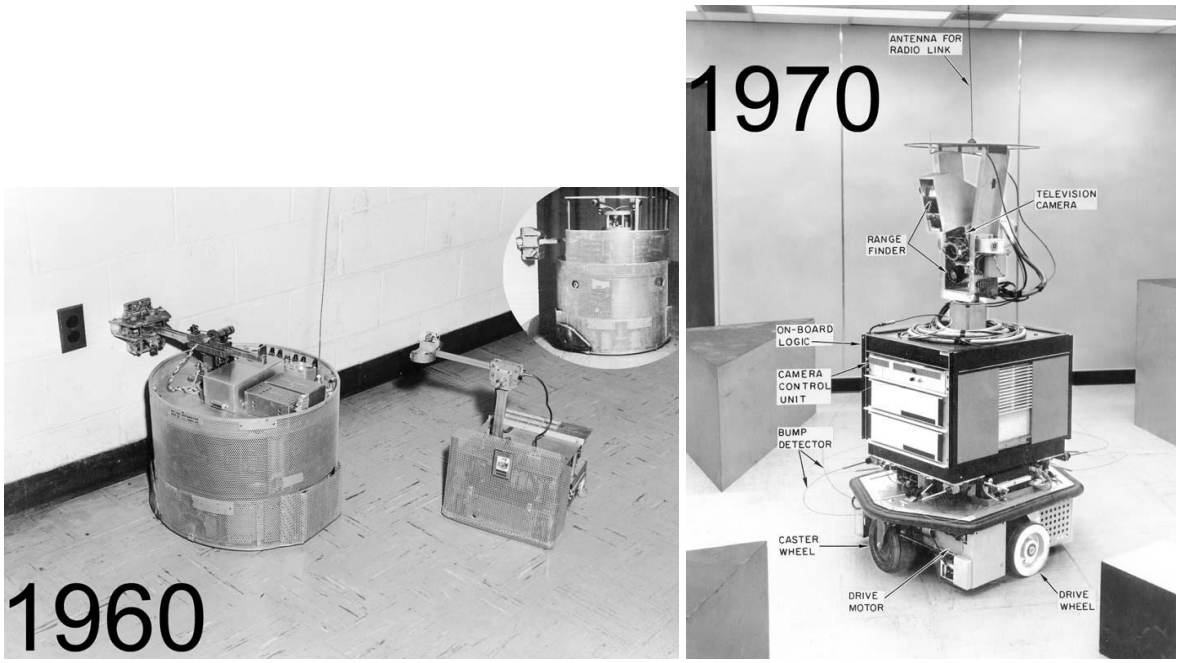


Figure 1.3: A gauche : Robot "Beast" de l'université John Hopkins dans les années 1960. A droite : Le robot Shakey de Stanford en 1969 a été une plate-forme de démonstration des recherches en intelligence artificielle.

Dans les années 60, les recherches en électronique vont conduire, avec l'apparition du transistor, à des robots plus complexes mais qui vont réaliser des tâches similaires. Ainsi le robot "Beast" (Figure 1.3) de l'université John Hopkins est capable de se déplacer au centre des couloirs en utilisant des capteurs ultrason, de chercher des prises électriques (noires sur des murs blanc) en utilisant des photo-diodes et de s'y recharger.

Les premier liens entre la recherche en intelligence artificielle et la robotique apparaissent à Stanford en 1969 avec Shakey (Figure 1.3). Ce robot utilise des télémètres à ultrason et une caméra et sert de plate-forme pour la recherche en intelligence artificielle, qui à l'époque travaille essentiellement sur des approches symboliques de la planification. La perception de l'environnement, qui à l'époque est considérée comme un problème séparé, voire secondaire, se révèle particulièrement complexe et conduit là aussi à de fortes contraintes sur l'environnement. Ces développements de poursuivent avec le Stanford Cart dans la fin des années 1970, avec notamment les premières utilisations de la stéréo-vision pour la détection d'obstacles et la modélisation de l'environnement. En France, le robot Hilare est le premier robot construit au LAAS, à

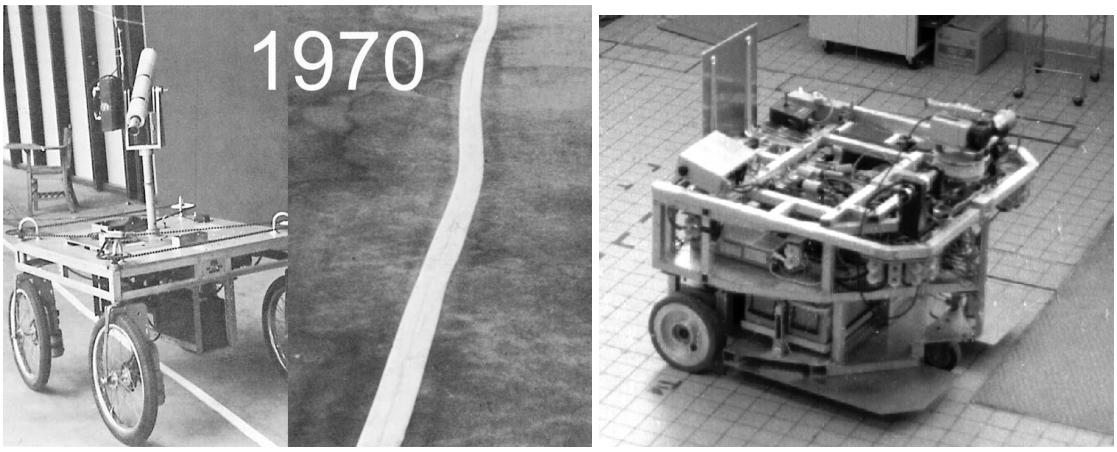


Figure 1.4: Le Stanford Cart date de la fin des années 1970. Le robot Hilare du LAAS a été construit en 1977.

Toulouse (Figure 1.4).

Une étape importante est à signaler au début des années 1990 avec la mise en avant de la robotique réactive, représentée notamment par Rodney Brooks. Cette nouvelle approche de la robotique, qui met la perception au centre de la problématique, a permis de passer de gros robots très lents à de petits robots (Figure 1.5), beaucoup plus réactifs et adaptés à leur environnement. Ces robots n'utilisent pas ou peu de modélisation du monde, problématique qui s'est avérée être extrêmement complexe.

Figure 1.5: Genghis, développé par Rodney Brooks au MIT au début des années 1990.

Ces développements ont continué et l'arrivée sur le marché depuis les années 1990 de plates-formes intégrées telles que le pioneer de la société Mobile Robots a permis à de très nombreux laboratoires de travailler sur la robotique mobile et à conduit à une explosion de la diversité des thèmes de recherche. Ainsi, même si les problèmes de déplacement dans l'espace et de modélisation de l'environnement restent difficiles et cruciaux, des laboratoires ont pu par exemple tra-

vailler sur des approches multi-robot, la problématique de l'apprentissage ou sur les problèmes d'interactions entre les hommes et les robots.

1.4 Exemples d'applications

Aujourd'hui, le marché commercial de la robotique mobile est toujours relativement restreint en dehors des robots aspirateurs vendus à plusieurs millions d'exemplaires. Cependant, il existe de nombreuses perspectives de développement qui en feront probablement un domaine important dans le futur. Les applications des robots peuvent se trouver dans de nombreuses activités "ennuyeuses, salissantes ou dangereuses" (3 D's en anglais pour Dull, Dirty, Dangerous), mais également pour des applications ludiques ou de service, comme l'assistance aux personnes âgées ou handicapées.

Figure 1.6: Exemples de robots commerciaux ou de recherche.

Parmi les domaines d'applications possibles de la robotique, citons :

- La robotique de service (hôpital, bureaux, maison),
- La robotique de loisir (jouets, robot 'compagnon'),
- La robotique industrielle ou agricole (entrepôts logistiques, récolte de productions agricoles, mines),
- La robotique en environnement dangereux (spatial, industriel, militaire, catastrophes naturelles).

A cela, s'ajoute a l'heure actuelle des nombreuses plates-formes conçues essentiellement pour les laboratoires de recherche. La figure 1.6 montre quelques exemples de robots existants.

1.5 Pour aller plus loin

Les illustrations de ce chapitre sont, entre autre, tirées du livre "ROBOT: mere machine to transcendent mind" de Hans Moravec [102], dont les illustrations sont disponibles en ligne[1].

Le livre de Daniel Ichbiah "Robots, Genèse d'un peuple artificiel" [71] donne également un bon aperçu "grand public" de la robotique et de sont histoire[2].

[1] http://www.frc.ri.cmu.edu/ hpm/book98/
[2] http://ichbiah.online.fr/pagerobots.htm

Part I

Les bases de la navigation

Dans cette partie, nous présentons les différentes catégories de méthodes de navigation utilisables pour un robot mobile et les architectures de contrôle associées. Nous présentons ensuite les informations qu'un robot pourra utiliser pour se déplacer, ainsi que les capteurs et les plates-formes couramment utilisées en robotique.

Chapter 2

Les différents types de navigation

2.1 Les stratégies de navigation

Les stratégies de navigation permettant à un robot mobile de se déplacer pour rejoindre un but sont extrêmement diverses, de même que les classifications qui peuvent en être faites. Afin de situer les différentes méthodes de navigation que nous allons étudier dans un contexte général, nous reprenons ici une classification établie par Trullier et al. [138, 140]. Cette classification a été établie en prenant en compte à la fois les stratégies des robots et des animaux. Elle présente l'avantage de distinguer les stratégies sans modèles internes et les stratégies avec modèle interne.

Cette classification comporte cinq catégories, de la plus simple à la plus complexe :

- **Approche d'un objet :** cette capacité de base permet de se diriger vers un objet visible depuis la position courante du robot. Elle est en général réalisée par une remontée de gradient basée sur la perception de l'objet, comme dans l'exemple célèbre des *véhicules* de Valentino Braitenberg [19] (voir section 5.1) qui utilisent deux capteurs de lumière pour atteindre ou fuir une source lumineuse. Cette stratégie utilise des *actions réflexes*, dans lesquelles chaque perception est directement associée à une action. C'est une stratégie locale, c'est-à-dire fonctionnelle uniquement dans la zone de l'environnement pour laquelle le but est visible.

- **Guidage :** cette capacité permet d'atteindre un but qui n'est pas un objet matériel directement visible, mais un point de l'espace caractérisé par la configuration spatiale d'un ensemble d'objets remarquables, ou *amers*, qui l'entourent ou qui en sont voisins. La stratégie de navigation, souvent une descente de gradient également, consiste alors à se diriger dans la direction qui permet de reproduire cette configuration. Cette capacité semble utilisée par certains insectes, comme les abeilles [26], et a été utilisée sur divers robots [54, 86, 58, 114] (voir sections 5.2 et 5.3). Cette stratégie utilise également des actions réflexes et réalise une navigation locale qui requiert que les amers caractérisant le but soient visibles.

- **Action associée à un lieu :** cette capacité est la première capacité réalisant une navigation globale, c'est-à-dire qui permette de rejoindre un but depuis des positions pour lesquelles

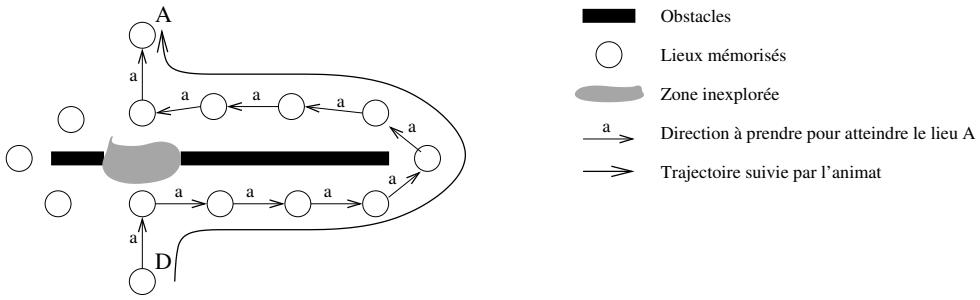


Figure 2.1: **Action associée à un lieu.** *En chaque lieu, représenté par un cercle, l'action à accomplir pour rejoindre le but A est représentée par une flèche indiquant la direction à suivre à partir de ce lieu. Cette stratégie permet de rejoindre un but distant dans l'environnement mais repose sur des chemins figés. Dans cet exemple, le chemin joignant le lieu D au lieu A et passant par la droite de l'obstacle a été appris. Rejoindre le lieu A depuis le lieu D ne pourra alors être réalisé que par ce chemin. Le raccourci empruntant le chemin de gauche, par exemple, est inutilisable.*

ce but ou les amers qui caractérisent son emplacement sont invisibles (par exemple [114]. Elle requiert une représentation interne de l'environnement qui consiste à définir des lieux comme des zones de l'espace dans lesquelles les perceptions restent similaires, et à associer une action à effectuer à chacun de ces lieux (cf. figure 2.1). L'enchaînement des actions associées à chacun des lieux reconnus définit une *route* qui permet de rejoindre le but. Ces modèles permettent donc une autonomie plus importante mais sont limités à un but fixé. Une route qui permet de rejoindre un but ne pourra en effet pas être utilisée pour rejoindre un but différent. Changer de but entraînera l'apprentissage d'une nouvelle route, indépendante des routes permettant de rejoindre les autres buts.

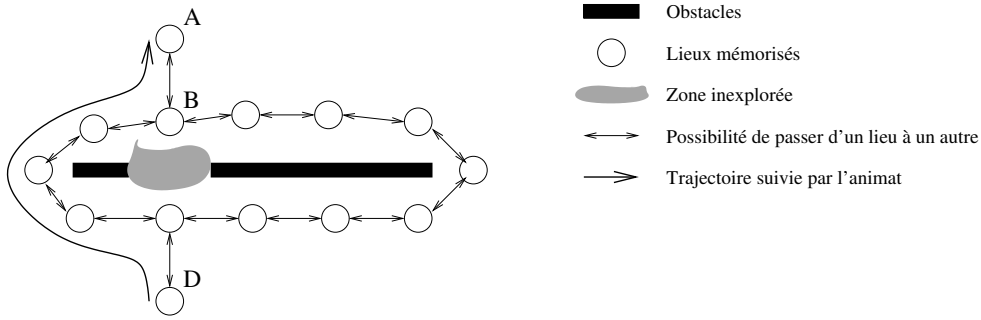


Figure 2.2: **Navigation topologique.** *Cette stratégie permet de mémoriser un ensemble de lieux et les possibilités de passer de l'un à l'autre, indépendamment de tout but. Pour rejoindre un but, il faut alors une étape de planification qui permet de rechercher, parmi tous les chemins possibles, le chemin rejoignant le but. Dans notre exemple, le chemin le plus court entre D et A peut alors être calculé, mais uniquement parmi les lieux et les chemins déjà connus. Cette stratégie permet, par exemple, de contourner l'obstacle par la gauche mais ne permet pas de le traverser en ligne droite de D à A.*

- **Navigation topologique :** cette capacité est une extension de la précédente qui mémorise dans le modèle interne les relations spatiales entre les différents lieux. Ces relations indiquent la possibilité de se déplacer d'un lieu à un autre, mais ne sont plus associées à un but particulier. Ainsi le modèle interne est un graphe qui permet de calculer différents chemins entre deux lieux arbitraires. Ce modèle ne permet toutefois que la planification de déplacements parmi les lieux connus et suivant les chemins connus (cf. figure 2.2).

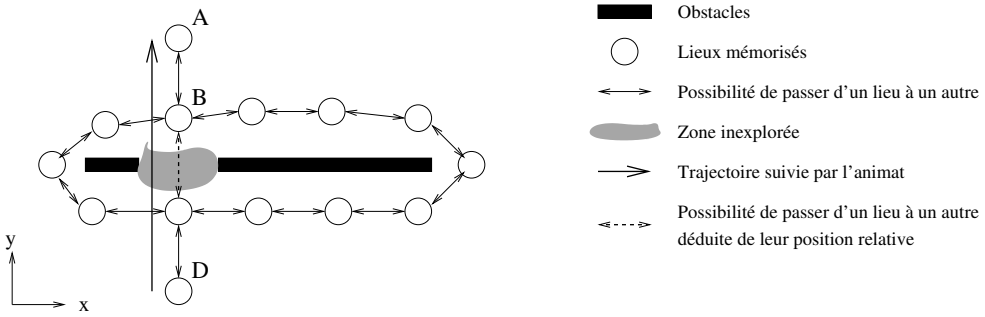


Figure 2.3: **Navigation métrique.** *Cette stratégie permet de calculer le chemin le plus court entre deux lieux mémorisés, permettant même de planifier des raccourcis au sein de zones inexplorées de l'environnement. Pour cela, la carte mémorise la position métrique relative de chacun des lieux visités par le robot. Ainsi il est possible de prévoir un déplacement entre deux lieux, même si la possibilité de ce déplacement n'est pas enregistrée dans la carte. Dans cet exemple, cette stratégie permet de d'aller du lieu A au lieu D en traversant la zone inexplorée.*

- **Navigation métrique :** cette capacité est une extension de la précédente car elle permet au robot de planifier des chemins au sein de zones inexplorées de son environnement. Elle mémorise pour cela les positions métriques relatives des différents lieux, en plus de la possibilité de passer de l'un à l'autre. Ces positions relatives permettent, par simple composition de vecteurs, de calculer une trajectoire allant d'un lieu à un autre, même si la possibilité de ce déplacement n'a pas été mémorisée sous forme d'un lien (cf. figure 2.3).

Les modèles des trois premières catégories utilisent des actions réflexes pour guider le robot et se différencient essentiellement par le type de perceptions utilisées pour déclencher ces actions. Ils se regroupent sous le terme générique de *navigation réactive* dont nous parlerons dans la partie II. Ils peuvent être très simple, ne nécessitent pas de modèle global de l'environnement mais ont un domaine d'application souvent restreint. Dans le monde vivant, ces stratégies sont très répandues, notamment chez les insectes. Les comportements de ce type restent toutefois essentiels dans les robots modernes car, du fait de leur simplicité, il sont généralement exécutés très rapidement et ils permettent de réaliser des taches de bas-niveau, comme l'évitement des obstacles imprévus, essentielles à la sécurité d'un robot.

Les modèles des deux dernières catégories autorisent pour leur part une navigation globale et permettent de rejoindre un but arbitraire au sein de l'environnement. Ils s'appuient pour cela sur un modèle interne du monde, une carte, qui supporte une *planification*. Ce modèle interne

mémorise donc la structure spatiale de l'environnement, indépendamment d'un but précis. Chacune des positions mémorisées dans ce modèle interne peut alors être utilisée comme but par le processus de planification dont le rôle est de calculer une route vers ce but. Ce sont ces deux stratégies qui sont regroupées sous le terme de navigation par carte, objet du chapitre III .

Une telle représentation interne est naturelle pour les êtres humains, pour lesquels des processus cognitifs de haut niveau sont utilisés pour créer et utiliser une carte. Ces processus de haut niveau sont toutefois très difficile à copier pour un robot réel qui ne dispose que de systèmes rudimentaires de perception et de traitement des informations en comparaison avec un homme. Par exemple, en environnement urbain, le processus de mise en correspondance de la carte avec l'environnement réel afin de déterminer sa position fait souvent appel, pour l'homme, à la lecture du nom des rues inscrit sur les bâtiments, ce qui est relativement difficile à automatiser, à cause de la diversité des configurations dans lesquelles peuvent ce trouver ces noms. On notera au passage que l'homme a quasiment toujours recours à des aménagements particuliers de l'environnement pour connaitre sa position, par exemple celui qui consiste à nommer les rues ou à lancer des satellites dans l'espace pour bénéficier du GPS. Le système de navigation idéal pour un robot mobile sera probablement celui qui sera capable de tirer partie de toutes ces informations, qui ne lui étaient pas destinées à l'origine.

L'utilisation de cartes par un robot mobile comme le font les hommes est probablement hors de notre portée pendant quelques années, cependant il existe également des preuves de l'existence de représentations internes similaires à de telles cartes chez les animaux, par exemple chez les rats. Ces représentations sont identifiables au niveau neurologiques dans certaines parties de leur cerveau, notamment dans l'hippocampe. Cela montre que des cartes sont utilisée par des êtres vivants, sans le support de concept abstraits tels que les utilisent les humains. Ce type de carte qui fait appel à des structures neurologiques de base et probablement à des perceptions relativement simples, est un paradigme intéressant pour les robots mobiles.

En robotique mobile, comme pour l'homme ou certains animaux, l'utilisation de cartes est quasiment indispensable pour permettre d'effectuer des tâches de navigation dans des conditions environnementales complexes, qui ne sont pas spécialement adaptées pour le robot. La construction et l'utilisation de telles cartes posent cependant de nombreux problèmes, notamment pour garantir l'adéquation entre la carte et le monde réel. Pour cette raison, la plupart des robots trouvent aujourd'hui un compromis entre une approche réactive et une approche utilisant une carte afin de bénéficier de la rapidité et de la robustesse de la première et de la capacité de déplacement à long terme de la seconde.

2.2 Les architectures de contrôle

Un robot est un système complexe qui doit satisfaire à des exigences variées et parfois contradictoires. Un exemple typique pour un robot mobile est l'arbitrage qui doit être fait entre l'exécution la plus précise possible d'un plan préétabli pour atteindre un but et la prise en compte d'éléments imprévus, tels que les obstacles mobiles. Ces arbitrages, que ce soit au niveau du choix de stratégie, ou au niveau de l'utilisation des capteurs, des effecteurs ou des ressources de calcul, sont réglés par un ensemble logiciel appelé architecture de contrôle du robot. Cette architecture

permet donc d'organiser les relations entre les trois grandes fonctions que sont la perception, la décision et l'action .

Nous pouvons reprendre la définition de Ronald Arkin [4] de l'art de concevoir de telles architectures :

Robotic architecture is the discipline devoted to the design of highly specific and individual robots from a collection of common software building blocks.

Selon cette définition une architecture doit donc être conçue pour un robot précis, mais en utilisant des modules génériques. De manière plus générale il existe également des règles de conception relativement générales qui permettent de réaliser ces implémentations. En fonction de ces règles, les architectures de contrôle peuvent être classées en trois grandes catégories que nous détaillerons par la suite : les contrôleurs hiérarchiques, les contrôleurs réactifs et les contrôleurs hybrides (Figure 2.4). Comme le précise cette définition, toutes ces architectures ne diffèrent pas forcement par les méthodes élémentaires employées mais plutôt par leur agencement et leur relations.

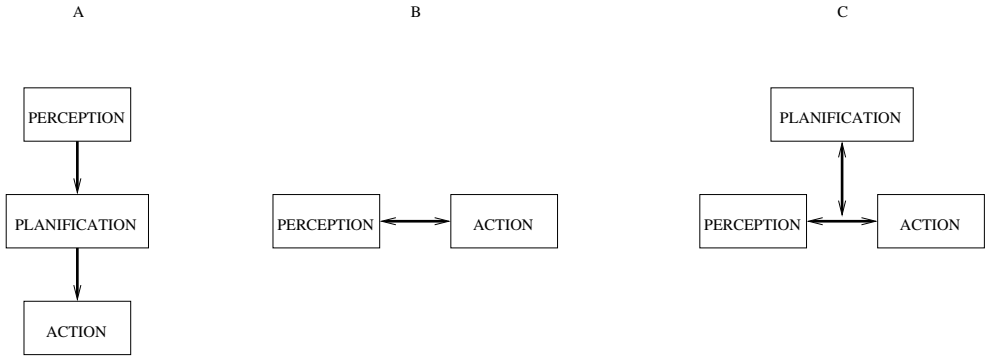


Figure 2.4: Illustration des architectures classiques des contrôleurs pour les robots mobiles : Hiérarchique (A), Réactive (B) et Hybride (C) (Voir le texte pour la description détaillée).

2.2.1 Contrôleurs Hiérarchiques

Historiquement, les premiers robots mobiles dérivés des recherches en intelligence artificielle utilisaient des *contrôleurs hiérarchiques* (cf. figure 2.4 A) dont le fonctionnement repose essentiellement sur la capacité de décision travaillant sur un modèle du monde supposé quasiment-parfait. Ces architectures fonctionnent selon un cycle rigide de modélisation de l'environnement, planification des actions au sein de cette représentation, puis exécution du plan. La capacité de décision était issue des premiers travaux en intelligence artificielle et reposait essentiellement sur des traitements de données symboliques. Ces architectures ont rapidement montré leurs limites et leur incapacité à fonctionner dans un environnement qui ne soit pas statique et simplifié à l'extrême.

L'essentiel des problèmes de ces architectures provient de l'utilisation d'un modèle interne central qui est le seul pris en compte pour guider le robot. Elles se trouvent confrontées à tout les problèmes des premiers développements de l'intelligence artificielle symbolique.

Ces architectures supposent premièrement qu'un modèle informatique du monde puisse représenter toutes les informations pertinentes pour le déplacement du robot. Or un tel modèle ne peut être suffisant dans un environnement dynamique car au moment de la réalisation de l'action l'environnement peut avoir suffisamment changé pour que la décision ne soit plus valide. Ce problème était particulièrement crucial au début de la robotique mobile ou les capacités de calcul limitées entraînaient des temps de planification de l'ordre de plusieurs dizaines de minutes. Mais l'augmentation des capacités de calcul ne suffit pas à résoudre ce problème qui ne permet pas de gérer un environnement de travail réaliste. C'est un problème intrinsèque lié d'une part à la trop grande longueur de la boucle qui relie la perception à l'action et d'autre part à l'invalidité de l'hypothèse de monde clos faite en intelligence artificielle [1].

De plus, ces architectures permettent peu de contrôle sur l'exécution des actions. En effet, une fois l'action choisie, elle est exécutée en supposant le modèle du monde correct et il n'y a pas de retour direct de la perception sur l'exécution de l'action. Les écarts modèles/environnement ne peuvent être pris en compte que via un nouveau cycle perception/modélisation/planification, ce qui, par définition, est très peu réactif et conduit rapidement à de graves problèmes.

2.2.2 Contrôleurs réactifs

Rodney Brooks [20] a proposé une solution radicale à tous ces problèmes sous la forme d'une *architecture réactive* (cf. figure 2.4 B). Dans cette architecture, un ensemble de comportements réactifs, fonctionnant en parallèle, contrôle le robot sans utiliser de modèle du monde. Cette architecture supprime évidemment les problèmes dûs aux différences entre la réalité, d'une part, et le modèle de l'environnement du robot, d'autre part, mais limite clairement les tâches que peut effectuer le robot (cf. [77] pour une critique). En effet, sans représentation interne de l'état de l'environnement, il est très difficile de planifier une suite d'actions en fonction d'un but à atteindre. Les robots utilisant cette architecture sont donc en général efficaces pour la tâche précise pour laquelle ils ont été conçus, dans l'environnement pour lequel ils ont été prévus, mais sont souvent difficiles à adapter à une tâche différente.

Les réussites de ces architectures sont liées au couplage direct entre la perception et l'action qui permet une prise en compte très rapide des phénomènes dynamiques de l'environnement. En donc une bonne robustesse dans des environnements complexes.

Comme nous l'avons mentionné, ces architectures sont en général basées sur plusieurs comportements : évitement d'obstacles, déplacement aléatoire, déplacement vers un but, fuite d'un point... Pour guider le robot, il faut donc choisir à chaque instant lequel de ces comportements activer. Ce problème est connu dans la litérature scientifique sous le nom de *sélection de l'action*. La solution proposée par Brooks, l'*architecture de subsomption* [21] est devenue un classique et

[1] L'hypothèse de monde clos dit que la représentation symbolique d'un problème va être suffisante pour pouvoir représenter toutes les conséquences des actions réalisées dans ce monde. Cela s'avère impossible en pratique pour des problèmes autre que des problèmes jouets (par exemple un monde de cubes posés sur une table).

utilise une hiérarchie des comportements qui se déclenchent donc selon un ordre de priorité en fonction des perceptions du robot.

2.2.3 Contrôleurs hybrides

La plupart des contrôleurs actuellement utilisés choisissent une solution intermédiaire entre ces deux approches sous la forme d'une *architecture hybride* [104, 3] (cf. figure 2.4 C). Cette architecture se compose de deux niveaux. Le premier est chargé des tâches de navigation de haut niveau, telles que la localisation, la cartographie et la planification. Pour cela, il s'appuie sur un second niveau réactif qui est chargé d'exécuter les commandes avec le plus de précision possible et de gérer les éléments non modélisés de l'environnement tels que les obstacles inconnus ou dynamiques. L'action conjointe de ces deux niveaux permet de réagir rapidement face aux variations imprévues de l'environnement, tout en permettant la réalisation d'actions planifiées à plus long terme.

Le bas niveau de ces architectures peut être réalisé sous forme de comportements, tels que ceux utilisés dans les architectures réactives. Ces comportements sont des boucles sensorimotrices qui relient les action aux perceptions avec un phase de décision très courte, qui assure la réactivité. Dans le même temps, les informations sensorielles sont utilisées par le haut niveau dans une boucle sensorimotrice à une échelle de temps beaucoup plus longue. C'est la mise en parallèles de ces deux échelles de temps qui fait la force de ces architectures.

Les exemples d'architectures hybrides foisonnent (4D/RCS, 3T, Harpic...) car de très nombreux laboratoires et organismes travaillant sur la robotique ont développé leur architecture de ce type.

2.2.4 Pour aller plus loin

Deux livres intéressants sur le sujet :

- Introduction to AI Robotics de Robin Murphy, MIT Press
- Behavior based robotics de Ronald C. Arkin, MIT Press

Chapter 3

Les sources d'information

Tous les capteurs utilisés en robotique mobile fournissent des informations appartenant à l'une de deux grandes catégories d'informations : les informations proprioceptives et les informations extéroceptives.

- Les informations *proprioceptives* sont des informations internes au robot qui le renseignent, dans le cas de la navigation, sur son déplacement dans l'espace. Ces informations peuvent provenir de la mesure de la rotation de ses roues ou de la mesure de l'accélération grâce à une centrale inertielle. Un processus d'intégration permet alors, en accumulant ces informations au cours du temps, d'estimer la position relative de deux points par lesquels le robot est passé.

- Les informations *extéroceptives* ou plus simplement les *perceptions*, sont des informations caractéristiques d'une position que le robot peut acquérir dans son environnement. Ces informations peuvent être de nature très variée. Par exemple, un robot peut mesurer la distance des obstacles avec des capteurs infrarouges ou utiliser une caméra.

Ces deux sources d'information ont des propriétés opposées que nous détaillons dans les deux sections suivantes.

3.1 Informations proprioceptives

Les informations proprioceptives renseignent sur le *déplacement* du robot dans l'espace. Elles constituent donc une source d'information très importante pour la navigation. Cependant, la précision de cette information se dégrade continuellement au cours du temps, la rendant inutilisable comme seule référence à long terme. Cette dégradation continuelle provient de l'intégration temporelle des mesures effectuées par les capteurs internes. En effet, chaque capteur produit une mesure bruitée du déplacement instantané, de la vitesse ou de l'accélération du robot. Ce bruit, via le processus d'intégration qui a pour but d'estimer le déplacement, conduit inévitablement à une erreur croissante.

Malgré ce défaut important, les informations proprioceptives ont l'avantage de dépendre assez peu des conditions environnementales qui perturbent fortement les informations perceptives. La vision, par exemple sera fortement perturbée si l'environnement est plongé dans le noir, mais les informations proprioceptives fourniront une information identique, que l'environnement soit éclairé ou non. De plus, comme nous le verrons dans la section suivante, si deux lieux identiques du point de vue des perceptions se trouvent dans l'environnement, les informations perceptives ne permettent pas de les différencier. Les informations proprioceptives sont alors le seul moyen de les distinguer.

En robotique, cette information a de plus l'avantage de la simplicité de manipulation. En effet, le processus d'intégration fournit directement une estimation de la position du robot dans un espace euclidien doté d'un repère cartésien. Dans ce type de repère, tous les outils de la géométrie mathématique sont utilisables. Ils permettent, par exemple, d'effectuer des calculs de chemin relativement simples lorsque l'on connaît la position du but et des obstacles.

3.2 Informations extéroceptives

Les informations extéroceptives, ou plus simplement les perceptions , fournissent un lien beaucoup plus fort entre le robot et son environnement. En effet, les informations proprioceptives fournissent des informations sur le déplacement du robot, alors que les informations perceptives fournissent des informations directement sur la *position* du robot dans l'environnement. Ces informations assurent un ancrage dans l'environnement, en permettant de choisir des perceptions qui peuvent être utilisées comme points de repère. Ces points de repère sont indépendants des déplacements du robot et pourront être reconnus quelle que soit l'erreur accumulée par les données proprioceptives. La reconnaissance de ces points est évidemment soumise à une incertitude, mais pas à une erreur cumulative, ce qui les rend utilisables comme référence à long terme.

3.2.1 Variabilité perceptuelle

Pour être utile, un système de perception doit donc permettre de distinguer le plus de lieux possible. Pour cela, il doit être capable de distinguer le plus de détails possibles, afin de faire la différence entre deux lieux différents mais d'apparences similaires. Or l'augmentation de cette capacité à distinguer de petites variations dans l'environnement rend le système sensible au problème de la *variabilité perceptuelle* , c'est à dire au changement de perception au cours du temps pour un lieu donné. Cette variabilité peut être due au bruit inhérent au processus de mesure où à des variations de l'environnement non significatives pour le problème de navigation qui nous concerne, par exemple le changement de luminosité. Pour s'affranchir de ce problème, il faut en général mettre en place des processus de traitement des perceptions qui permettront de ne pas dépendre de ces variations et de correctement identifier un lieu donné.

3.2.2 Perceptual aliasing

En cherchant à limiter la dépendance aux variations de l'environnement, le concepteur de robot aboutit en général au problème du *perceptual aliasing* ou d'*Ambiguïté des perceptions*. Ce problème désigne l'incapacité d'un système de perception à distinguer de manière unique tous les lieux d'un environnement. Cette situation est très courante lorsque les robots utilisent des capteurs de distance aux obstacles tels que les capteurs à ultrasons. Dans un environnement intérieur de tels capteurs sont, par exemple, capables de mesurer la position du robot par rapport à un coin, mais ne fournissent aucune information sur la position le long d'un couloir rectiligne. Toutes les positions le long d'un couloir correspondent alors à des perceptions identiques.

Il est possible d'utiliser des capteurs qui fournissent des données plus précises ou plus discriminantes. Dans le cas des capteurs de distance, il est, par exemple, possible d'utiliser un télémètre laser qui pourra distinguer les renfoncements des portes et sera ainsi plus précis. Mais même en utilisant des capteurs plus informatifs, comme une caméra, ce problème finit par apparaitre lorsque la taille de l'environnement augmente. Il existe toujours une limite matérielle ou logicielle au delà de laquelle l'identification unique de toutes les positions d'un environnement est impossible. Il n'est donc pas possible, en général, de régler complètement le problème du *perceptual aliasing*, mais seulement d'en repousser l'apparition. Il faut donc bien étudier les capteurs nécessaires en fonction des traitements réalisables et de l'environnement visé pour limiter ce problème.

3.2.3 Utilisation directe

Les capteurs sur un robot mobile peuvent être de nature très variée et être utilisés de nombreuses façons différentes. Il est toutefois possible de distinguer deux utilisations distinctes de leurs données pour la navigation. Ces deux utilisations dépendent de l'utilisation ou non d'un modèle métrique associé au capteur, modèle qui permet de traduire les valeurs brutes du capteur en informations sur la géométrie de l'environnement. Ce modèle permet notamment de prévoir la variation des mesures renvoyées par ce capteur en fonction du déplacement du robot.

Les perceptions peuvent être utilisées de manière directe, sans aucun modèle métrique, pour comparer directement deux positions en examinant les perceptions recueillies en ces lieux. Cette méthode ne permet cependant que de reconnaître des lieux de l'environnement préalablement explorés par le robot. Sans modèle de la variation des capteurs, il est en effet impossible de prévoir les valeurs que les capteurs relèveront dans un lieu inexploré, même s'il est proche ou entouré de lieux connus.

Pour une telle utilisation directe, seules deux procédures permettant, d'une part, de mémoriser une perception et, d'autre part, de comparer deux perceptions, sont alors nécessaires. Ces procédures peuvent être mises en œuvre à partir de tous les types de capteurs existants. Il est, par exemple, possible d'utiliser la couleur dominante de l'environnement autour du robot, la température (en supposant qu'elle caractérise une zone de l'environnement, comme pour une chambre froide), la force du signal wifi ou le temps de retour d'une onde sonore quand elle est envoyée dans une direction donnée. La seule propriété utilisée est la constance des valeurs mesurées par un capteur pour un lieu donné. Cette constance permet de reconnaître un lieu déjà

visité ou d'identifier un lieu nouveau dans l'environnement.

3.2.4 Utilisation d'un modèle métrique

La seconde méthode d'utilisation d'un capteur consiste à utiliser un modèle métrique associé . Un tel modèle permet de traduire les informations données par le capteur dans un espace métrique qui est en général le même que celui utilisé pour estimer la position du robot grâce à l'odométrie. Il est ainsi possible d'estimer la position d'objets de l'environnement par rapport au robot, et ainsi de prévoir les données que ce capteur relèvera pour des positions différentes du robot. L'utilisation d'un tel modèle n'est toutefois possible que pour certains capteurs. Il est, par exemple, possible d'utiliser un tel modèle associé à un capteur à ultrasons, à un télémètre laser ou à une paire de caméras stéréoscopique, mais pas à un capteur d'odeur.

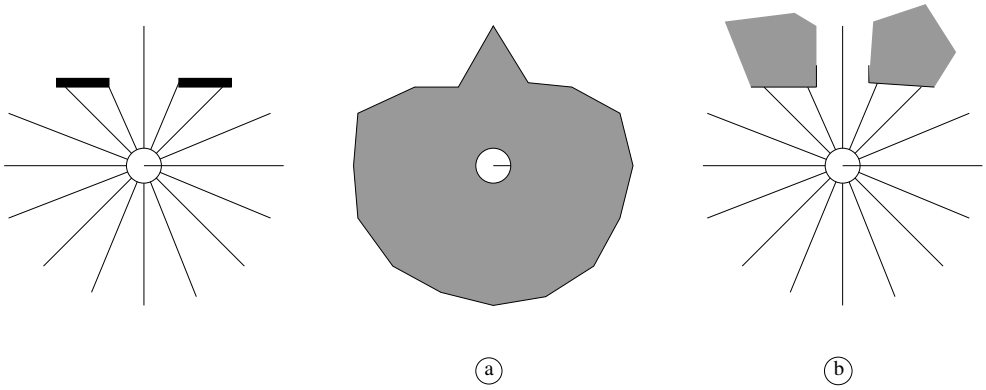


Figure 3.1: Un modèle métrique pour un capteur permet deux utilisations de ses données. La première est similaire à celle qui traite de telles données sans modèle métrique et requiert la simple mémorisation de ce qui est perçu en un lieu donné (Partie a). La seconde utilise ces données pour reconstituer les objets rencontrés dans l'environnement, objets qui pourront tous être mémorisés dans un cadre de référence commun, indépendamment de la position depuis laquelle ils ont été perçus (Partie b).

Avec un tel modèle, les valeurs des capteurs peuvent être utilisées simplement pour caractériser chaque lieu atteint par le robot (cf. figure 3.1a). La méthode est alors la même que celle mise en place quand les capteurs sont utilisés sans modèle métrique. L'utilisation d'un modèle métrique présente toutefois l'avantage que les informations recueillies ont une sémantique plus forte et une certaine indépendance au point de vue du robot. En effet, ces informations caractérisent la structure spatiale locale de l'environnement, en plus de la simple apparence de l'environnement depuis la position du robot. Cette structure spatiale peut alors être utilisée lors de la comparaison de différents lieux. Il est par exemple possible de reconnaître un couloir en fonction de sa largeur, indépendamment de la position du robot dans ce couloir. En effet, sans utilisation de modèles métriques, deux perceptions recueillies en des positions différentes du couloir seront simplement différentes. En utilisant un modèle métrique, il est possible de calculer la largeur du couloir, par exemple, à partir des données recueillies et ainsi de déterminer si ces deux positions peuvent correspondre au même couloir.

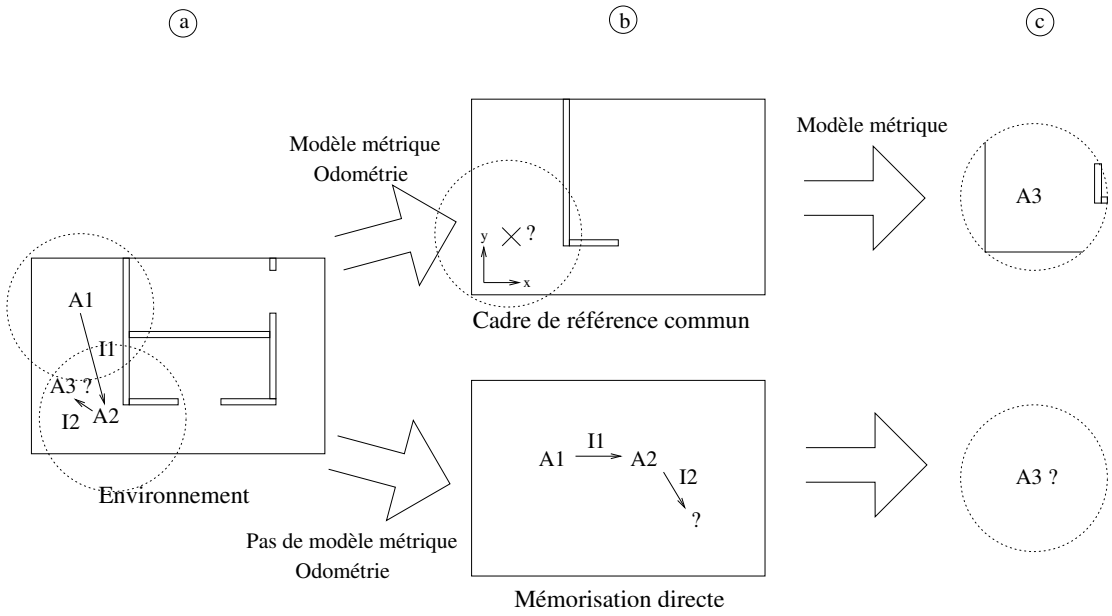


Figure 3.2: Un modèle métrique des perceptions permettent d'inférer les valeurs qui devraient être perçues pour des positions encore non visitées. Dans cet exemple, les données $A1$ et $A2$ sont perçues en deux positions reliées par des données proprioceptives $I1$ (partie a). L'utilisation d'un modèle métrique permet de fusionner ces informations dans un cadre de référence commun où des objets sont représentés, ici deux murs orthogonaux (partie b, haut). Sans modèle métrique, ces données peuvent seulement être mémorisées de manière séparée (partie b, bas). Dans le cas de l'utilisation d'un modèle métrique, les données peuvent ensuite être utilisées pour estimer la perception $A3$ pour une nouvelle position reliée à la précédente par les données proprioceptives $I2$. Ici, le modèle permet d'inférer que les données $A3$ correspondent à un coin de murs (partie c, haut). Sans un tel modèle, seules les positions visitées peuvent être reconnues, et aucune inférence ne peut être faite pour les positions non visitées (partie c, bas).

Cependant, grâce à un modèle métrique, les perceptions peuvent être utilisées de manière différente. En effet, dans l'utilisation précédente, sans modèle métrique, elles sont utilisées pour caractériser l'apparence de l'environnement depuis un lieu. Cette caractérisation ne permet pas d'identifier individuellement des objets distants du robot qui pourraient servir de points de repères, les *amers*. L'utilisation d'un modèle métrique permet l'identification de tels points (cf. figure 3.1b). La perception de ces amers permet alors, en retour, d'obtenir des informations sur la position du robot. Cette utilisation des perceptions offre l'avantage supplémentaire de permettre au robot d'inférer les valeurs que mesureront les capteurs dans des positions différentes, mais voisines de sa position courante (cf. figure 3.2). Par exemple, si un robot perçoit un mur à cinq mètres devant lui, il peut prédire qu'en avançant d'un mètre, il percevra le mur à quatre mètres. Un autre moyen de présenter cette propriété est de dire que les perceptions seules permettent d'estimer la position métrique relative de deux lieux (cf. figure 3.3). Ainsi, si un robot perçoit deux fois un mur devant lui, d'abord à cinq mètres puis à quatre mètres, il pourra en déduire qu'il a avancé d'un mètre. Cette propriété permet au robot d'estimer sa position avec précision sur une part plus importante

de son environnement et ne limite plus la localisation aux lieux déjà visités. Cet avantage est une conséquence directe de la fusion des informations proprioceptives et des perceptions au sein d'une même représentation, qui permet le passage d'un type d'information à l'autre.

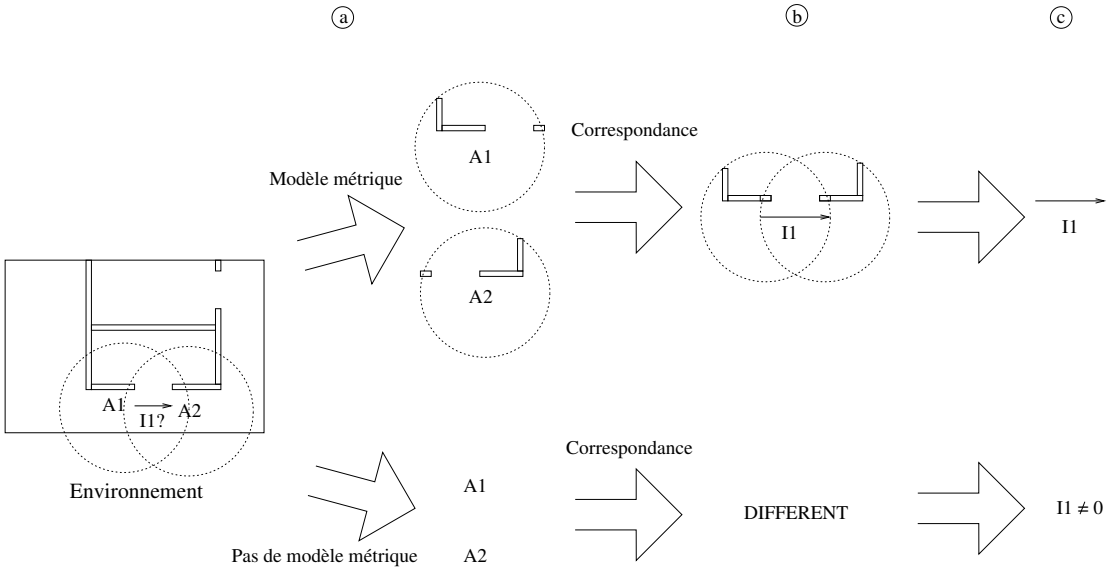


Figure 3.3: Un modèle métrique des capteurs permet d'inférer la position relative $I1$ de deux lieux depuis lesquelles des perceptions $A1$ et $A2$ ont été réalisées (partie a). Cette estimation requiert d'abord la recherche d'un objet de l'environnement commun aux deux perceptions (partie b, haut). L'utilisation de cet objet commun rend alors possible l'estimation de la position relative $I1$ des deux lieux (partie c, haut). Sans modèle métrique, seule la similarité de deux perceptions peut être mesurée (partie b, bas). Il est alors seulement possible d'estimer si ces deux situations peuvent correspondre au même lieu ou non, c'est-à-dire si $I1$ est nulle ou non (partie c, bas).

Toutefois, la mise au point d'un tel modèle métrique peut être difficile. La relation qui lie la valeur mesurée par un capteur à la position des objets du monde réel peut être, en effet, très complexe. Dans le cas des capteurs à ultrasons, par exemple, si un mur se trouve juste dans l'axe du capteur, sa distance est simplement mesurée par le temps mis par l'onde sonore pour revenir au capteur. Mais, dans le cas où le mur est fortement incliné par rapport au capteur, l'écho peut ne pas revenir en direction du capteur qui ne détectera alors aucun obstacle. Un autre problème vient de la texture des murs. Un mur recouvert de textile ou d'un matériau souple renverra les échos très différemment d'un mur de béton. En conséquence, pour une distance donnée, le capteur percevra des distances différentes suivant le matériau des murs. Ces deux exemples montrent que le modèle métrique associé à un capteur ne dépend pas que du capteur. Il dépend aussi fortement de propriétés locales de l'environnement qui sont difficiles ou impossibles à prendre en compte dans un modèle du capteur seul.

3.3 Fusion d'informations

En résumé, les informations proprioceptives sont simples à utiliser, mais dérivent au cours du temps, tandis que les perceptions ne dérivent pas, mais souffrent des problèmes de variabilité perceptuelle et d'Ambiguïté.

La solution pour résoudre ces problèmes est de fusionner ces deux types d'information. Il est par exemple possible d'utiliser les informations proprioceptives afin de distinguer deux positions physiquement différentes mais similaires pour le système perceptif. Ainsi deux lieux, dont la position relative mesurée par les données proprioceptives est non nulle, ne seront pas confondus. Cette solution est celle qui est mise en œuvre dans la majorité des systèmes de navigation, car elle permet d'utiliser les deux sources d'informations en limitant les défauts inhérents à chacune. Ainsi la dégradation progressive des informations proprioceptives est compensée par la reconnaissance de positions de l'environnement grâce aux perceptions. Inversement, le problème de perceptual aliasing est réglé par l'utilisation des données proprioceptives.

Comme nous le verrons dans ce cours, il existe de nombreuses méthodes pour utiliser conjointement les deux sources d'informations. Ces méthodes diffèrent par leur capacité à utiliser de manière plus ou moins efficace les avantages des deux types d'informations. D'une manière générale, la qualité d'un système de navigation dépend fortement de cette capacité.

Chapter 4

Matériels courants en robotique mobile

4.1 Les bases mobiles

Nous présentons rapidement les différents types de bases mobiles utilisées en robotique, en nous focalisant sur les plateformes mobiles terrestres pour le milieu intérieur. Ce cours ne portant pas sur les méthodes de commande, nous ne rentrerons pas en détails dans les modèles cinématiques ou dynamiques associés. Nous ne parlerons pas non plus des effecteurs permettant au robot d'agir sur son environnement, tels que les bras articulés.

4.1.1 Holonomie

En robotique, une plateforme est dite *holonome* lorsque que le nombre de degrés de libertés contrôlables est égal au nombre total de degrés de liberté.

Pour un robot se déplaçant sur un plan, il y a 3 degrés de liberté (deux translations et une rotation). A partir d'une position donnée, une plateforme holonome devra donc pouvoir se déplacer en avant, sur le coté et tourner sur elle-même. Cette capacité permet de contrôler très simplement le robot car tous les déplacements imaginables sont réalisables, ce qui simplifie le problème de planification de trajectoire.

De nombreuses plateformes simples ne sont pas holonomes. C'est par exemple le cas des voitures, ce qui oblige à manœuvrer pour réaliser certaines trajectoires. Par exemple, il est nécessaire de faire un créneau pour réaliser un déplacement latéral. Ces contraintes devront donc être prises en compte lors de la planification de trajectoires. Nous allons cependant voir quelques mécanismes permettant d'obtenir des plateformes holonomes, ou s'en approchant.

4.1.2 Les plates-formes différentielles

Une des configurations les plus utilisées pour les robots mobiles d'intérieur est la configuration différentielle qui comporte deux roues commandées indépendamment. Une ou plusieurs roues folles sont ajoutées à l'avant ou à l'arrière du robot pour assurer sa stabilité (Figure 4.1). Cette plate-forme est très simple à commander, puisqu'il suffit de spécifier les vitesses des deux roues,

et permet de plus au robot de tourner sur place. Cette possibilité permet de traiter dans certains cas le robot comme un robot holonome, ce qui va simplifier la planification de déplacement et la commande du robot.

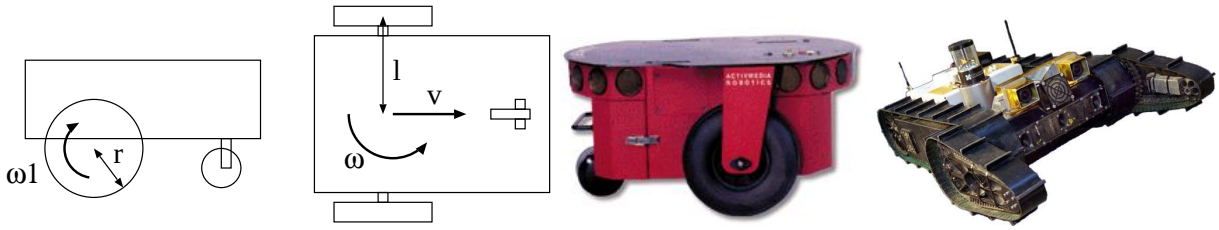


Figure 4.1: Exemple de plate-forme différentielle. Pioneer 2 DX de la société MobileRobots. Urban Robot de la société iRobot.

L'estimation du déplacement par odométrie est également très simple à partir de la mesure des vitesses de rotation des deux roues ω_1 et ω_2. Les vitesses de translation v et de rotation ω sont en effet données par:

$$\begin{aligned} v &= \frac{\omega_1 r + \omega_2 r}{2} \\ \omega &= \frac{\omega_1 r - \omega_2 r}{2l} \end{aligned}$$

Ce type de plate-forme peut également être utilisé avec des chenilles ce qui fournit une capacité de franchissement de petits obstacles intéressante (Figure 4.1). Ces plates-formes peuvent ainsi être utilisées en extérieur ou dans des décombres. L'utilisation de chenilles conduit cependant à une odométrie très bruitée à cause du contact mal défini entre les chenilles et le sol qui glissent beaucoup, notamment lors des rotations. L'estimation de la direction par l'odométrie sur ce type de plates-formes est donc en général rapidement inutilisable.

4.1.3 Les plates-formes omnidirectionnelles

Les plates-formes omnidirectionnelles permettent de découpler de manière plus nette le contrôle de la rotation et de la translation d'un robot et sont donc quasiment holonomes.

Il existe différents types de plateformes omnidirectionnelles. Le premier utilise trois ou quatre roues qui tournent à la même vitesse pour fournir une translation et un mécanisme qui permet d'orienter simultanément ces roues dans la direction du déplacement souhaitée (Figure 4.2). Le corps du robot lui-même n'effectue pas de rotation mais uniquement des translations. Ce système permet un contrôle très simple et relativement rapide car les changement de direction ne concernent que les roues et peuvent donc se faire très vite. Par contre ces plates-formes sont relativement limitées en capacité de franchissement et requièrent un sol très plan.

Une deuxième catégorie de plateformes utilise des roues dites "suédoises", qui n'offrent pas de résistance au déplacement latéral (Figure 4.3). La plateforme comporte trois roues dont les axes sont fixes. Les déplacements dans toutes les directions et en rotation sont obtenus en faisant varier individuellement les vitesses des roues. La plateforme tourne sur place lorsque

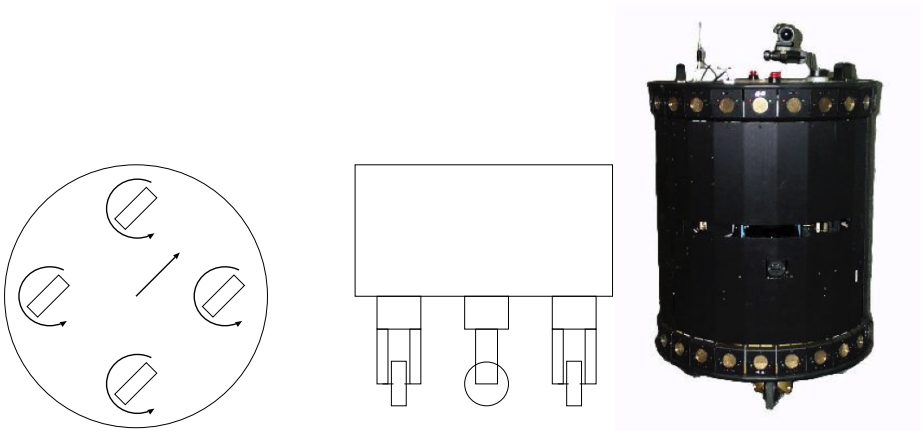


Figure 4.2: Exemple de plate-forme omnidirectionnelle à roues orientables.

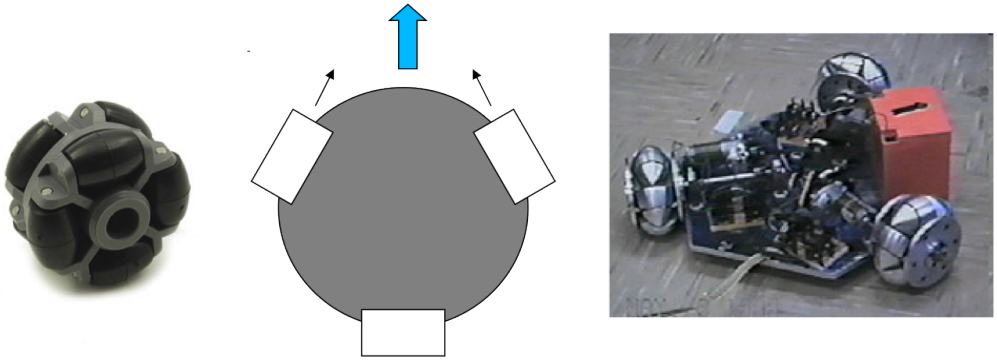


Figure 4.3: Exemple de plate-forme omnidirectionnelle à roues suédoises.

les trois roues tournent dans le même sens, à la même vitesse. Lorsque une roue est fixe, et que les deux autres tournent en sens opposé, la plateforme avance en direction de la roue fixe. Différentes combinaisons de vitesses permettent d'obtenir des déplacements quelconques.

4.1.4 Les plates-formes non holonomes

Des plates-formes non holonomes, telles que les voitures, sont également utilisées en robotique mobile (Figure 4.4). C'est plus particulièrement le cas dans le domaine des véhicules intelligents. Ces plates-formes sont toutefois plus difficile à commander car elle ne peuvent pas tourner sur place et doivent manœuvrer, ce qui peut être difficile dans des environnements encombrés. La commande de ces plates-formes pour réaliser un déplacement particulier est un problème à part entière que nous n'aborderons pas dans ce cours. Par contre, il est possible de prendre en compte ces contraintes de manière relativement simple dans la planification (voir chapitre III).

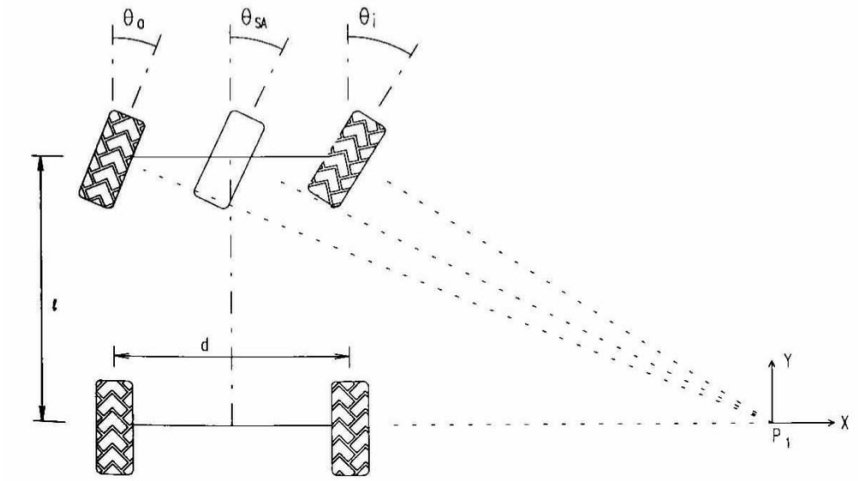


Figure 4.4: Exemple de plate-forme non holonome de type Ackerman.

Figure 4.5: Exemples de robots à pattes. Hexapode de AAI Canada, Aibo de Sony, Nao de Aldebarran Robotics.

4.1.5 Les plates-formes à pattes

Des plates-formes à deux, quatre ou six pattes peuvent également être utilisée. Elle ont l'avantage théorique de pouvoir se déplacer sur des terrains assez complexes, même si en pratique la plupart de ces plates-formes ne fonctionnent que sur des sols plans.

Les plates-formes à six pattes sont relativement pratiques car le robot peut être en équilibre permanent sur au moins 3 pattes, ce qui facilite le contrôle. Les plates-formes à deux ou quatre pattes sont plus complexes à commander et le simple contrôle de la stabilité et d'une allure de marche correcte reste aujourd'hui difficile, ce qui les rend en général relativement lentes. L'odométrie de ce type de plates-formes est de plus généralement d'assez faible qualité. Ces différents facteurs font que ces plates-formes sont rarement utilisées quand l'application visée a un besoin précis de positionnement et de navigation. De telles plates-formes commencent cependant à apparaître à relativement grande échelle (par exemple le robot Nao de Aldebarran Robotics).

4.2 Les capteurs

Nous présentons dans cette section les capteurs les plus couramment utilisés en robotique mobile pour les besoins de la navigation ainsi que des modèles probabilistes associés qui seront utilisés dans plusieurs méthodes de navigation.

4.2.1 Les capteurs proprioceptifs

Les capteurs proprioceptifs permettent une mesure du déplacement du robot. Ce sont les capteurs que l'on peut utiliser le plus directement pour la localisation, mais ils souffrent d'une dérive au cours du temps qui ne permet pas en général de les utiliser seuls.

Odométrie

L'odométrie permet d'estimer le déplacement de la plateforme à partir de la mesure de rotation des roues (ou du déplacement des pattes). La mesure de rotation est en général effectuée par un codeur optique disposé sur l'axe de la roue, ou sur le système de transmission (par exemple sur la sortie de la boite de vitesse pour une voiture). Le problème majeur de cette mesure est que l'estimation du déplacement fournie dépend très fortement de la qualité du contact entre la roue (ou la patte) et le sol. Elle peut être relativement correcte pour une plate-forme à deux roues motrices sur un sol plan de qualité uniforme, mais est en général quasiment inutilisable seule pour un robot à chenille par exemple. Pour limiter ce problème, il peut être intéressant de positionner le codeur optique sur une roue non motrice qui glissera moins. Notons cependant que l'erreur de ces méthodes se retrouve en général principalement sur l'estimation de la direction du robot, tandis que la mesure de la distance parcourue est souvent de meilleure qualité.

Modèle probabiliste

La majorité des modèles de localisation et de cartographie présentés dans ce cours (voir chapitre III) vont faire appel à un modèle probabiliste de cette mesure. Il existe deux types de modèles : les modèles directs (donnant la probabilité de la mesure en fonction du déplacement réel) et les modèles inverses (donnant la probabilité du déplacement réel en fonction de la mesure). Dans le cas de l'odométrie, la plupart des méthodes utilisent un modèle inverse afin d'interpréter les mesures réalisées.

Il existe divers types de modèles, mais les plus simples et les plus utilisés sont des modèles supposant que les paramètres du mouvement (direction θ et longueur d du déplacement, changement de direction ϕ du robot, cf Figure 4.6, gauche) sont statistiquement indépendants et soumis à un bruit Gaussien :

$$P(d,\theta,\phi|d_o,\theta_o,\phi_o) = e^{-\left(\frac{d-d_o}{\sigma_d}\right)^2} \times e^{-\left(\frac{\theta-\theta_o}{\sigma_\theta}\right)^2} \times e^{-\left(\frac{\phi-\phi_o}{\sigma_\phi}\right)^2}$$

où d,θ,ϕ sont les valeurs réelles et d_o,θ_o,ϕ_o les valeurs observées.

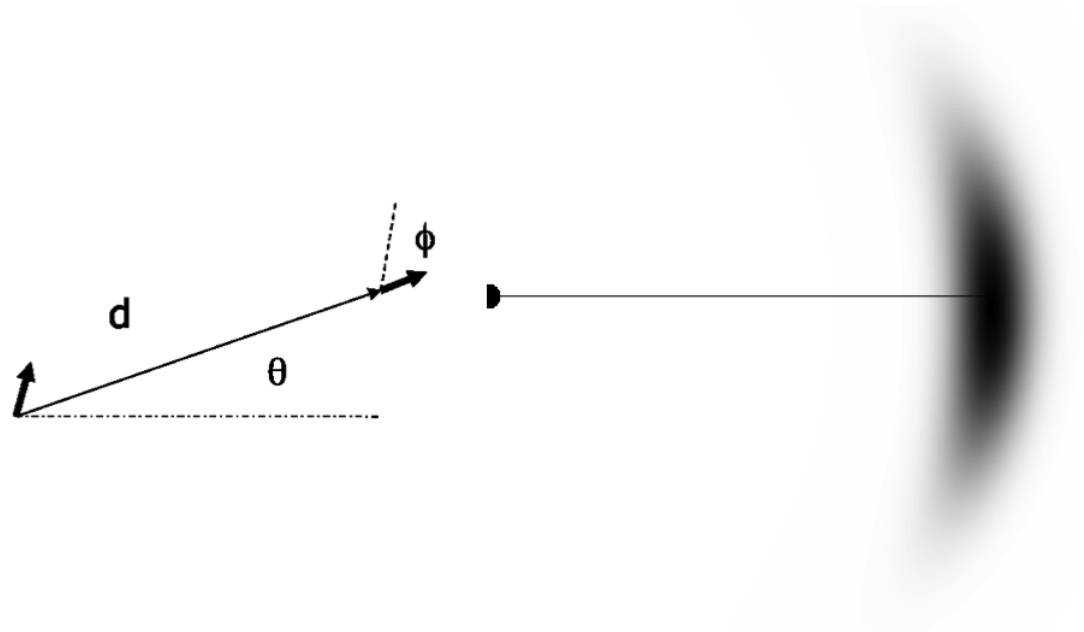


Figure 4.6: Modèle probabiliste de l'odométrie. Paramètres du déplacement à gauche. Exemple de densité de probabilité après un déplacement rectiligne à droite.

En général, les écarts types de ces différentes gaussiennes ($\sigma_d, \sigma_\theta, \sigma_\phi$) dépendent de la valeur de la mesure : l'erreur sur la longueur du déplacement pourra par exemple être proportionnelle à cette longueur :

$$\sigma_d = \gamma \times d$$

Il est possible d'utiliser des modèles beaucoup plus fins de l'odométrie reposant sur le processus physique utilisé pour la mesure du déplacement. Il est par exemple possible de faire une hypothèse de bruit gaussien sur le capteur réalisant la mesure de rotation de chaque roue puis, par calcul, d'en déduire l'erreur sur l'estimation du déplacement du robot. Cependant, une telle précision n'est souvent pas nécessaire dans de nombreux algorithmes.

Comme nous le verrons au chapitre sur la localisation, ces modèles probabilistes peuvent être utilisés pour générer des positions possibles du robot selon la distribution de probabilité déduite de la mesure de l'odométrie.

Les systèmes radar doppler et optiques

Au lieu de mesurer le déplacement par des mesures sur les roues, il est possible d'utiliser un radar pointé vers le sol qui permet de mesurer la vitesse du véhicule par effet Doppler. Il existe aussi des systèmes optiques, basés sur le même principe que les souris d'ordinateur, qui mesurent le déplacement du véhicule en analysant le mouvement relatif du sol (figure 4.7). Ces systèmes présentent l'avantage d'être plus précis que la mesure passant par les roues, notamment car ils sont indépendants des dérapages possible de ces roues. Il sont cependant en général relativement chers et encombrants et sont assez rares sur les petites plates-formes.

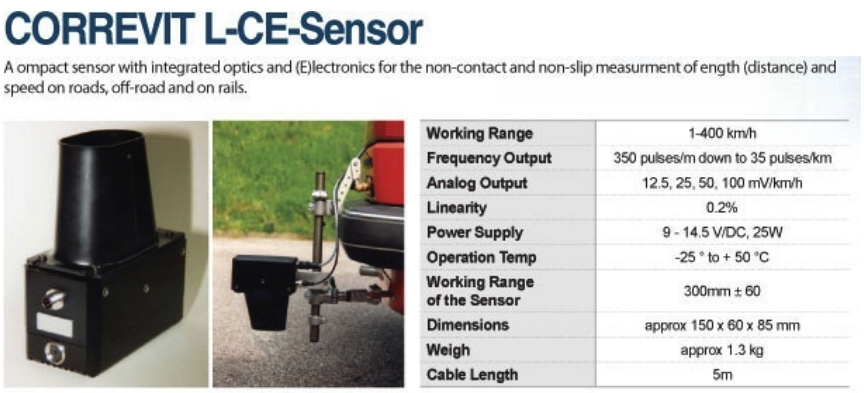


Figure 4.7: Exemple d'odomètre optique Correvit L-CE de CORRSYS- DARTON Sensorsysteme GmbH.

Les systèmes inertiels

La mesure de déplacement potentiellement la plus fiable provient de la mesure des accélérations de la plate-forme par des capteurs inertiels. Cette mesure est potentiellement fiable car elle ne dépend pas de la nature locale de l'environnement, cependant les capteurs inertiels sont tous entachés de bruit de mesure qui produit une dérive de l'estimation de la position au cours du temps.

La qualité des mesures inertielles dépend très fortement du type de capteurs utilisées. Historiquement, les premiers capteurs ont été réalisés à base de systèmes mécaniques et peuvent fournir des mesures extrêmement précise, au prix d'un coût et d'une masse très élevés. Ces dernières années ont vu apparaître de nouvelles technologies de capteurs, notamment basés sur les techniques de micro-électronique, qui ont permis la réalisation de capteurs inertiels "bas coût" et l'apparition de ces capteurs dans des produits grand public. La précision de ces capteurs est toutefois de quelques ordres de grandeur plus faible, ce qui rend leur utilisation isolée quasiment impossible. Ces capteurs fournissent toutefois un très bon complément à l'odométrie, notamment pour l'estimation de la direction.

L'accélération en translation de la plate-forme est mesurée par des accéléromètres. On dispose en général deux accéléromètres pour prendre des mesures dans deux directions perpendiculaires du plan de déplacement du robot. Un troisième peut être disposé verticalement afin de mesurer l'accélération en trois dimensions.

L'accélération angulaire est mesurée par des gyromètres. On dispose en général un gyromètre selon l'axe vertical, qui permet ainsi de mesurer l'angle de lacet du robot. Deux autres gyromètres peuvent être positionnés selon deux axes du plan de déplacement afin d'estimer la direction en trois dimensions.

Il est également possible de mesurer la rotation du robot par rapport à un axe de référence en utilisant un gyroscope. Cette mesure s'effectue en général par rapport à un axe de référence mis en rotation et isolé mécaniquement le plus possible du robot, ce qui rend sa direction indépendante de la direction du robot. Cette mesure peut être moins bruitée que l'intégration du signal d'accélération mais dépend très fortement de la qualité de la réalisation mécanique du système,

qui dépend très directement du prix du gyroscope.

Enfin, les magnétomètres permettent, par la mesure du champ magnétique terrestre, de déduire la direction du nord. Ces capteurs peuvent utiliser différentes technologies et ont l'avantage de fournir une direction de référence stable au cours du temps (au contraire des gyroscopes qui dérivent). Ces capteurs sont toutefois très délicats à utiliser en intérieur car ils sont très sensibles aux masses métalliques présentes dans les bâtiments et leur structure. En pratique, on les utilise donc principalement en extérieur en apportant le plus grand soin à leur positionnement sur le robot pour éviter les influences des composants du robot, notamment les moteurs électriques.

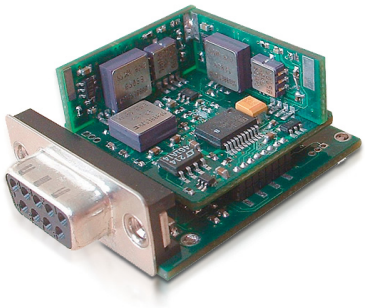


Figure 4.8: Centrale intertielle Crista de Cloud Cap Technology.

L'ensemble de ces éléments (accéléromètres, gyromètres, magnétomètres) peut être réuni pour former une centrale inertielle qui permet d'estimer complètement les six degrés de libertés de la position dans un espace à 3 dimensions. Les centrales inertielles "bas coût" sont cependant aujourd'hui de qualité insuffisante pour une utilisation isolée, tandis que les centrales de qualité correcte restent très chères. Ce domaine est cependant en évolution rapide avec l'arrivée de nouvelles technologies et l'apparition de centrales "bas coût" de bonne qualité devrait se faire dans les prochaines années.

L'utilisation des données fournies par ce type de senseurs passe aussi en général par un modèle probabiliste, qui peut être du type de celui présenté pour l'odométrie. Cependant, la gestion du bruit interne de ces capteurs demande en général des modèles beaucoup plus précis, qui estiment explicitement la dérive des capteurs afin de la corriger. Ceci permet de bénéficier de modèles plus précis en sortie également.

4.2.2 Les télémètres

Il existe différents types de télémètres, qui permettent de mesurer la distance aux éléments de l'environnement, utilisant divers principes physiques.

Télémètres à ultrason

Les télémètres à ultrason sont historiquement les premiers à avoir été utilisés. Il utilisent la mesure du temps de vol d'une onde sonore réfléchie par les obstacles pour estimer la distance

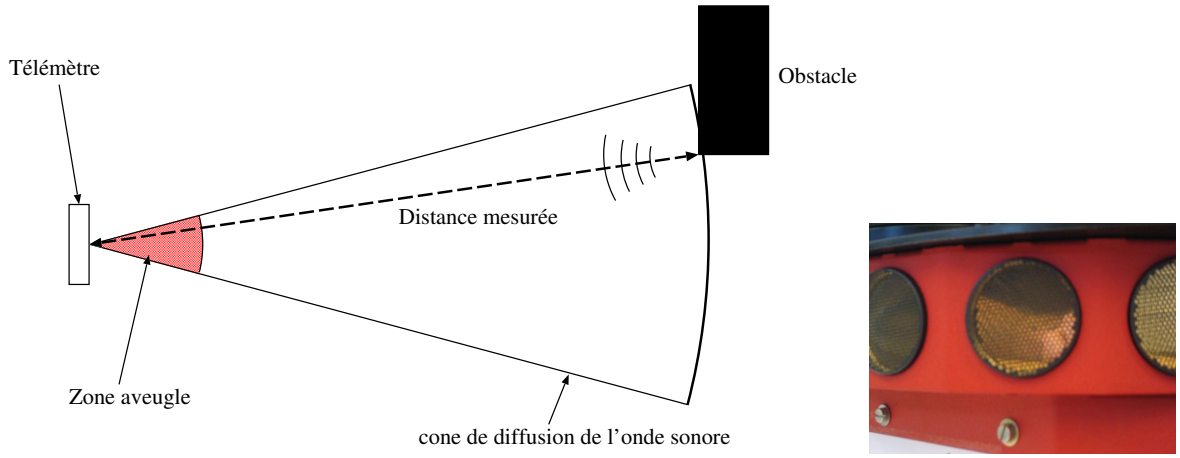


Figure 4.9: Principe du télémètre à ultrasons et exemple de télémètre réel.

(Figure 4.9). Ces télémètres sont très simple et peu cher, et sont donc très répandus, mais possèdent de nombreux inconvénients.

En premier lieu, deux télémètres voisins ne peuvent être utilisés simultanément, car il est impossible de savoir par lequel des deux télémètres une onde réfléchie a été émise (phénomène de "crosstalk"). Un robot possédant plusieurs télémètres doit donc les activer l'un après l'autre, ce qui entraîne un taux de rafraîchissement global des mesures relativement faible.

Ces télémètres possèdent une "zone aveugle", de quelques centimètres, en dessous de laquelle ils ne peuvent détecter les obstacles. Cette zone est due a une temporisation entre l'émission de l'onde sonore et le début de la détection de l'onde réfléchie qui est nécessaire pour ne pas perturber cette mesure.

De plus, l'onde réfléchie est très sensible aux conditions environnementales locales. Ainsi, si l'angle entre l'obstacle et la direction de l'onde sonore est trop faible, il n'y aura pas de retour de l'onde sonore et l'obstacle ne sera pas perçu. L'onde de retour dépend également de la texture de l'obstacle. Un mur couvert de moquette pourra par exemple ne pas être détecté.

Les télémètres ultrason détectent les obstacles se situant dans un cône relativement large (d'angle au sommet d'environ 30 degrés). Cette caractéristique peut être à la fois un avantage et un inconvénient. C'est un inconvénient car un obstacle détecté n'est pas localisé en angle à l'intérieur du cône de détection, et on obtient donc une mesure de la position relativement imprécise. C'est par contre un avantage car des éléments relativement fins (les pieds de table ou de chaise par exemple) sont détectés dans ce cône, alors qu'il pourraient ne pas être détectés par des télémètres ayant un angle d'ouverture très fin.

Télémètres à infrarouge

Les télémètres infrarouges possèdent l'avantage d'avoir un cône de détection beaucoup plus restreint. Il utilisent une lumière infrarouge au lieu d'une onde sonore pour la détection et peuvent être basés sur différentes techniques qui permettent de recueillir plus ou moins d'information.

Il est possible de mesurer simplement le retour ou le non-retour d'une impulsion codée, ce

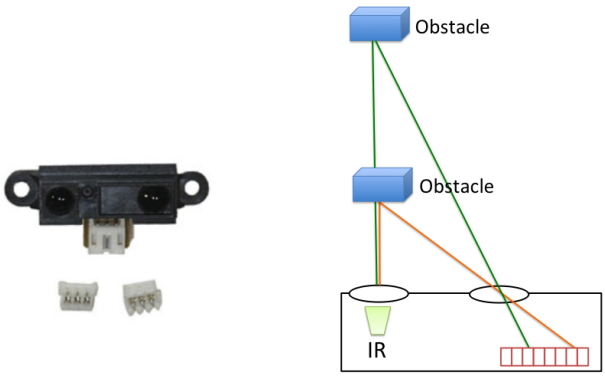


Figure 4.10: Principe du télémètre infrarouge à triangulation et exemple de télémètre réel (Sharp).

qui permet de détecter la présence ou l'absence d'un obstacle dans une certaine portion de l'espace. Il est également possible de réaliser une triangulation sur le faisceau de retour de l'onde lumineuse, ce qui permet d'avoir une mesure de la distance de l'obstacle (figure 4.10).

Les inconvénients de ces télémètres sont liés à leur portée, en général relativement restreinte, et à leur sensibilité aux sources de lumières qui contiennent un fort rayonnement infrarouge. Un projecteur du type de ceux utilisés pour la télévision pointé sur le robot, par exemple, sature en général complètement le récepteur et empêche toute détection d'obstacle. Ils sont également très sensibles à la couleur et à la nature de la surface de l'obstacle (par exemple, ils détectent difficilement les vitres et les obstacles noir mats).

Télémètres laser

Les télémètres les plus utilisés à l'heure actuelle pour des applications de cartographie et de localisation sont les télémètres laser à balayage. Ils utilisent un faisceau laser mis en rotation afin de balayer un plan, en général horizontal, et qui permet de mesurer la distance des objets qui coupent ce plan (Figure 4.11, 4.11). Cette mesure peut être réalisée selon différentes techniques soit en mesurant le temps de vol d'une impulsion laser, soit par triangulation.

Les télémètres courants ont une bonne résolution angulaire car ils permettent d'obtenir une mesure de distance tout les demi degrés, sur une zone de 180 ou 360 degrés selon les modèles. La mesure est de plus relativement précise (avec un bruit de l'ordre de quelques centimètres) à une distance relativement grande (plusieurs dizaines de mètres). La fréquence d'acquisition est en général de l'ordre de la dizaine de Hertz, voire proche de la centaine pour certains modèles.

Ces télémètres sont très utilisés en environnement intérieur car il fournissent des données abondantes et précises sur la position des objets caractéristiques de l'environnement tels que les murs. Ils possèdent toutefois un certain nombre d'inconvénients. En premier lieu, leur zone de perception est restreinte à un plan et ne permet donc pas de détecter les obstacles situés hors de ce plan (un petit objet posé au sol par exemple). Ils ne peuvent pas non plus détecter les objets ne réfléchissant pas correctement la lumière du laser (en premier lieu les vitres, mais aussi certains objets très réfléchissants, tels que les objets chromés). Pour limiter ces inconvénients,

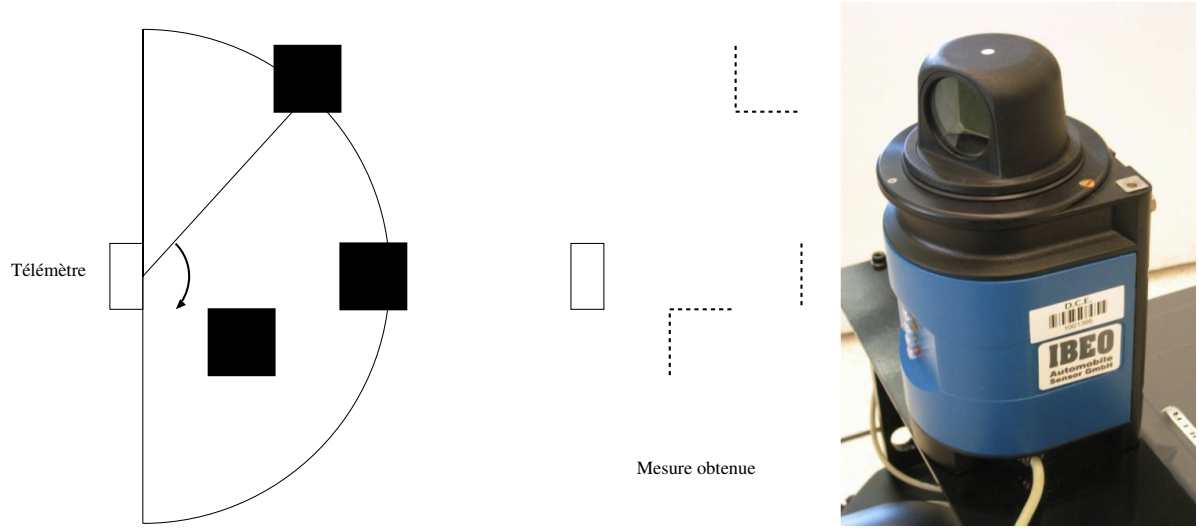


Figure 4.11: Illustration du principe de fonctionnement d'un télémètre Laser et un exemple de Télémètre Laser à balayage, fournissant 720 mesure réparties sur 360 degrés, à 5 Hz (marque Ibeo).

il est possible de les utiliser en conjonction avec des capteurs à ultrason qui ont un cône de détection plus large et qui peuvent détecter les vitres.

Enfin, la plupart des algorithmes de cartographie et de localisation existants supposent que le plan de mesure du télémètre laser reste horizontal et à hauteur constante, ce qui n'est plus vrai en cas de sol irrégulier ou, dans la majorité des cas, en extérieur. Il est alors nécessaire de passer à une localisation et une cartographie en 3D.

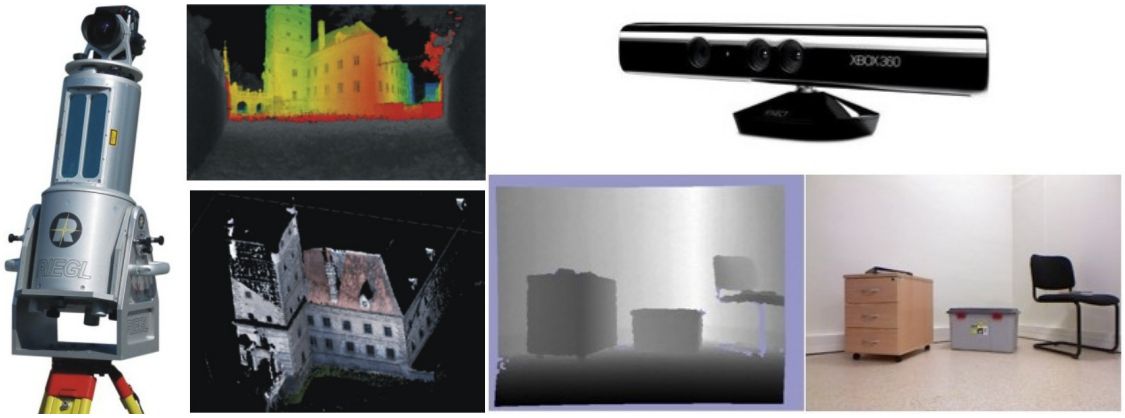


Figure 4.12: Un exemple de télémètre laser à balayage selon 2 axes (à gauche) et de camera permettant d'obtenir une image de profondeur de 320x240 pixels (à droite).

Il existe des télémètres laser balayant l'espace selon deux axes 4.12. Ils permettent ainsi d'obtenir une image de distance selon un angle solide de l'ordre de quelques dizaines de degrés dans les deux dimensions. Ces télémètres restent toutefois cher et fragile du fait de la mécanique

nécessaire au balayage. De plus, la fréquence d'acquisition est relativement faible (de l'ordre de quelques Hertz), ce qui pose problème lorsque le robot est en mouvement. Ces télémètres sont donc plutôt destinés à des applications relativement statiques comme la création de modèles 3D.

Il existe également des systèmes sans balayage permettant d'obtenir une image de profondeur de la même manière qu'une caméra couleur standard. Plusieurs technologies sont utilisées, soit par mesure de temps de vol d'une impulsion laser ou infrarouge, soit par triangulation à partir de projecteurs infrarouges. Ces capteurs sont récents, mais sont très intéressants pour la robotique mobile car ils permettent d'obtenir une information dense à une fréquence assez élevée (image de profondeur de 320x240 à 30 Hz pour la caméra Kinect par exemple, figure 4.12). Ces informations peuvent être couplées à une image couleur, on parle alors de caméra RGBD (D pour Depth). Il reste cependant certaines limitations, notamment pour l'emploi en extérieur où l'information de profondeur peut être perdue à cause de la lumière du soleil qui masque la lumière infrarouge.

Figure 4.13: Un exemple de télémètre laser à balayage a 64 nappes conçu par Velodyne.

Enfin, principalement pour les véhicules intelligents, il existe un compromis qui consiste à utiliser plusieurs nappes laser avec différentes inclinaisons afin d'avoir des modèles assez précis de l'environnement sur 360 degrés (figure 4.13). Ces capteurs restent assez lourds et chers, mais permettent de réaliser quasiment l'ensemble des tâches nécessaires pour un véhicule comme la localisation, la cartographie et la détection de piétons ou de véhicules.

Modèle probabiliste

Les modèles probabilistes associés aux télémètres permettent de donner la probabilité de la mesure en fonction de la distance réelle de l'obstacle. Pour les capteurs réalisant plusieurs mesures, les probabilités sont en général estimées pour chacune des mesures individuelles prises depuis une position, puis agglomérées par produit en supposant les mesures indépendantes :

$$P(Scan|Obstacles) = \prod_{i=1}^{M} P(mesure_i|Obstacles)$$

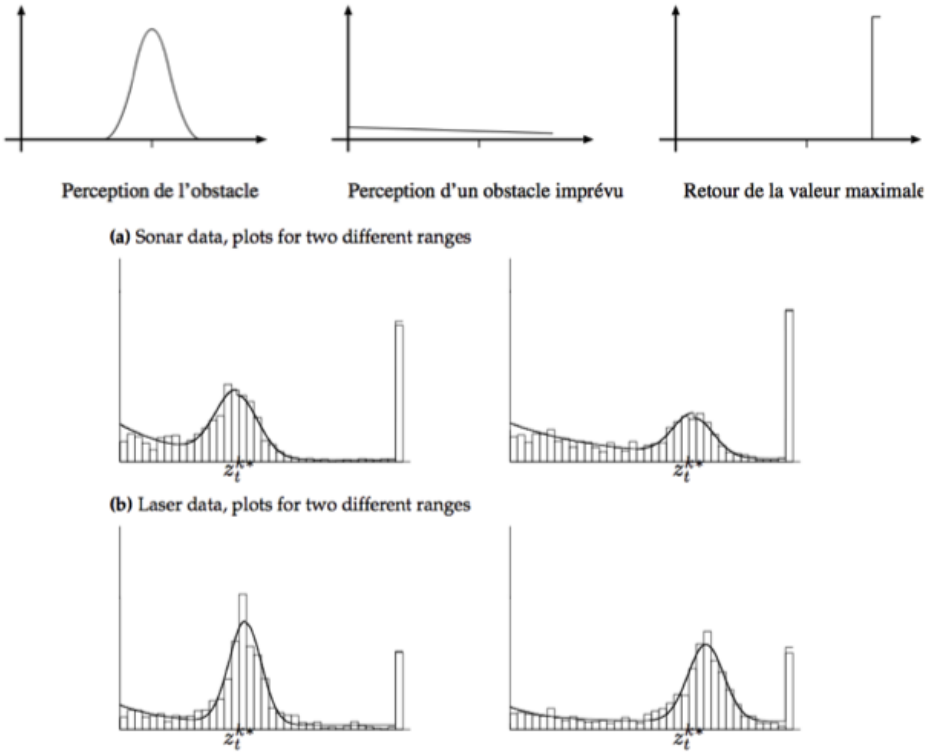


Figure 4.14: Modèle probabiliste de télémètre par composition d'évènements élémentaires et exemple d'application sur des données réelles de sonars ou de télémètres laser (tiré de [135]).

Pour estimer la probabilité d'une mesure individuelle, il est possible d'utiliser une simple loi gaussienne autour de la distance réelle comme modèle probabiliste, mais les modèles sont en général un peu plus évolués et utilisent une combinaison de lois qui modélisent plusieurs phénomènes qui peuvent être responsables de la mesure (Figure 4.14) :

- la mesure effective de l'obstacle visé, modélisé par une gaussienne en général
- la perception d'un obstacle imprévu, par exemple une personne ou un objet dynamique non présent dans la représentation du monde, modélisé par une loi décroissante telle qu'une exponentielle
- la non détection d'un écho, qui donne une mesure à la distance maximale du télémètre, modélisé par un pic.

Les paramètres de cette combinaison de lois peuvent être réglés manuellement ou estimés à partir d'un ensemble de mesures, par exemple en utilisant un algorithme de maximisation de l'espérance. Ces modèles peuvent être adaptés à tout les types de télémètres (figure 4.14).

4.2.3 Les caméras

L'utilisation d'une caméra pour percevoir l'environnement est une méthode attractive car elle semble proche des méthodes utilisées par les humains et fournit un grande quantité d'information sur l'environnement. Le traitement des données volumineuses et complexes fournies par ces capteurs est cependant souvent difficile, mais c'est une voie de recherche très explorée et prometteuse pour la robotique.

Caméras simples

Une caméra standard peut être utilisée de différentes manières pour la navigation d'un robot mobile. Elle peut être utilisée pour détecter des amers visuels (des points particuliers qui servent de repère, tels que des portes ou des affiches) à partir desquels il sera possible de calculer la position du robot. Si ces amers sont simplement ponctuels, ou de petite taille, il sera en général simplement possible d'estimer leur direction. Dans le cas ou les amers sont des objets connus en 2 ou 3 dimensions, il sera en général possible d'estimer complètement la position du robot par rapport à la leur. Elle peut également être utilisée pour détecter des "guides" de navigation pour le robot, tels que des routes ou des couloirs.

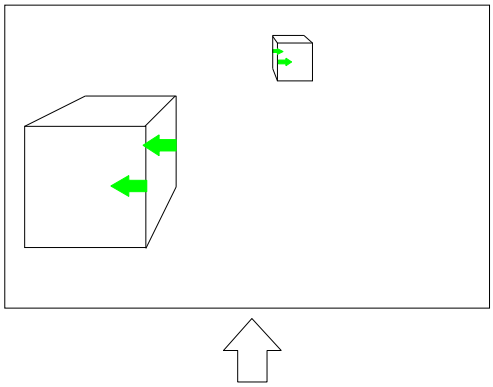


Figure 4.15: Illustration du principe de base du flot optique.

Il est également possible d'utiliser globalement une image pour caractériser une position ou un point de vue dans l'environnement. Il faudra alors comparer cette image aux nouvelles images acquises par le robot pour savoir si le robot est revenu à cette position. Cette comparaison peut faire appel à différentes techniques, notamment celles utilisées dans le domaine de l'indexation d'image.

Lorsque le robot est en mouvement, il est également possible de tirer parti du *flot optique* (le mouvement apparent des objets dans l'image, voir figure 4.15), afin d'avoir une estimation de la distance des objets. En effet, les objets les plus proches ont un déplacement apparent plus important que les objets lointains. Cette méthode permet notamment de réaliser un évitement d'obstacles ou de réaliser une reconstruction tridimensionnelle de l'environnement (par des techniques connues sous le nom de *structure from motion*, voir section 4.2.3).

Caméras stéréoscopiques

Figure 4.16: Exemple de données fournies par des caméras stéréoscopiques.

Lorsque l'on dispose de deux caméras observant la même partie de l'environnement à partir de deux points de vue différents, il est possible d'estimer la distance des objets et d'avoir ainsi une image de profondeur (Figure 4.16), qui peut être utilisée pour l'évitement d'obstacles ou la cartographie. Cette méthode suppose toutefois un minimum d'éléments saillants dans l'environnement (ou un minimum de texture) et peut être limitée, par exemple dans un environnement dont les murs sont peint de couleurs uniformes. La qualité de la reconstruction risque également de dépendre fortement des conditions de luminosité. La résolution et l'écartement des deux caméras impose également les profondeurs minimum et maximum qui peuvent être perçues, ce qui peut être limitatif pour la vitesse de déplacement du robot.

Des techniques similaires peuvent également être utilisées pour estimer la profondeur à partir d'une caméra en mouvement (méthodes de *structure from motion*, voir par exemple [67]), la difficulté étant alors d'estimer à la fois la profondeur et les positions relatives de la caméra lors de la prise des deux images.

Caméras panoramiques

Les caméras panoramiques (catadioptriques) sont constituées d'une caméra standard pointant vers un miroir de révolution (par exemple un simple cône, ou un profil plus complexe qui peut s'adapter à la résolution exacte que l'on veut obtenir sur le panorama) (figure 4.17). L'image recueillie permet d'avoir une vision de l'environnement sur 360 degrés autour de la camera. Le secteur angulaire vertical observé dépend de la forme du miroir et peut être adapté aux besoins de chaque application (Figure 4.17).

Ce type de caméra est très pratique pour la navigation car une image prise par une camera panoramique orientée verticalement permet de caractériser une position, indépendamment de la direction du robot. En effet, pour une position donnée et pour deux orientations différentes, la même image sera formée par la caméra, à une rotation autour du centre près, tandis que pour une caméra standard, orientée horizontalement, la scène serait différente.

Ces caméras sont donc très pratiques lorsque l'on caractérise une position de manière globale, mais peuvent aussi être utilisées pour détecter des amers ou pour estimer le flux optique. Dans ce cas, toutefois, comme la géométrie de l'image formée est relativement complexe et

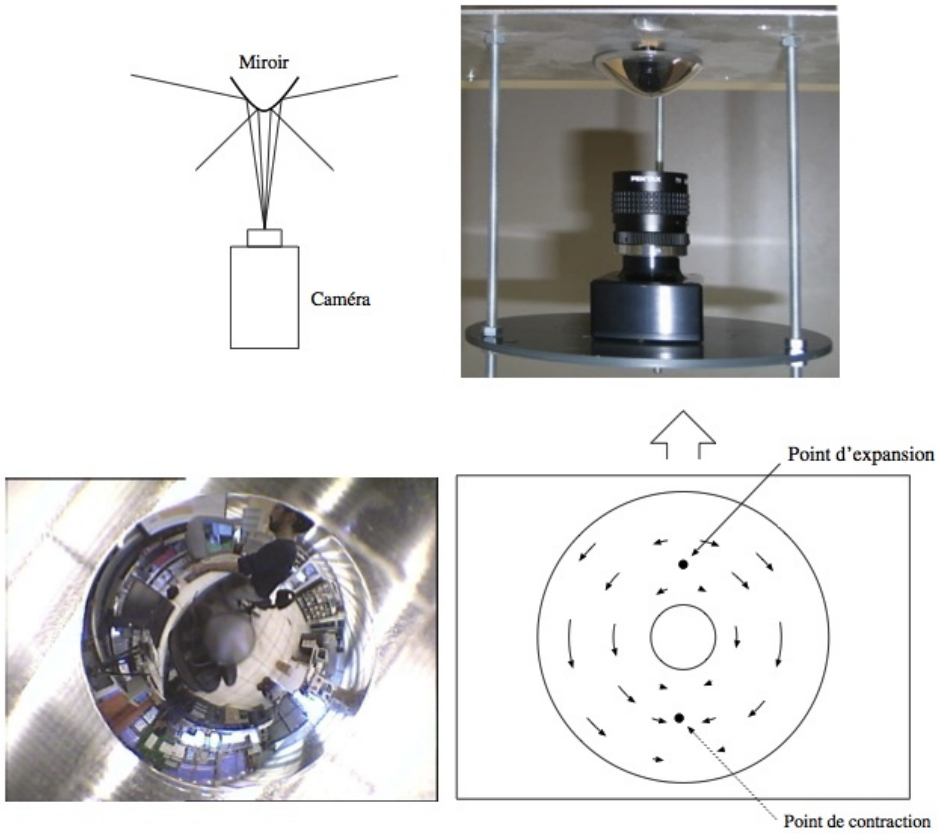


Figure 4.17: Principe des caméras panoramiques catadioptriques, exemple d'image obtenue et illustration du flux optique.

comme la résolution obtenue varie énormément selon la direction observée, les algorithmes doivent être adaptés, ce qui pose un certain nombre de problèmes.

Concernant le flux optique, cependant, les caméras panoramiques possèdent l'avantage de contenir toujours le point d'expansion et le point de contraction dans l'image, ce qui rend l'estimation du mouvement beaucoup plus aisée (figure 4.17).

4.2.4 Autres capteurs

Les capteurs tactiles

Les robots peuvent être équipés de capteurs tactiles, qui sont le plus souvent utilisés pour des arrêts d'urgence lorsqu'il rencontre un obstacle qui n'avait pas été détecté par le reste du système de perception.

Ces capteurs peuvent être de simples contacteurs répartis sur le pourtour du robot. Il ne détectent alors le contact qu'au dernier moment. Il est également possible d'utiliser des petites tiges arquées autour du robot pour servir d'intermédiaire à ces contacteurs, ce qui permet une

détection un peu plus précoce et donne ainsi plus de marge pour arrêter le robot.

Les balises

Dans certaines applications, il est également possible d'utiliser des balises dont on connaît la position, et qui pourront être facilement détectées par le robot, afin de faciliter sa localisation.

Des techniques très diverses peuvent être utilisées pour ces balises. On peut par exemple utiliser un signal radio, émis de manière omnidirectionnel par la balise. Le robot sera alors équipé d'une antenne directionnelle qui lui permettra de détecter la direction des différentes balises, afin de déduire sa position par triangulation.

On peut également utiliser des codes couleurs ou des codes barres qui pourront être détectés par une caméra.

Le GPS

Les besoins de localisation étant omniprésents dans de très nombreux secteurs de la vie actuelle, l'idée d'avoir un système de localisation le plus universel possible à donné lieu à l'apparition du Global Positionning System (GPS). C'est un système de balises dont on a placé les balises sur des satellites en orbite terrestre et qui est par conséquent accessible de quasiment partout à la surface du globe. Ce système permet donc d'avoir une mesure de sa position dans un repère global couvrant la terre avec une précision variant de quelques dizaines de mètres à quelques centimètres suivant les équipements.

Ce système est cependant loin de résoudre tous les problèmes de localisation des robots mobiles. Il fonctionne en effet difficilement dans des environnements urbains, et n'est pas utilisable à l'intérieur des bâtiments. Sa précision est de plus souvent trop faible pour qu'un robot terrestre puissent utiliser ces informations seules. En pratique, il est souvent couplé à un système inertiel qui permet de palier aux pertes du signal GPS et il ne remplace de toute façon pas les capteurs du robot qui lui permettent de percevoir son environnement immédiat, qui constitue la source d'information principale pour la navigation à court terme (par exemple l'évitement d'obstacles, par opposition à la navigation à long terme qui consiste à rejoindre un but distant).

4.3 Pour aller plus loin

Sensors for Mobile Robots: Theory and Application, Everett

Une version en ligne est disponible :

http://www-personal.engin.umich.edu/~johannb/my_book.htm

Part II

Navigation réactive

Dans cette partie, nous présentons différentes stratégies de navigation réactive. Ces stratégies peuvent être utilisées dans des architectures de contrôle purement réactives, mais aussi comme modules de bas-niveau dans une architecture hybride. Par définition, les stratégies de navigation réactives n'utilisent que les valeurs courantes des capteurs (ou des valeurs sur une petite fenêtre temporelle), et non des données provenant d'un modèle interne, pour décider de l'action à effectuer.

Chapter 5

Navigation vers un but

Nous commençons ici par des méthodes de navigation correspondant aux deux premières catégories de stratégies de navigation définies dans le chapitre 2, c'est à dire l'approche d'un but défini par un objet ou une configuration d'amers.

5.1 Véhicules de Braitenberg

Dans son livre "Vehicles: Experiments in Synthetic Psychology", Valentino Braitenberg [19] décrit une série d'expériences dans lesquelles des robots extrêmement simples peuvent montrer des comportements complexes, qu'un observateur humain associe en général à différents types d'émotions telles que la peur ou l'agression. Nous nous intéressons ici simplement à la structure de ces robots, qui permet de réaliser simplement des comportements pour rejoindre un but visible. Cette structure est devenue l'archétype des méthodes réactives simples.

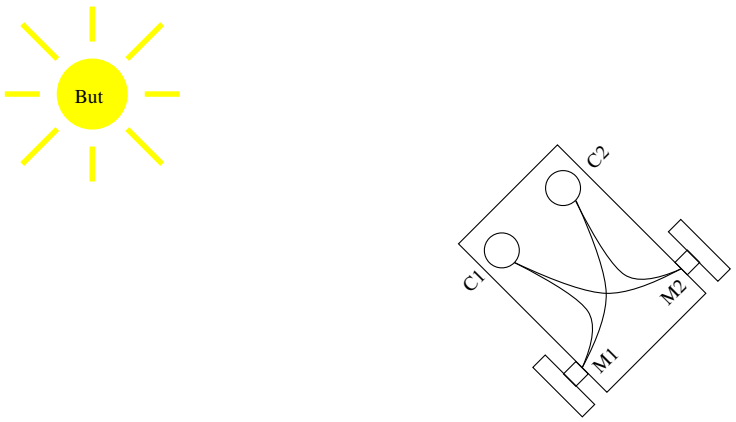


Figure 5.1: Dans les véhicules de Braitenberg, la vitesse de chacun des deux moteurs du robot dépend de la valeurs de deux capteurs qui détectent la lumière émise par le but.

Dans le livre de Braitenberg, le but est matérialisé par une lumière, visible depuis tout l'environnement. Le robot est simplement une plate-forme différentielle, constituée de deux roues dont on com-

mande les vitesses de rotation et munie de deux capteurs de lumière situés de part et d'autre de l'avant du robot (Figure 5.1). L'architecture interne du robot est simplement constituée de liens entre ces capteurs et les moteurs qui permettent de calculer la vitesse des moteurs en fonction des valeurs des capteurs.

En faisant varier les paramètres des connexions, il est alors possible de définir différents comportements du robot. Si la vitesse de chaque moteur est reliée à la valeur du capteur du coté opposé avec un coefficient positif, le robot se dirigera naturellement vers le but. Si, par contre, la vitesse de chaque moteur est reliée à la valeur du capteur du même coté avec un coefficient positif, le robot fuira le but.

Ces véhicules réalisent simplement une remontée ou une descente de gradient sur l'intensité de la lumière. Ils correspondent à un simple contrôleur proportionnel en automatique et sont donc relativement sujets à des oscillations dans le comportement du robot. Ils supposent de plus que le but est visible depuis tout l'environnement, ce qui est rarement le cas en pratique. Ce modèle est donc intéressant car c'est la méthode la plus simple possible pour réaliser un déplacement vers un but, mais est difficile à utiliser dans une application réelle.

5.2 Modèle de Cartwright et Collet

Le "snapshot model" a été conçu pour expliquer comment des abeilles peuvent utiliser des informations visuelles pour rejoindre un point donné de l'environnement. Il permet à un robot de rejoindre un but dont la position est définie par la configuration d'amers de l'environnement autour de ce but.

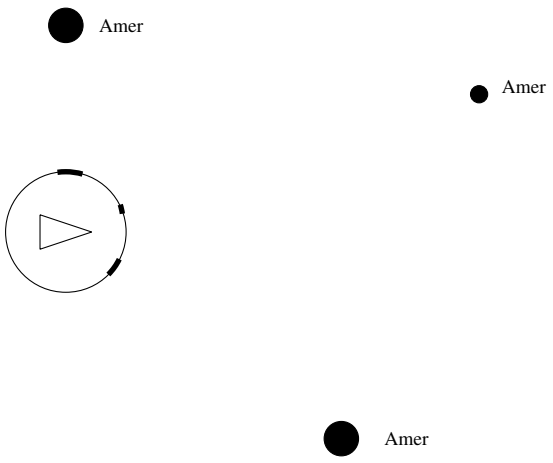


Figure 5.2: Exemple de snapshot *caractérisant la position du but. Le robot mémorise un panorama contenant la position et la taille apparente des amers.*

Le système perceptif du robot doit lui permettre de détecter la direction et la taille des amers autour de lui. Le robot commence par mémoriser le but en enregistrant la configuration des amers vus depuis la position de ce but (un *snapshot*, Figure 5.2).

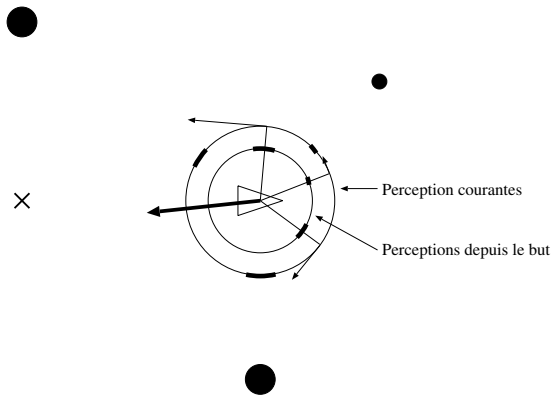


Figure 5.3: Pour atteindre le but, chaque amer perçu est associé à un des amers mémorisés. Pour chaque appariement, on déduit un vecteur tangentiel dont la norme augmente avec l'écart entre amer perçus et mémorisés. La somme de ces vecteurs donne la direction à prendre pour atteindre le but.

Lorsque, par la suite, le robot veut rejoindre ce but, il prend une nouvelle image des amers et, par comparaison entre la configuration courante et la configuration mémorisée au but, il peut déduire de manière très simple la direction dans laquelle se déplacer pour atteindre le but. Cette comparaison est basée sur un appariement entre les amers perçus et les amers mémorisés, chaque appariement permettant de calculer un vecteur dont la somme, pour tous les appariements d'amers, donne la direction à prendre pour rejoindre le but (Figure 5.3). Le robot effectue alors un déplacement de longueur fixée dans cette direction puis recommence le processus tant que le but n'est pas atteint.

Là encore, le système est très simple et réalise une descente de gradient sur la configuration des amers afin d'atteindre le but. Il ne fonctionne cependant pas sur l'ensemble de l'environnement et la qualité du comportement obtenu dépend beaucoup de la configuration des amers qui sont utilisés, un ensemble d'amers lointains et bien répartis tout autour du robot donnant les meilleurs résultats. La qualité de l'appariement entre les amers est également primordiale, en effet, si un amer perçu est associé au mauvais amer mémorisé, le vecteur de déplacement déduit sera faux. Le modèle original supposait des amers noirs sur fond blanc, sans identité particulière, pour lequel l'appariement est relativement hasardeux. Il n'est donc pas applicable en pratique. D'autres travaux ont utilisé des amers colorés et différentes contraintes sur l'appariement qui permettent une meilleur robustesse et sont donc applicables à des robots réels [58].

La plupart des implantations de ce modèles supposent de plus que la direction du robot est connue afin de faciliter l'appariement. Avoir une estimation correcte de cette direction peut se révéler difficile en pratique.

5.3 Asservissement visuel

L'asservissement visuel [30] (document disponible en ligne[1]) est une technique d'asservissement de la position d'un robot qui est basée directement sur l'information extraite d'une image, sans modélisation intermédiaire de l'environnement. Développées à l'origine pour la commande des robots manipulateurs, ces techniques permettent également la commande de robots mobiles.

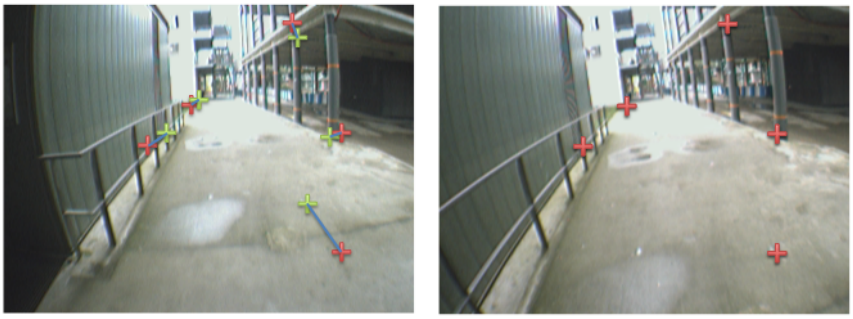


Figure 5.4: Illustration du principe de l'asservissement visuel: l'erreur entre une image courante et une image but (écart entre les croix rouges et vertes ici) est utilisée pour calculer une commande qui permettra au robot d'atteindre la position correspondant à l'image but.

Dans ces approches, le but à atteindre est spécifié par l'image que le robot devra percevoir depuis cette position. Différentes mesures sont réalisées sur cette image (par exemple la détection de points d'intérêts) et la commande du robot est conçue pour amener à 0 l'écart entre la mesure réalisée sur l'image courante et la mesure réalisée sur l'image cible (figure 5.4). Les choix de mesures dans l'image et de la loi de commande peuvent être très variés, et vont conditionner les trajectoires obtenues par le robot, leur stabilité, leur robustesse aux mauvaises perceptions ou aux mauvaises modélisations du système, etc...

Nous ne détaillerons pas ici ces approches, mais il existe plusieurs applications intéressantes en robotique mobile [15, 123, 34]. Notons que ces modèles sont souvent étendus pour fournir une navigation à long terme en enchainant des tâches de contrôle local sur des séquences d'images. Par exemple, [15] présente un système permettant de guider un robot en environnement intérieur à partir du suivi de motifs détectés sur le plafond par une caméra pointée à la verticale. En enchaînant des asservissements sur une séquence d'images, ce système permet au robot de refaire une trajectoire qui a été montrée au préalable par un opérateur. De même, [123] et [34] réalisent le guidage d'un véhicule en extérieur à l'aide d'une caméra pointée vers l'avant.

[1] http://www.irisa.fr/lagadic/pdf/2002_hermes_chaumette.pdf

Chapter 6

Évitement d'obstacles

L'évitement d'obstacles est un comportement de base présent dans quasiment tous les robots mobiles. Il est indispensable pour permettre au robot de fonctionner dans un environnement dynamique et pour gérer les écarts entre le modèle interne et le monde réel.

Les méthodes que nous présentons sont efficaces à condition d'avoir une perception correcte de l'environnement. Elles seront par exemple très efficaces avec un télémètre laser, mais donneront des résultats plus bruités avec des sonars. Pour limiter ce problème, il est possible d'appliquer ces méthodes sur une représentation locale (c'est-à-dire de l'environnement proche du robot et centrée sur le robot) de l'environnement qui sera construite en fonction des données de quelques instants précédents. Cette représentation intermédiaire permettra de filtrer une grande partie du bruit des données individuelles (en particulier pour les sonars).

Il faut également faire attention à ce que les capteurs détectent tous les obstacles. Par exemple un laser à balayage ne verra pas les objets au dessous ou au dessus de son plan de balayage, et pourra voir du mal à percevoir les vitres. Pour cette raison, on utilise souvent une nappe laser couplée à des sonars, ou un système de plusieurs nappes laser inclinées.

6.1 Méthode des champs de potentiel

Dans la méthode d'évitement d'obstacles par champs de potentiels, on assimile le robot à une particule se déplaçant suivant les lignes de courant d'un potentiel créé en fonction de l'environnement perçu par le robot. Ce potentiel traduit différents objectifs tels que l'évitement d'obstacles ou une direction de déplacement préférée. Il est calculé par sommation de différentes primitives de potentiels traduisant chacun de ces objectifs (Figure 6.1). Ces différents potentiels peuvent avoir une étendue spatiale limitée ou non (par exemple, n'avoir une influence que près des obstacles) et leur intensité peut dépendre ou non de la distance.

Le gradient de ce potentiel donne, en chaque point de l'espace, la direction de déplacement du robot (Figure 6.1). Comme c'est ce gradient, et non la valeur absolue du potentiel, qui nous intéresse, il est possible de calculer directement en chaque point sa valeur par une simple somme vectorielle en ajoutant les valeurs issues des différents potentiels primitifs. Ainsi, pour un robot se déplaçant en ligne droite en espace ouvert et évitant les obstacles qu'il peut rencontrer, nous

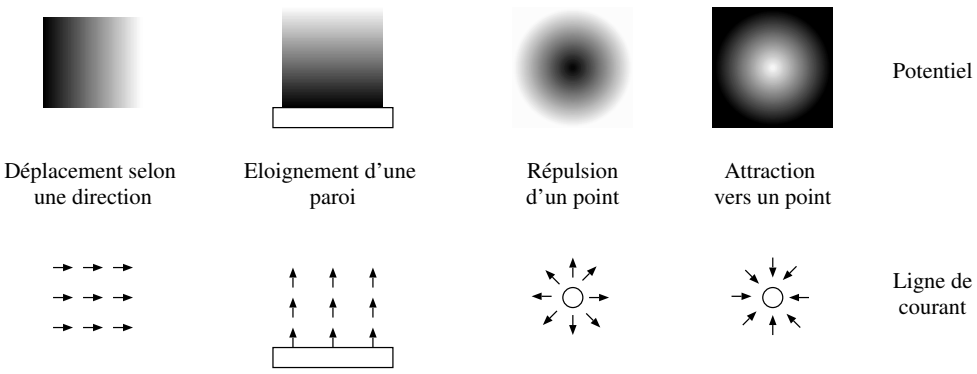


Figure 6.1: Illustration de potentiels primitifs dont la combinaison guide les déplacements du robot. Le robot se déplacera selon les lignes de courant.

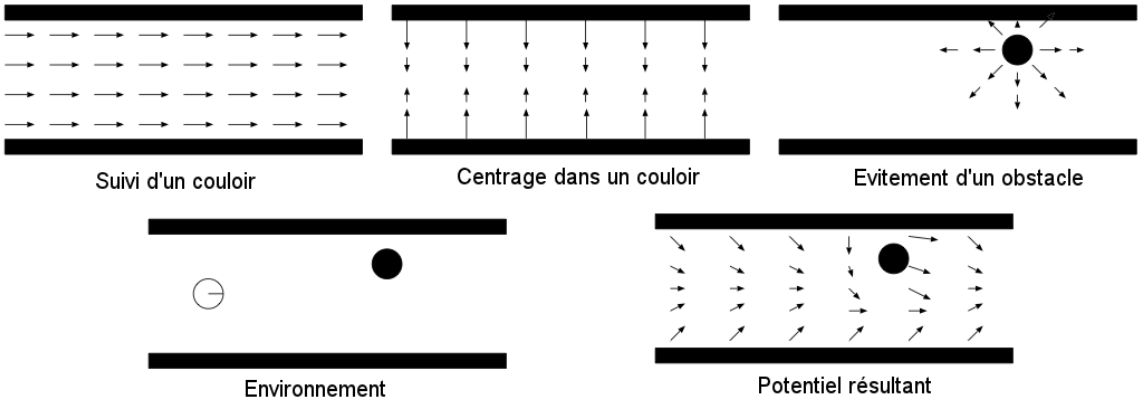


Figure 6.2: Illustration de la combinaison de différents potentiels primitifs.

obtenons par exemple les lignes de courant illustrées figure 6.2.

De plus, dans la pratique, pour l'évitement d'obstacles, le potentiel est en général calculé dans l'espace relatif au robot et ne sert qu'a décider de la vitesse et de la direction courante. Il n'est donc nécessaire de l'estimer que pour la position courante du robot, en sommant simplement la contribution des différents éléments perçus (Figure 6.3).

Le principal inconvénient de cette méthode d'évitement d'obstacles est l'existence, pour certaines configurations d'obstacles (relativement courantes) de minimum locaux du potentiel qui ne permettent pas de décider de la direction à prendre (Figure 6.3). Ce problème peut être traité de différentes façons. Il est par exemple possible de déclencher un comportement particulier lorsque l'on rencontre un tel minimum (déplacement aléatoire, suivi de murs ....). Il est aussi possible d'imposer que le potentiel calculé soit une fonction harmonique, ce qui garanti qu'il n'ait pas de minima, mais rend son estimation beaucoup plus lourde en calcul.

Le principe de ces champs de potentiels est formalisé sous le nom de *schéma moteur* par R. Arkin [4]. Pour lui, un schéma moteur est une action définie sous forme de potentiel en fonction des perceptions du robot. Ces schémas sont utilisés comme contrôleur de bas niveau dans une architecture hybride.

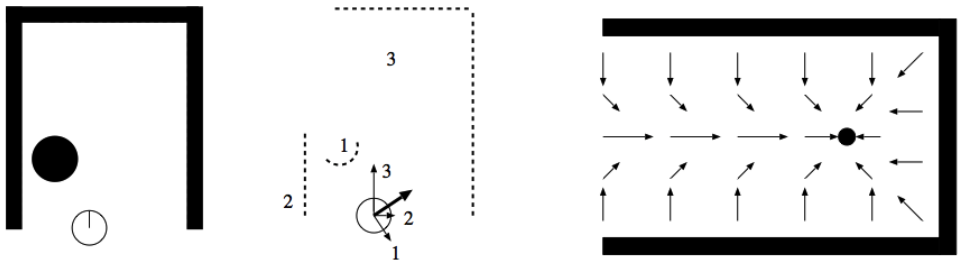


Figure 6.3: Gauche: Illustration de la combinaison de différents potentiels primitifs dans l'espace relatif au robot. Droite: Exemple de minimum local dans un champ de potentiel.

6.2 Méthode Vector Field Histogram

La méthode "Vector Field Histogram" [17] a été conçue spécifiquement pour utiliser une grille d'occupation locale construite a partir de capteurs à ultrasons. Cette grille est construite de manière très rapide par la méthode "Histogrammic in motion mapping" (voir section 11.2.3) qui produit une grille dont chaque cellule contient un nombre d'autant plus élevé qu'elle a souvent été perçue comme contenant un obstacle (Figure 6.4 haut).

Un histogramme représentant l'occupation de l'environnement autour du robot est ensuite construit à partir de cette grille d'occupation locale. Pour cela, l'environnement est discrétisé en secteurs angulaires pour lesquels la somme des valeurs des cellules est calculée (figure 6.4 bas). Un seuil permettant de tolérer un certain bruit est ensuite utilisé pour déterminer les directions possibles pour le robot : toutes les directions dont la valeur est inférieure au seuil sont considérées. Le choix de la direction est finalement réalisé parmi les directions possibles en fonction de contraintes externes (par exemple la direction la plus proche de la direction du but).

Cette méthode est extrêmement rapide (elle fonctionne sur un PC 386 à 20MHz !) et a permis historiquement un déplacement réactif à des vitesses assez élevées (environ 1 m/s). Diverses améliorations pour permettre le réglage de la vitesse du robot en fonction de la densité des obstacles ou de la largeur de l'espace angulaire libre sont possibles.

6.3 Méthode de la fenêtre dynamique

La méthode de la fenêtre dynamique [47] permet de sélectionner une trajectoire locale du robot qui va éviter les obstacles et dont les variations dans le temps vont respecter des contraintes telles que les capacités de freinage maximales du robot. Pour appliquer l'algorithme, les trajectoires locales sont paramétrées et peuvent prendre des formes différentes en fonction des contraintes d'holonomie du robot par exemple. Une méthode simple applicable à de nombreuses plateformes est d'utiliser les vitesses de translation et de rotation du robot.

La méthode de la fenêtre dynamique permet donc, à partir de la perception locale de l'environnement, de sélectionner un couple (v, ω) de vitesses de translation et de rotation du robot qui répond à

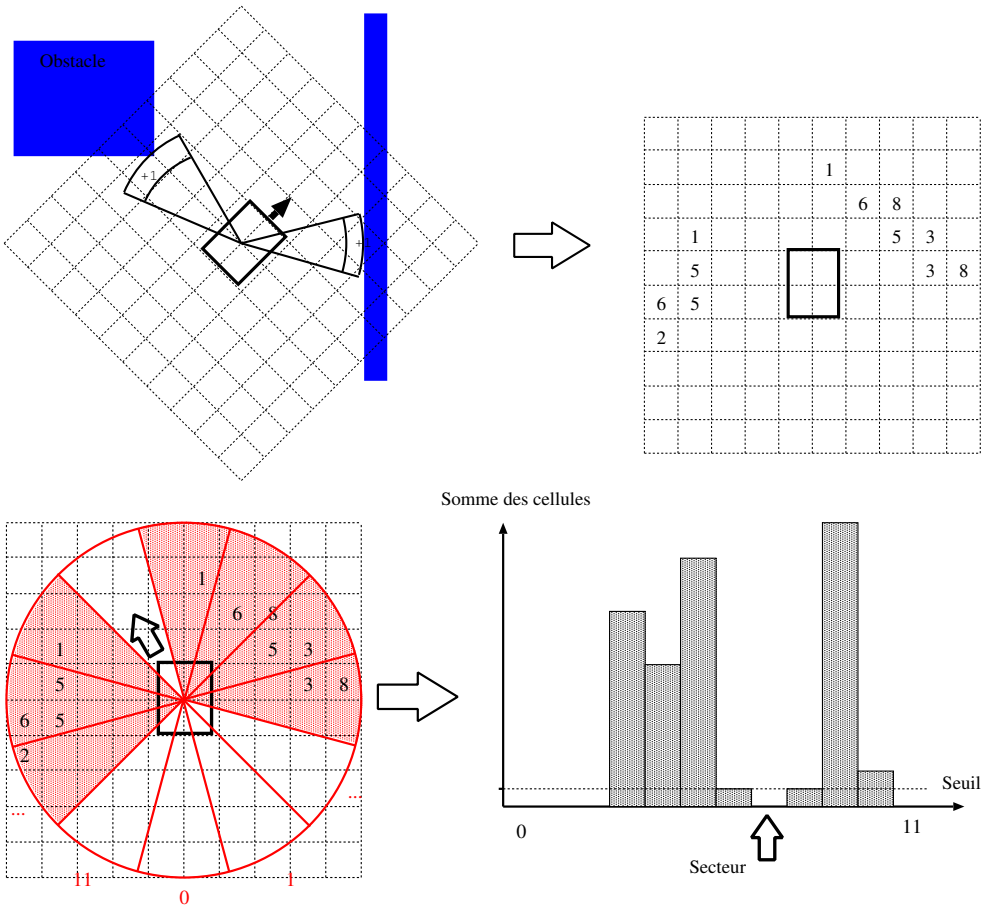


Figure 6.4: ***Partie supérieure :*** *Grille d'occupation locale construite par la méthode "Histogrammic in motion mapping". La grille est construite dans le référentiel du robot : un compteur est incrémenté pour chaque cellule appartenant au secteur angulaire dans lequel un obstacle a été détecté et les valeurs sont déplacées d'une cellule à l'autre en fonction des déplacements du robot.* ***Partie Inférieure :*** *Utilisation de l'histogramme des obstacles pour déterminer la direction de déplacement du robot.*

différentes contraintes, dont celle d'éviter les obstacles. Un tel couple de vitesses, lorsqu'il est appliqué au robot, produit une trajectoire circulaire, pour laquelle la satisfaction des différentes contraintes peut être évaluée. A l'issu de l'évaluation de toutes les contraintes pour tous les couples de vitesses possibles, la méthode de la fenêtre dynamique permet de sélectionner le couple le plus pertinent (qui répond le mieux aux contraintes).

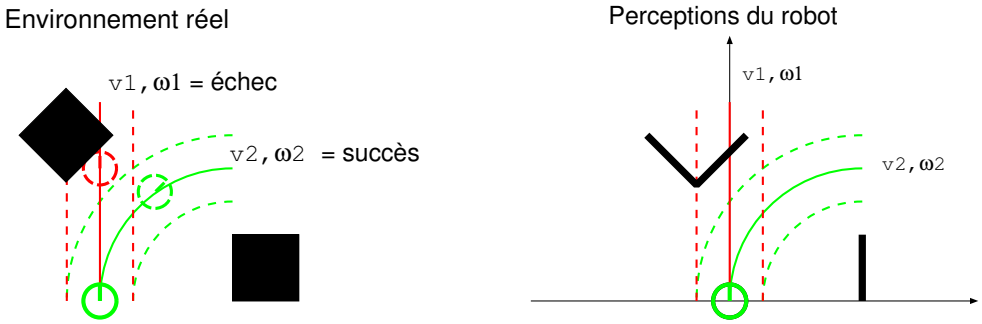


Figure 6.5: Contrainte d'évitement d'obstacles pour la méthode de la fenêtre dynamique.

La première contrainte est la contrainte d'évitement d'obstacles. C'est une contrainte dure au sens ou elle est binaire (succès / échec) et doit obligatoirement être satisfaite. Elle est évaluée pour chacune des trajectoires possibles à partir de la perception locale de l'environnement à un instant donné et de la position estimée du robot à un pas de temps fixé dans le futur pour la trajectoire courante. Si le robot n'a pas rencontré d'obstacles à cet horizon, la contrainte est respectée; dans le cas contraire, elle ne l'est pas (Figure 6.5).

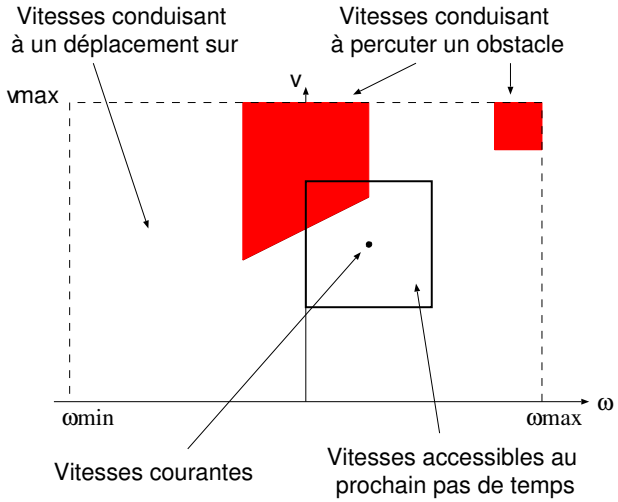


Figure 6.6: Fenêtre de sélection des vitesses tenant compte de la dynamique du robot.

Le respect ou le non respect de cette contrainte est reporté dans un graphe des vitesses qui indique, pour chaque couple de vitesses possible (donc chaque trajectoire), si le robot va ou ne va pas rencontrer un obstacle (Figure 6.6). Dans ce graphe, il est alors possible de tracer la fenêtre

des vitesses accessibles au prochain pas de temps à partir des vitesses courantes du robot et des valeurs d'accélération et décélération maximales. C'est cette fenêtre qui donne son nom à la méthode car elle permet de prendre en compte la dynamique du robot (à travers la capacité de freinage et d'accélération). Il reste alors à choisir, au sein de cette fenêtre, un couple de vitesses qui ne conduise pas à percuter un obstacle pour garantir un déplacement sûr du robot.

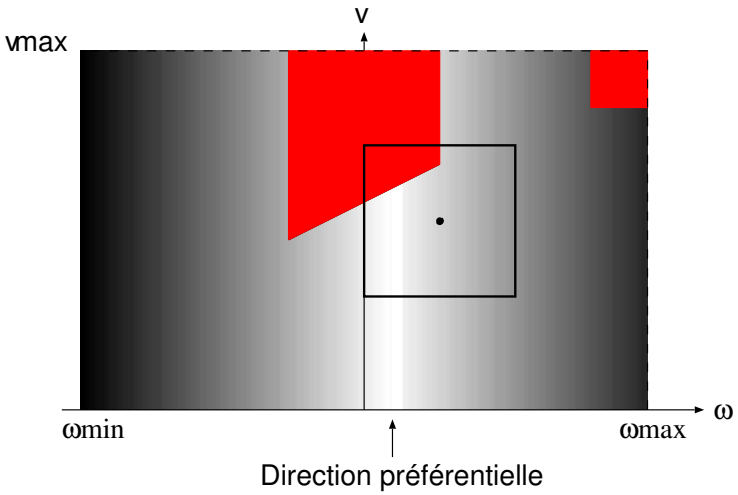


Figure 6.7: Contrainte "souple" exprimant une préférence sur la direction à prendre.

Pour faire le choix parmi toutes les vitesses possibles au sein de cette fenêtre, il est possible d'utiliser des contraintes "souples" supplémentaires pour exprimer des préférences au sein de cet espace des vitesses accessibles. Ces contraintes s'expriment par une fonction de coût $G(v, \omega)$ qui est en général la somme de plusieurs termes. Ces termes peuvent exprimer une préférence a priori sur les vitesses, une préférence pour les trajectoires s'éloignant le plus des obstacles, ou une préférence de direction si l'on dispose par exemple d'une estimation de la direction d'un but à long terme (Figure 6.7). Le couple de vitesses minimisant ce coût au sein de la fenêtre est alors sélectionné. Il garantit un déplacement sans rencontrer d'obstacles et le meilleur respect possible des contraintes souples dans ce cadre.

Dans la pratique, les valeurs des différentes contraintes sont évaluées en différents points du graphe des vitesses, le nombre de points dépendant notamment de la puissance de calcul disponible et de la complexité de l'évaluation de chaque contrainte. L'utilisation de la fenêtre dynamique est très intéressante pour un robot se déplaçant rapidement, ou pour un robot ayant des capacités d'accélération et de ralentissement limitées. Elle permet alors de produire un déplacement du robot sûr et régulier. Pour des robots qui ont une forte capacité d'accélération et de décélération (par exemple un robot léger avec de bons moteurs électriques), on peut considérer que toutes les vitesses sont accessibles presque instantanément. Il peut alors être suffisant de ne considérer que la cinématique, et non la dynamique, ce qui se traduit par la prise en compte d'un seul point du graphe, et non d'une fenêtre. La recherche du couple de vitesse est ainsi simplifiée.

Chapter 7

Apprentissage par renforcement

Les méthodes que nous avons vu jusqu'à présent sont des associations entre perceptions et actions conçues par des ingénieurs. Or il existe des techniques d'apprentissage (notamment l'apprentissage par renforcement) permettant de créer des associations de ce type à partir d'informations d'assez haut niveau sur la tâche à réaliser.

L'apprentissage par renforcement est une méthode qui permet de trouver, par un processus *d'essais et d'erreurs*, l'action optimale à effectuer pour chacune des situations que le robot va percevoir afin de maximiser une récompense. C'est une méthode d'apprentissage *orientée objectif* qui va conduire à un contrôleur optimal pour la tâche spécifiée par les récompenses. Cette méthode est de plus *non supervisée* car la récompense ne donne pas l'action optimale à réaliser mais simplement une évaluation de la qualité de l'action choisie. Elle permet enfin de résoudre les problèmes de *récompense retardée* pour lesquels il faut apprendre a sacrifier une récompense à court terme pour obtenir une plus forte récompense à long terme et donc apprendre de bonnes *séquences d'actions* qui permettront de maximiser la récompense à long terme.

Du fait de toutes ces caractéristiques, l'apprentissage par renforcement est une méthode particulièrement adaptée à la robotique.

7.1 Formalisation

Le problème de l'apprentissage par renforcement pour un agent se formalise à partir des éléments suivants :

- Un ensemble d'états $\mathcal{S}$ correspondant à la perception que l'agent a de l'environnement,
- Un ensemble d'actions possibles $\mathcal{A}$,
- Une fonction de récompense $R : \{\mathcal{S}, \mathcal{A}\} \rightarrow \mathbb{R}$.

L'agent va interagir avec son environnement par pas de temps discrets, en percevant l'état de l'environnement s_t, en choisissant une action a_t en fonction de cet état et en recevant la récompense r_{t+1} associée (Figure 7.1).

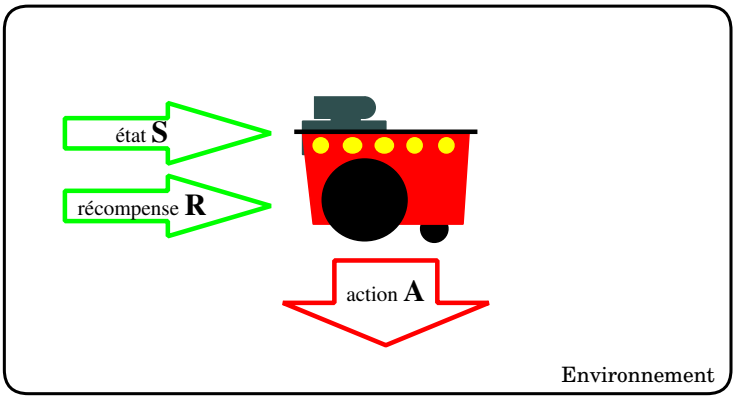


Figure 7.1: Formalisation du problème d'apprentissage par renforcement.

L'évolution du robot dans son environnement est régi par un *Processus de Décision Markovien* (MDP en anglais) qui décrit l'évolution de l'état et de la récompense en fonction des actions du robot. Ce MDP (7.2), qui spécifie complètement la tâche du robot par le jeu des récompenses, se décrit simplement à l'aide de deux fonctions :

- Une fonction de transition $\mathcal{P}^a_{ss'} = P(s_{t+1} = s' | s_t = s, a_t = a)$ qui donne la probabilité de passer dans l'état s' lorsque l'agent effectue l'action a dans l'état s,
- Une fonction de récompense $\mathcal{R}^a_{ss'} = E(r_{t+1} | s_t = s, a_t = a, s_{t+1} = s')$ qui donne la récompense moyenne lorsque l'agent passe de l'état s à s' en faisant l'action a.

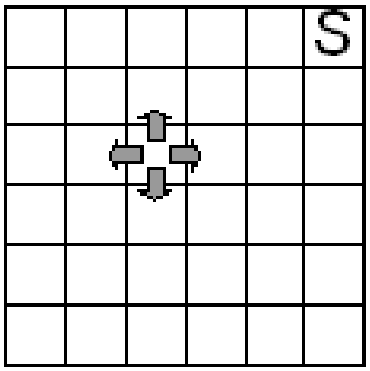


Figure 7.2: Exemple de MDP très simple. Chaque case correspond à un état et, pour chaque état, 4 actions sont possibles qui conduisent aux cases voisines. La récompense est nulle partout, sauf pour les actions qui mènent à la case "S", pour lesquelles la récompense est 1.

Le comportement de l'agent est défini par une *politique* $\pi : \{\mathcal{S}, \mathcal{A}\} \rightarrow [0, 1]$, qui guide l'agent de manière probabiliste en spécifiant, pour chaque état s la probabilité de réaliser l'action a (et donc $\sum_a \pi(s, a) = 1$). Le but de l'apprentissage par renforcement va être de trouver la politique

optimale π^* maximisant la récompense à long terme[1].

La récompense à long terme, que nous appellerons revenu R_t, peut être définie de différentes manières en fonction de la tâche considérée. Si la tâche consiste à répéter des épisodes qui durent un nombre de pas de temps fixe, le revenu pourra être la somme des récompenses instantanées pendant un épisode. Si au contraire la tâche se déroule de manière continue, le revenu pourra se définir comme la somme des récompenses futures pondérées par une exponentielle décroissante :

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

où $\gamma \in [0,1]$ est un facteur indiquant l'importance que l'on accorde aux récompenses futures.

Les algorithmes d'apprentissage par renforcement que nous verrons plus loin utilisent quasiment tous une *fonction de valeur* V^π (Figure 7.3) qui permet, pour une politique π donnée, d'estimer le revenu moyen (les récompenses futures) pour un état donné si l'on suit la politique considérée :

$$V^\pi(s) = E_\pi(R_t | s_t = s)$$

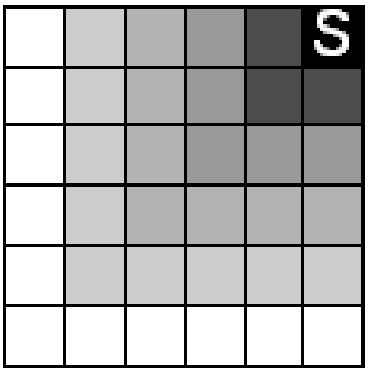


Figure 7.3: La fonction de valeur optimale dans notre exemple : pour chaque état, le niveau de gris indique la récompense à long terme qui sera obtenue en prenant le chemin le plus court vers le but.

Ces fonctions de valeurs peuvent aussi se définir non pas pour un état mais pour un état et une action réalisée dans cet état:

$$Q^\pi(s,a) = E_\pi(R_t | s_t = s, a_t = a)$$

[1] L'apprentissage par renforcement n'utilise que l'état courant pour prendre une décision, il suppose donc que toute l'information nécessaire est contenue dans cet état. Le problème est donc considéré comme étant Markovien, ce qui est rarement le cas en pratique en robotique. Si le problème est non markovien, c'est a dire si pour un même état deux actions différentes sont optimales en fonction d'une variable inconnue au robot, l'apprentissage par renforcement fournira la politique optimale, mais seulement dans l'ensemble des politiques "myopes", n'ayant pas toutes les informations pour une décision optimale.

La fonction de valeur pour un état s étant la moyenne des $Q^{\pi}(s,a)$, pondérées par la probabilité de chaque action :

$$V^{\pi}(s) = \sum_{a} \pi(s,a) Q^{\pi}(s,a)$$

Une propriété essentielle de ces fonctions de valeur va permettre de créer les différents algorithmes d'apprentissage, il s'agit de la relation de récurrence connue sous le nom d'*équation de Bellman* :

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s') \right]$$

Cette équation traduit une cohérence de la fonction de valeur en reliant la valeur d'un état à la valeur de tous les état qui peuvent lui succéder . Elle se déduit simplement de la définition de v^{π} de la manière suivante :

$$\begin{aligned}
V^{\pi}(s) &= E_{\pi}(R_t | s_t = s) \\
&= E_{\pi}(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s) \\
&= E_{\pi}(r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s) \\
&= \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma E_{\pi} \left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_{t+1} = s' \right) \right] \\
&= \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s') \right]
\end{aligned}$$

La fonction de valeur permet de caractériser la qualité d'une politique, elle donne, pour chaque état, le revenu futur si l'on suit cette politique. Elle permet également de comparer les politiques en définissant un ordre partiel :

$$\pi \geq \pi' \Leftrightarrow \forall s, V^{\pi}(s) \geq V^{\pi'}(s)$$

Cet ordre permet de définir la fonction de valeur de la politique optimale (Figure 7.3) que l'apprentissage par renforcement va chercher à estimer :

$$V^{\star}(s) = \max_{\pi} V^{\pi}(s)$$

fonction qui peut aussi s'exprimer pour un couple état-action :

$$Q^{\star}(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

avec la relation suivante :

$$Q^{\star}(s,a) = E(r_{t+1} + \gamma V^{\star}(s_{t+1}) | s_t = a, a_t = a)$$

Il est également possible d'écrire une relation de récurrence pour la fonction de valeur optimale qui sera légèrement différente de l'équation de Bellman. On parle alors d'*équation d'optimalité de Bellman*, qui peut s'écrire :

$$\begin{aligned} V^*(s) &= \max_a E\left(r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\right) \\ &= \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^*(s')\right] \end{aligned}$$

soit, pour la fonction Q :

$$\begin{aligned} Q^*(s,a) &= E\left(r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a\right) \\ &= \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q^*(s', a')\right] \end{aligned}$$

Intuitivement, cette équation traduit par l'opérateur max_a le fait que la politique optimale choisit l'action qui va maximiser le revenu.

Si l'environnement est un MDP fini, ces équations de Bellman ont une solution unique qui va donc donner pour chaque état la récompense maximale que pourra recueillir l'agent à partir de cet instant. A partir de la connaissance de V^*, il est très simple de construire une politique optimale, en associant à chaque état la ou les actions a' qui permettent de réaliser le maximum de l'équation d'optimalité de Bellman :

$$a' = \underset{a}{argmax}\, E\left(r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\right)$$

ou, si l'on utilise la fonction Q :

$$a' = \underset{a}{argmax}\, Q^*(s,a)$$

il faut cependant noter que la construction de la politique à partir de V^* nécessite la connaissance du modèle du monde afin d'estimer r_{t+1} et s_{t+1}. Cette connaissance est inutile si l'on utilise Q^*.

Tout le problème pour l'apprentissage va donc être d'estimer V^* ou Q^* pour en déduire une politique optimale. On peut remarquer que si l'on connaît complètement le problème (c'est a dire si l'on connaît $\mathcal{P}^a_{ss'}$ et $\mathcal{R}^a_{ss'}$), il est possible de calculer directement V^* ou Q^* en résolvant le système des équations d'optimalité de Bellman pour chaque état. Cette solution est peu applicable en robotique car l'environnement est en général inconnu et de plus elle ne correspond pas à des caractéristiques souhaitables de l'apprentissage tel que la capacité à apprendre par essais et erreurs.

7.2 Programmation dynamique

La programmation dynamique est un ensemble de méthodes permettant de calculer une politique optimale dans un MDP connu, en utilisant les propriétés des équations de Bellman. Nous supposons donc dans cette section que le modèle de l'environnement est connu. La programmation dynamique va chercher à estimer la fonction de valeur optimale V^* afin d'en déduire une politique optimale.

7.2.1 Évaluation d'une politique

La première étape de la programmation dynamique est l'estimation de la fonction de valeur pour une politique donnée π. Cela peut se faire soit en résolvant le système des équations de Bellman, soit en utilisant une procédure itérative, que nous préférerons car elle s'applique plus naturellement, notamment en cas de contraintes temps-réel. Cette procédure utilise le fait que la fonction de valeur V^π est le point fixe de l'équation de Bellman :

$$V(s) = \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s') \right]$$

Nous pouvons ainsi utiliser cette équation comme étape de mise-à-jour permettant de calculer une suite de fonctions V^k qui convergera vers V^π :

$$V^{k+1}(s) = \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^k(s') \right]$$

La méthode de programmation dynamique va donc utiliser cette mise-à-jour tant que les modifications $max_s(|V^{k+1} - V^k)$ seront supérieures à un seuil donné pour fournir une approximation de V^π.

7.2.2 Amélioration d'une politique

Après l'évaluation d'une politique, il va être possible de calculer une meilleur politique à partir de sa fonction de valeur associée. En effet, pour une politique donnée, il n'y a aucune raison que la fonction de valeur satisfasse l'équation d'*optimalité* de Bellman, c'est à dire que l'on peut avoir :

$$\pi(s) \neq \underset{a}{argmax} E\left(r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s, a_t = a\right)$$

Par contre, on peut prouver que la politique π' définie par :

$$\pi'(s) = \underset{a}{argmax} E\left(r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s, a_t = a\right) \tag{7.1}$$

est meilleure ou équivalente à la politique π, ce qui nous permet d'améliorer notre politique initiale.

De plus, si la politique π' ainsi définie n'est pas meilleure que π (c'est à dire si $V^{\pi'} = V^\pi$), la définition de π' (eq 7.1) est l'équation d'optimalité de Bellman, qui prouve donc que la politique obtenue est optimale.

7.2.3 Algorithmes d'apprentissage

L'évaluation et l'amélioration de politique peuvent être utilisées de différentes manières pour estimer une politique optimale pour un MDP donné.

Le première méthode, l'*itération de politique* utilise simplement ces deux phases de manière itérative :

$$\pi_0 \rightarrow V^{\pi_0} \rightarrow \pi_1 \rightarrow V^{\pi_1} \ldots \pi^* \rightarrow V^{\pi^*}$$

Dans ce processus, cependant, l'évaluation de politique est elle-même un processus itératif, qui ne va converger qu'a une erreur donnée près et de plus être potentiellement très long.

Une autre méthode pour converger vers la politique optimale est d'améliorer la politique avant même d'avoir une estimation correcte de sa valeur. On peut par exemple faire un nombre fixe d'itérations d'évaluation avant de faire une amélioration. Le cas ou on ne fait qu'une itération d'évaluation de la politique est l'algorithme d'*itération de valeur*, pour lequel les deux étapes se réduisent à une seule :

$$V^{k+1}(s) = \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^k(s') \right]$$

et qui converge vers V^*.

Pour les problèmes avec de très grands espaces d'états, le fait de parcourir tout les états pour la mise-à-jour peut être difficile en soit. Il est dans ce cas possible d'utiliser une méthode de *programmation dynamique asynchrone* qui réalise la mise-à-jour de l'itération de valeur pour un état sélectionné au hasard, ou en fonction du comportement de l'agent. Cette méthode converge également vers V^* à condition de visiter à la limite tout les états un nombre infini de fois. Elle possède l'avantage de fournir rapidement une approximation de la fonction de valeur.

7.3 Méthodes de Monte-Carlo

Le fait de devoir connaître l'environnement pour apprendre une stratégie rend les méthodes de programmation dynamique peu utiles en robotique. Les méthodes de Monte-Carlo que nous allons voir dans cette section vont utiliser les mêmes idées (estimer la fonction de valeur puis améliorer la politique), mais en ayant recours à des expériences réalisées dans l'environnement plutôt qu'à un modèle.

7.3.1 Évaluation d'un politique

L'estimation de la fonction de valeur va se réaliser à partir d'un ensemble de séquences "état-action-récompenses-état-action ...." réalisées par l'agent. Pour les états de ces séquences, il est alors possible d'estimer V simplement par la moyenne des revenus :

$$V(s) = \frac{1}{N(s)} \sum Revenu(s)$$

où $N(s)$ est le nombre de séquences ou apparaît s et $Revenu(s)$ est le revenu après la première visite de l'état s, c'est à dire la somme pondérée des récompenses après cette visite[2].

Cette méthode de Monte-Carlo a de plus l'avantage d'estimer la valeur de chaque état indépendemment, contrairement à la programmation dynamique qui doit estimer simultanément tout les états, ce qui permet par exemple de se concentrer sur des zones de l'espace d'états plus importantes pour l'objectif du robot.

[2] Il est également possible d'intégrer les nouvelles séquences de manière itérative en utilisant un mise à jour du type $V(s) \leftarrow V(s) + \alpha [R(s) - V(s)]$.

Cette méthode s'applique de la même façon pour une fonction $Q(s,a)$, ce qui est encore plus intéressant, car pour trouver la politique optimale à partir de V^* il faut disposer d'un modèle de l'environnement, ce qui n'est pas le cas en utilisant Q^*. L'utilisation de Q et de la méthode de Monte-Carlo permet donc de découvrir la politique optimale sans aucun modèle de l'environnement, en utilisant uniquement des expériences réalisée dans cet environnement.

7.3.2 Besoin d'exploration

La méthode de Monte-Carlo présentée doit estimer les valeurs $Q(s,a)$ à partir des récompenses obtenues après avoir réalisé l'action a dans l'état s. Ceci suppose donc que tout ces couples (s,a) soient rencontrés une infinité de fois à la limite. Ceci est particulièrement problématique, car toutes les politiques ne peuvent garantir cette propriété. Il faut donc ajouter au comportement défini par la politique un comportement d'exploration qui va assurer que toutes les actions seront réalisées avec un certaine probabilité (même faible).

Deux solutions existent pour résoudre ce problème. La première consiste à contraindre les politiques pour qu'elles associent au moins une faible probabilité proportionelle à un paramètre ε à toutes les actions. L'apprentissage converge ainsi vers la politique optimale au sein de cette classe. Cette probabilité garanti une exploration exhaustive et peut être diminuée au cours du temps lorsque les données suffisantes pour évaluer la politique ont été recueillies. Cette méthode s'appelle contrôle *"on policy"* car elle modifie la politique effectivement suivie par l'agent et évalue cette politique modifiée.

La seconde méthode est une méthode *"off policy"* car elle évalue une politique tandis que l'agent en suit une autre. Cette autre politique choisit en général l'action de la politique originale avec une probabilité $1-\varepsilon$ et une action aléatoire avec une probabilité ε. Pour évaluer la politique originale, l'évaluation de Monte-Carlo utilise simplement les parties finales des séquences pour lesquelles le choix d'action correspond au choix qui aurait été fait par la politique originale, mais modifie les pondérations des récompenses pour compenser les différences de probabilité de choix des actions entre les deux politiques.

7.3.3 Algorithmes d'apprentissage

L'apprentissage utilisant un méthode de Monte-Carlo se fait en alternant l'évaluation d'une politique et son amélioration en prenant l'action maximisant le revenu dans chaque état. Cette alternance peut se réaliser de plusieurs manières, comme pour la programmation dynamique. Soit l'évaluation est complète avant de réaliser une amélioration, soit il est possible d'alterner une évaluation utilisant une seule séquence, puis une amélioration.

7.4 Apprentissage par différences temporelles

Les deux méthodes que nous avons vu précédemment ont chacune un avantage important. La méthode de Monte-Carlo permet d'apprendre à partir d'expériences, sans aucun modèle

de l'environnement. La programmation dynamique pour sa part possède la propriété intéressante d'utiliser les estimations des états successeurs pour estimer la valeur d'un état, on parle de *"bootstrap"*. Cette caractéristique permet une convergence beaucoup plus rapide (en terme de nombre d'exemples états/actions/récompenses) que la méthode de Monte-Carlo. Les méthodes d'apprentissage par différences temporelles vont réunir ces deux propriétés et constituent les méthodes les plus utiles en pratique en robotique.

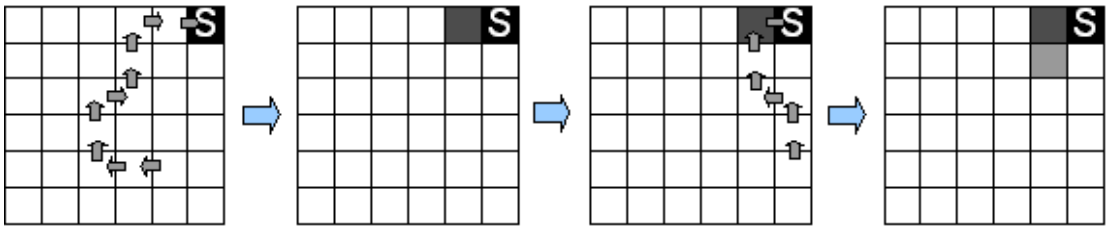


Figure 7.4: Illustration de la méthode des différences temporelles. La valeur d'un état est mise à jour en fonction de la récompense immédiate (exemple du schéma 2) et des estimations précédentes de la fonction de valeur (exemple du schéma 4).

L'évaluation de politique va donc se faire à partir d'expériences comme pour la méthode de Monte-Carlo qui utilise une moyenne pour estimer la valeur d'un état. Cette moyenne mise sous forme itérative conduit à l'équation suivante :

$$V(s_t) \leftarrow V(s_t) + \alpha [R_t - V(s_t)]$$

avec

$$R_t = r_{t+1} + \gamma r_{t+2} + ... + \gamma^p r_{t+p+1}$$

Par rapport à cette équation, l'idée est ici d'utiliser l'estimation du revenu à partir de la récompense suivante et de la valeur de l'état suivant, au lieu d'utiliser les récompenses de la suite de l'expérience. La méthode des différences temporelles utilise donc la mise-à-jour suivante :

$$V(s_t) \leftarrow V(s_t) + \alpha [r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

Cette mise a jour se fait naturellement de manière incrémentale, au fur et à mesure des expériences de l'agent. L'équivalent de cette mise à jour pour la fonction Q est donné par :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Cette méthode est appelée *Sarsa* (pour State, Action, Reward, State, Action) et réalise une estimation "on policy" de la politique suivie car elle utilise l'action a_{t+1} qui dépend de de la politique.

La méthode la plus importante de l'apprentissage par renforcement est probablement le *Q-Learning* qui est la variante "off policy" de Sarsa qui va utiliser le maximum de la fonction sur les actions suivantes au lieu de l'action effectivement réalisée :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{7.2}$$

Cet algorithme fait converger Q vers Q^* indépendemment de la politique suivie, tant que cette politique garanti une exploration exhaustive (cad une probabilité non nulle pour toutes les actions dans tous les états).

Comme pour les autres méthodes, la politique optimale se déduit simplement de la fonction valeur optimale.

7.5 Traces d'éligibilité

Les méthodes utilisant les différences temporelles que nous avons vues précedement permettent de remplacer la fin d'un épisode qui serait utilisé par une méthode de Monte-Carlo pour estimer le revenu par la valeur estimée de l'état suivant. Or au cours de l'apprentissage, cette valeur n'est pas forcément correcte et pourrait être remplacée par d'autres évaluations.

Le premier exemple est l'utilisation de n pas du futur, qui conduit à la méthode des différences temporelles à n pas. Dans cette méthode, le revenu est estimé par une équation de la forme :

$$R_t = R_t^n = r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^n (r_{t+n+1} + \gamma V(s_{t+n+1}))$$

La mise a jour se ferait alors par l'équation :

$$V(s_t) \leftarrow V(s_t) + \alpha [R_t^n - V(s_t)]$$

Il est également possible d'estimer R_t en utilisant des moyennes pondérées de R_t^n :

$$R_t = \sum_i a_i R_t^i$$

avec $\sum a_i = 1$.

Un cas particulier interessant de cette dernière méthode est l'utilisation d'une pondération exponentielle qui va faire décroitre l'importance des expériences au fur et a mesure de leur éloignement dans le temps :

$$R_t = (1-\lambda) \sum_{i=1}^{\infty} \lambda^i R_t^i$$

Ce cas particulier est intéressant car il peut s'appliquer simplement en utilisant les valeurs passées des récompenses, au lieu des valeurs futures (on démontre que les mises à jour sont les mêmes). On utilise pour cela une valeur auxiliaire appelée *trace d'éligibilité* que l'on définit de la manière récursive suivante :

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & si \quad s \neq s_t \\ \gamma\lambda e_{t-1}(s) + 1 & si \quad s = s_t \end{cases}$$

La mise à jour des valeurs se fait alors pour chaque état proportionellement à la valeur de son éligibilité (Figure 7.5) :

$$V(s_t) \leftarrow V(s_t) + \alpha e(s_t) [r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

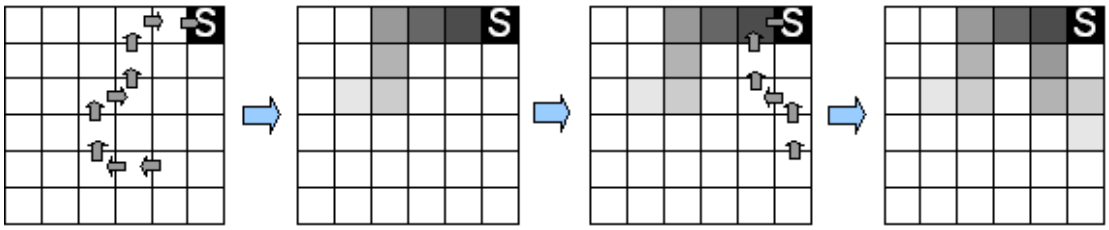


Figure 7.5: Illustration de la méthode des traces d'eligibilités. La méthode des différences temporelles est utilisée, mais propagée avec une décroissance à tous les états de l'historique de l'agent.

L'idée de cette mise à jour est de remplacer la mise à jour d'un état en utilisant des récompenses futures par la mise à jour des états passés en utilisant la récompense courante. Cela donne au final l'algorithme TD(λ)[3] (algorithme 1). Cet algorithme se décline simplement pour les extension Sarsa(λ) et Q(λ).

Algorithm 1 Algorithme TD(λ)

Initialiser $V(s)$ aléatoirement et $e(s) = 0$ pour tout s
for all épisodes **do**
  Initialise s
  for all pas de l'épisode **do**
    $a \leftarrow \pi(s)$
    Executer a, recueillir r et l'état suivant s'
    $\delta \leftarrow r + \gamma V(s') - V(s)$
    $e(s) \leftarrow e(s) + 1$
    for all s **do**
      $V(s) \leftarrow V(s) + \alpha\delta e(s)$
      $e(s) \leftarrow \gamma\lambda e(s)$
    end for
    $s \leftarrow s'$
  end for
end for

7.6 Application pratique

En robotique mobile, les espaces d'entrée et de sortie des capteurs et des effecteurs sont rarement discrets, ou, si ils le sont, le nombre d'état est très grand. Or l'apprentissage par renforcement tel que nous l'avons décrit utilise des espaces d'état et d'action qui doivent être de taille raisonnable pour permettre aux algorithmes de converger en un temps utilisable en pratique.

[3] TD pour Temporal Differences

Pour permettre d'utiliser l'apprentissage par renforcement, plusieurs approches sont possibles. La première consiste à discrétiser manuellement le problème afin de fournir des espaces de quelques centaines d'états qui pourront être utilisés directement par des versions naïves des algorithmes (utilisant par exemple simplement des tableaux de valeurs $Q[s][a]$, c'est cette méthode que nous avons utilisé dans l'exemple de la section suivante). Il faut cependant bien faire attention au choix des discrétisations afin qu'elle permettent un apprentissage correct en fournissant des états et des actions qui conduisent notamment à des récompenses cohérentes. Ce choix peut être relativement simple pour des capteurs intuitifs comme les capteurs de distance, mais être complexe voir impossible si l'espace d'entrée est plus abstrait ou si sa structure est peu connue (par exemple pour un jeu comme le backgammon).

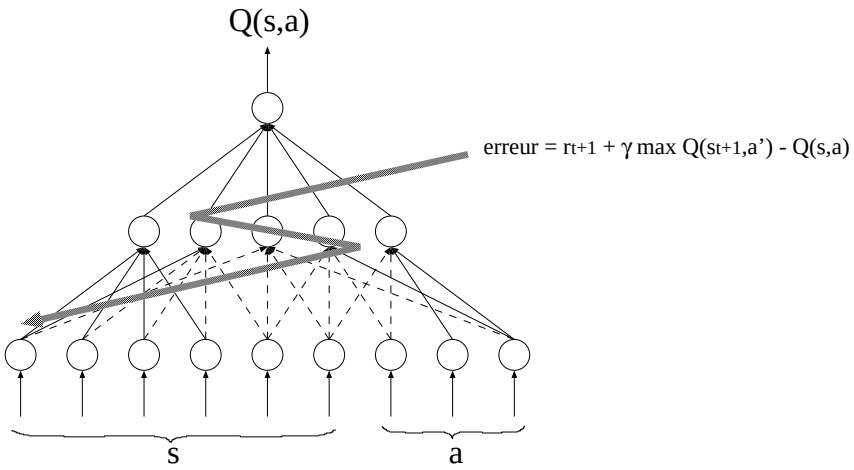


Figure 7.6: Exemple de réseau de neurones simple permettant l'approximation de $Q(s,a)$.

La seconde méthode va permettre de travailler directement dans les espaces d'états continus des capteurs en utilisant des méthodes d'approximation de fonction. En effet, pour utiliser l'apprentissage par renforcement, il est simplement nécessaire d'estimer correctement la fonction $Q(s,a)$ (par exemple). Or cette estimation peut se faire directement par un approximateur de fonction continu, par exemple un réseau de neurones (figure 7.6), que l'on entraine à l'aide des données recueillies sur le problème. L'utilisation de ce type d'approximation permet de travailler directement dans l'espace continu et donc de limiter les effets parasites qui pourraient apparaître suite à un mauvais choix de discrétisation. Ces méthodes peuvent posséder cependant des inconvénients, notamment de réaliser un apprentissage non local (comme dans le cas des réseaux de neurones), ce qui entraine des modifications incontrôlées de valeurs pour des couples (s,a) qui ne sont pas ceux pour lesquels on réalise l'apprentissage. Certaines méthodes utilisant d'autres types d'approximation (comme la régression pondérée localement [118]) permettre de s'affranchir de ces contraintes.

7.7 Exemple de mise en œuvre

Dans cette exemple, nous avons utilisé un algorithme de Q-Learning très simple pour apprendre à un robot à éviter les obstacles. Le robot possède un télémètre laser et une base mobile différentielle. Nous avons discrétisé les données du télémètre selon le schéma de la figure 7.7 pour constituer un espace d'entrée de 2187 états. Les actions ont été discrétisée en trois actions élémentaires : avancer, tourner à droite, tourner à gauche.

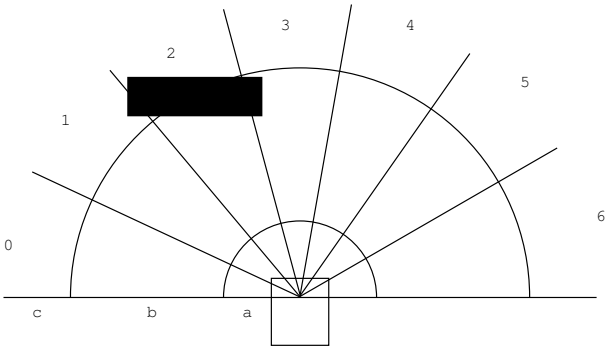


Figure 7.7: Discrétisation de l'espace d'états : pour chaque secteur, de 0 à 6, une valeur O_i est calculée en fonction de la présence ou de l'absence d'obstacles dans les zones a, b et c : 0 si un obstacle est dans a, 1 si un obstacle est dans b et 2 sinon. L'état s est la valeur en base 3 fournie par les des O_i : $\sum_k 3^k O_k$. Dans l'exemple, $O_0 = 2, O_1 = 2, O_2 = 1, O_3 = 1, O_4 = 2, O_5 = 2, O_6 = 2$ et donc $s = 2150$.

La récompense était de -10 lorsque le robot percute un obstacle (il est alors remis à son point de départ) et +3 lorsque le robot choisit l'action d'avancer. L'algorithme de Q-Learning utilise un simple tableau de 2187x3 cases pour représenter la fonction Q. La courbe 7.8 donne l'évolution au cours du temps des récompenses obtenues sur des épisodes de 100 pas de temps. Les pas de temps ont une longueur variable et correspondent aux changements d'état. Un épisode dure en moyenne 35 secondes. La figure illustre également les trajectoires aléatoires initiales et le résultat après convergence de l'algorithme.

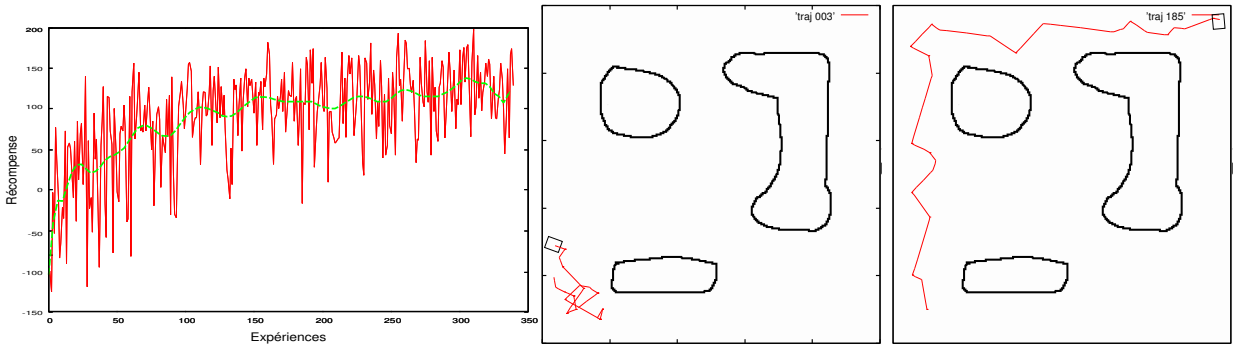


Figure 7.8: Récompenses obtenues au cours du temps et exemples de trajectoires obtenues au premier épisode et à l'épisode 185.

7.8 Pour aller plus loin

Reinforcement Learning: An Introduction de Richard S. Sutton and Andrew G. Barto. MIT Press, Cambridge, MA, 1998. A Bradford Book. Disponible en ligne[4]

Le reinforcement learning repository : http://www-anw.cs.umass.edu/rlr/

[4] http://www.cs.ualberta.ca/~sutton/book/the-book.html

Part III

Navigation utilisant une carte

Cette partie présente les méthodes de navigation basées sur des cartes qui permettent à un robot de prendre en compte des buts à long terme en utilisant des informations mémorisées sur la structure de son environnement. Ces méthodes se basent sur trois processus : *la cartographie, la localisation et la planification*, détaillés dans les différents chapitres.

Chapter 8

Localisation, Cartographie et Planification

8.1 Les trois problèmes de la navigation par carte

Le processus complet qui permet à un robot de mémoriser son environnement, puis de s'y déplacer pour rejoindre un but, peut être découpé en trois phases : la cartographie, la localisation et la planification. Ces trois phases permettent de répondre aux trois questions fondamentales pour la tâche de navigation [93] : Où suis-je ? Où sont les autres lieux par rapport à moi ? et Comment puis-je atteindre mon but ?

- La cartographie est la phase qui permet la construction d'une carte reflétant la structure spatiale de l'environnement à partir des différentes informations recueillies par le robot.
- Une telle carte étant disponible, la localisation permet alors de déterminer la position du robot dans la carte qui correspond à sa position dans son environnement réel.
- La planification , enfin, est la phase qui permet, connaissant la carte de l'environnement et la position actuelle du robot, de décider des mouvements à effectuer afin de rejoindre un but fixé dans l'environnement.

Ces trois phases sont évidemment fortement interdépendantes. L'ordre dans lequel elles sont citées fait directement apparaître le fait que la seconde phase dépend de la première. En effet, estimer sa position au sein d'une carte de l'environnement suppose implicitement que cette carte existe et qu'elle contient la position courante du robot. De même, la troisième phase dépend des deux premières, car la planification suppose que l'on connaisse sa position et que la carte de l'environnement représente une portion de l'environnement contenant au moins un chemin reliant cette position au but qui doit être atteint.

Mais la relation entre les deux premières phases est plus subtile qu'une simple relation d'antériorité : c'est le même problème que pour l'œuf et la poule. Chacun des deux éléments peut, en effet, être considéré comme préalable à l'autre mais dépend aussi de l'autre pour sa réalisation. Dans le cas de la cartographie et de la localisation, nous avons déjà vu que la localisation repose sur une phase préalable de cartographie. Mais pour construire une carte, il est

nécessaire de savoir où ajouter, dans la carte partielle déjà existante, toute nouvelle information recueillie par le robot. Cela requiert donc une phase préalable de localisation au sein de la carte partielle déjà existante. Pour un robot complètement autonome, il est donc impossible de ne pas traiter ces deux problèmes simultanément. Dans la littérature scientifiques, on parle ainsi de problème de "Simultaneous Localization and Mapping" (SLAM) .

Dans le cas où l'on autorise un opérateur humain à intervenir dans le processus, il est évidemment possible de découpler ces deux phases. Dans les applications réelle, il est fréquent que l'on fournisse au robot une carte construite au préalable et qu'on ne s'intéresse qu'à l'estimation de la position au sein de cette carte pour qu'il puisse accomplir sa tâche. La carte peut alors être obtenue de différentes manières. Il est par exemple possible d'utiliser un plan d'architecte d'un bâtiment pour le transformer en une carte utilisable par le robot. Il est également possible d'utiliser le robot dans une phase supervisée de cartographie. Au cours de cette phase, la position du robot peut-être calculée de manière précise par un dispositif externe au système de navigation, et ne nécessite donc pas que le système estime de lui-même la position. Connaissant la position précise du robot, il est alors relativement simple de construire une carte de l'environnement. Il est également possible d'utiliser un algorithme de SLAM comme sur un robot autonome, mais de corriger la carte avant de l'utiliser réellement. Cela permet notamment d'ajouter des obstacles "virtuels" pour interdire certains passages tels que les escaliers.

8.2 Quelques hypothèses de travail

8.2.1 Estimation de la position et de la direction

A ce stade, il convient de préciser la notion de position que nous emploierons. En effet, la position d'un robot est définie à la fois par son emplacement spatial, estimé par rapport à un point de référence et par sa direction, estimée par rapport à une direction de référence. Ces deux quantités sont couplées mais ont des statuts qui peuvent être distincts en pratique.

Lors du mouvement, la direction du robot influence la manière dont varie sa position, mais peut parfois être contrôlée indépendamment. Dans le cas de plates-formes holonomes, ce découplage permet notamment de simplifier le processus de planification en ne tenant pas compte de la direction du robot, laquelle peut être contrôlée sans influencer la position. La variable importante est alors la position du robot, la direction devant être estimée pour pouvoir agir, mais non pour planifier.

Cette indépendance relative au niveau de la planification peut conduire à des systèmes d'estimation de la position et de la direction séparés. Cette séparation est supportée par le fait que la direction d'un robot peut être mesurée par des capteurs indépendamment de l'estimation de sa position. Il est par exemple possible d'utiliser une boussole qui mesure la direction par rapport à la direction du pôle magnétique, ou un gyroscope qui mesure la direction par rapport à une direction arbitraire fixe.

Le choix de représenter et d'estimer de manière séparée la position et la direction n'interdit toutefois pas des interactions entre ces informations. L'estimation de la position utilisera évidemment celle de la direction pour pouvoir intégrer de nouvelles données lors du mouvement du robot.

L'estimation de la direction pourra également dépendre de la position par un système de recalage qui utilisera la perception d'un point de référence connu depuis une position connue pour estimer la direction.

Ceci dit, la majorité des systèmes métriques (voir section 9.2) vont néanmoins estimer la position et la direction ensemble, soit en 2D (3 degrés de liberté), soit en 3D (6 DDL).

8.2.2 Environnements statiques et dynamiques

Il convient également de préciser les types d'environnements que nous considérons ici. En effet, les robots sont amenés à se déplacer dans une grande variété d'environnements qui peuvent être regroupés en deux grandes catégories : les environnements statiques et les environnements dynamiques. Les environnements statiques sont des environnements qui ne subissent pas de modifications au cours du temps. Cette stabilité concerne à la fois leur structure spatiale et leur apparence pour les capteurs du robot. Cela exclut la majorité des environnements dans lesquels les humains évoluent quotidiennement. Les environnement dynamiques, pour leur part, présentent des caractéristiques qui évoluent au cours du temps. La plupart des environnements courants appartiennent évidemment à la seconde catégorie. Par exemple, un environnement de bureau est dynamique, du fait des personnes qui y travaillent, des chaises qui y sont déplacées ou des portes qui y sont ouvertes ou fermées.

Il est, de plus, possible de distinguer deux catégories d'éléments dynamiques. La première catégorie regroupe les éléments variables qui ne caractérisent pas l'environnement. De tels éléments peuvent être considérés comme du bruit qui n'a pas d'intérêt dans la modélisation de l'environnement pour la planification. C'est, par exemple, le cas des personnes évoluant dans un bureau, ou des chaises déplacées. Ces environnements peuvent être considérés comme constitués d'une partie statique sur laquelle se superposent différentes sources de bruit. La partie statique est la partie la plus importante à modéliser pour parvenir à une navigation efficace. Deux effets du bruit doivent toutefois être pris en compte. Il faut premièrement veiller à ce qu'il n'empêche pas la réalisation de commandes issues de la planification. Cela est en général réalisé par le système de contrôle (dans le cadre d'une architecture hybride, voir section 2.2.3), séparé du système de navigation, qui réalise l'interface avec la partie physique du robot. De plus, il faut prendre en compte ce bruit au niveau de la cartographie et de la localisation afin qu'il ne nuise pas à la modélisation de la seule partie statique de l'environnement et ne conduise pas à une mauvaise estimation de la position. Les méthodes de navigation actuelles (présentées dans ce cours) sont plus ou moins robustes face à ces bruits, mais cette robustesse reste généralement limitée, surtout pour la partie cartographie. Il commence cependant à apparaître des méthodes prenant explicitement en compte ces éléments dynamiques, qui permettent d'envisager d'utiliser des robots dans des environnements assez fortement bruités (par exemple [143]).

La seconde catégorie d'éléments dynamiques regroupe les éléments variables qui caractérisent l'environnement et peuvent avoir un intérêt pour la planification. C'est, par exemple, le cas des portes qui modifient la structure spatiale de l'environnement et peuvent entraîner des modifications de trajectoires en fonction de leur état. Ils doivent donc être enregistrés dans la carte si l'on veut pouvoir les prendre en compte.

La plupart des systèmes de navigation robotiques s'intéressent aux environnements appartenant à l'une des deux premières catégories. Les environnements sont donc supposés être soit statiques, soit entachés d'un bruit qui n'influence pas la planification. Ces systèmes s'intéressent donc à modéliser la partie statique des environnements qui va être utile pour la localisation et la planification. Il faut toutefois noter que ces systèmes, qui ne modélisent pas les éléments dynamiques de la seconde catégorie, sont néanmoins capable d'évoluer dans des environnements qui contiennent de tels éléments. Pour ce faire, ces systèmes sont en général capables de vérifier que la trajectoire planifiée est correctement exécutée. En cas de problème d'exécution, un chemin alternatif ne passant pas par la zone qui ne peut être atteinte est alors recherché. Cette méthode, qui ne modélise pas explicitement les portes, par exemple, est néanmoins capable de provoquer des détours si une porte fermée bloque un chemin.

Chapter 9

Les représentations de l'environnement

Les deux utilisations possibles des perceptions présentées dans le chapitre 3 (avec et sans modèle métrique) trouvent un parallèle dans deux types de représentations de l'environnement.

Lorsqu'aucun modèle métrique n'est utilisé pour les capteurs, les données sont en général mémorisées dans une carte *topologique* [82, 131] (cf. figure 9.1). Dans une telle carte, un ensemble de lieux et leurs relations de voisinage sont mémorisées. Chaque lieu est défini au moyen de perceptions recueillies lorsque le robot se trouve à la position correspondante. Les relations entre lieux sont en général déduites des données proprioceptives.

En revanche, lorsqu'un modèle métrique des capteurs est utilisé, les données peuvent être mémorisées au sein d'une carte *métrique* [101, 29] (cf. figure 9.1) qui rassemble dans un même cadre de référence les données proprioceptives et les perceptions. La carte contient alors un ensemble d'objets, ayant chacun une position associée. Naturellement, il est possible de construire une carte topologique lorsqu'un modèle métrique est utilisé. Dans ce cas, toutefois, les perceptions ne sont en général pas utilisées pour estimer la position relative des lieux visités, mais seulement pour caractériser ces lieux.

Il est également possible d'utiliser des représentation hybrides qui vont avoir des caractéristiques à la fois topologiques et métriques afin de bénéficier des avantages de chacune des approches.

Notons que la notion de topologique et de métrique est différente de celle mentionnée pour les stratégies de navigation dans l'introduction. Ici, cette notion fait référence à la manière dont les informations sont mémorisées et non à la stratégie de navigation utilisée. Ainsi une carte topologique pourra contenir des informations métriques et pourra être utilisée pour une stratégie de navigation métrique, au sens donné dans l'introduction. Dans la suite de ce cours, le concept topologique/métrique fera toujours référence au type de carte utilisé, et non à la stratégie correspondante.

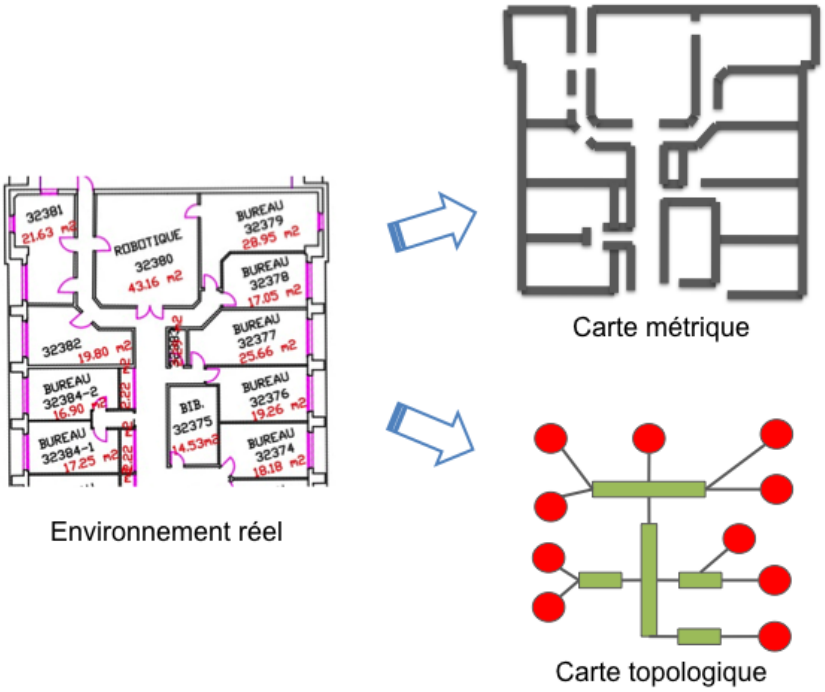


Figure 9.1: Les cartes utilisées en robotique peuvent être de deux types : les cartes topologiques, d'une part, qui mémorisent un ensemble de lieux, ainsi que les manières de se déplacer de l'un à l'autre (dans cet exemple, des pièces et des couloirs) et les cartes métriques, d'autre part, qui mémorisent un ensemble d'objets perçus (des murs dans cet exemple) avec une position dans un cadre de référence global.

9.1 Cartes topologiques

9.1.1 Description

Les cartes topologiques permettent de représenter l'environnement du robot sous forme de graphe. Les nœuds du graphe correspondent à des lieux, c'est-à-dire des positions que le robot peut atteindre. Les arêtes liant les nœuds marquent la possibilité pour le robot de passer directement d'un lieu à un autre et mémorisent en général la manière de réaliser ce passage.

La détection et la mémorisation des lieux reposent en général sur deux procédures qui utilisent les perceptions. La première permet simplement de comparer deux perceptions et donc de reconnaître un lieu de la carte ou de détecter un lieu nouveau. La seconde procédure permet de mémoriser un nouveau lieu ou d'adapter la définition d'un lieu lors des passages successifs du robot en ce lieu. Comme nous l'avons déjà mentionné, la reconnaissance d'un lieu est soumise aux problèmes de la *variabilité perceptuelle* et du *perceptual aliasing* . En conséquence, la première procédure peut donner des résultats erronés. Par exemple, un lieu déjà visité peut ne pas être reconnu, ou un lieu nouveau peut être confondu avec un lieu déjà mémorisé. Pour résoudre ces problèmes, la reconnaissance des lieux fera donc appel aux données proprioceptives en plus des perceptions. De nombreuses méthodes, dont les plus importantes seront décrites dans la

suite du cours, ont été mises en œuvre dans ce but.

Les données mémorisées dans les arêtes du graphe sur les relations de voisinage entre lieux proviennent, pour leur part (en général), des données proprioceptives. Cela est caractéristique des cartes topologiques, dans lesquelles les perceptions ne sont en général pas utilisées pour estimer les positions relatives des lieux visités, mais seulement pour reconnaître un lieu. Ces données peuvent être des informations sur les positions relatives des nœuds, ou des informations sur les actions à effectuer pour parcourir cette arête.

9.1.2 Avantages

Un avantage important des cartes topologiques est qu'elles ne requièrent pas de modèle métrique des capteurs pour fusionner les données proprioceptives et les perceptions au sein d'une représentation unifiée de l'environnement. Cela est avantageux pour deux raisons. D'une part, ces modèles métriques peuvent, comme nous l'avons vu, être difficiles à obtenir ou s'avérer peu fiables. Se passer de modèle métrique permet ainsi par exemple d'utiliser simplement des images panoramiques pour la reconnaissance de lieux. D'autre part, le fait de ne pas fusionner les deux sources d'informations permet de séparer les influences des erreurs correspondantes. En effet, l'estimation de la position d'objets, lorsque l'on utilise un modèle métrique, dépend à la fois des valeurs mesurées par les capteurs et de la position du robot. Une erreur sur la position d'un objet peut donc provenir des deux sources. Déterminer la contribution de chacune des sources peut être difficile. Dans les cartes topologiques, au contraire, le bruit sur les mesures des capteurs influe principalement sur la reconnaissance des lieux, tandis que le bruit sur les données proprioceptives influe principalement sur la position associée à chaque lieu.

La mémorisation de l'environnement sous forme d'un ensemble de lieux distincts autorise en général une définition des lieux plus directement reliée aux capacités perceptives du robot. En effet, comme les perceptions ne sont pas transformées dans un repère métrique, il n'y a pas de limitation au type de capteurs utilisables (cf. la section 3.2). Cette utilisation directe des perceptions permet un meilleur ancrage dans l'environnement, c'est-à-dire une meilleure mise en relation du robot avec son environnement. Puisque la carte est très proche des données brutes perçues par le robot, il est en général assez simple de comparer et de mémoriser des lieux de l'environnement.

Cette proximité avec les données brutes conduit en général la représentation topologique à utiliser beaucoup moins de concepts de haut niveau que les représentations métriques. La carte topologique reste ainsi proche des possibilités du robot, en mémorisant ses perceptions et ses déplacements possibles, indépendamment de concepts de plus haut niveau tels que des objets ou des obstacles.

La discrétisation de l'environnement correspondant au choix des lieux représentés dans la carte est un autre point fort des cartes topologiques. Cette discrétisation est en effet très utile pour la planification des mouvements du robot, qui se réduit alors à la recherche de chemin dans un graphe. Cette recherche est, en terme de complexité algorithmique, beaucoup plus simple que la recherche d'un chemin dans un espace continu à deux dimensions. Cet avantage est encore plus important lorsque les lieux représentés dans la carte correspondent à des structures

humaines telles que les portes, les couloirs ou les pièces. La discrétisation permet alors de décrire et de résoudre les problèmes de manière naturelle pour les humains, par exemple en donnant l'ordre d'aller au bureau B744, plutôt que de dire d'aller à la position définie par les coordonnées x=354,y=285.

9.1.3 Inconvénients

Comme nous l'avons déjà mentionné, l'utilisation directe des perceptions sans modèle métrique empêche d'estimer ces données pour des positions non visitées. En conséquence, les cartes topologiques nécessitent en général une exploration très complète de l'environnement pour le représenter avec précision. En particulier, tous les lieux intéressants que l'on souhaite trouver dans la carte devront être visités au moins une fois au cours de la construction de la carte, parce qu'ils ne peuvent pas être perçus à distance. Dans le cas où les lieux représentés sont des structures d'assez haut niveau (comme des couloirs ou des pièces), cela n'est pas gênant car ces lieux sont peu nombreux et une exploration exhaustive est donc relativement rapide. En revanche, dans les cartes topologiques représentant des lieux avec une assez grande densité spatiale, cela peut être un inconvénient, car l'exploration complète de l'environnement demandera un temps important.

La reconnaissance des lieux de l'environnement peut également être difficile dans le cas de capteurs très bruités, ou d'environnements très dynamiques. Elle est, de plus, très sensible au problème de *perceptual aliasing* . Ces difficultés conduisent à des problèmes de fausse reconnaissance, c'est-à-dire à la reconnaissance d'un lieu donné alors que le robot se trouve dans un autre lieu. À leur tour, ces fausses reconnaissances conduisent à une mauvaise topologie de la carte et à des liens qui relient des nœuds de la carte qui ne sont pas physiquement reliés dans l'environnement. Ces difficultés rendent problématique la construction de cartes topologiques dans des environnements de grande taille, car la carte résultante risque d'être incohérente. Il devient alors très difficile d'estimer correctement la position du robot au sein de cette carte et de lui ajouter de nouvelles informations sans erreurs.

Comme nous l'avons vu, la représentation de l'environnement peut être assez proche des données brutes des capteurs du robot, ce qui peut être un avantage du point de vue de l'autonomie du robot. Toutefois, cette représentation centrée sur l'individu peut poser des problèmes pour la réutilisation de la carte. En effet, le manque de représentation de l'environnement indépendante de l'individu et l'absence de modèle métrique des capteurs ne permettra pas d'adapter la carte à un robot avec des capteurs légèrement différents. En effet, si l'on dispose d'un tel modèle, l'adaptation se fait simplement au niveau du modèle de capteur, sans modification de la carte elle-même. Cela est plus difficile avec une carte topologique, au sein de laquelle il est quasiment impossible de changer les données recueillies par un capteur pour les transformer en données telles qu'un autre capteur aurait pu les acquérir. De plus, cette représentation centrée sur un individu est moins naturelle pour un opérateur humain, plus habitué aux représentations objectives du type plan d'architecte, ce qui peut être gênant lorsque l'on souhaite une interaction forte entre un opérateur et le robot.

9.1.4 Mise en œuvre

Définition des nœuds

Le choix de ce que vont représenter les nœuds de la carte détermine tout le processus de construction de la carte topologique. Ce choix est lié aux capacités de perception dont on a doté le robot, lequel devra être capable de détecter les lieux en question. La localisation et la mise à jour de la carte se feront chaque fois qu'un tel lieu aura été détecté. La détection de ces lieux peut être contrainte par les choix d'un opérateur humain ou être complètement autonome.

Nœuds définis par le concepteur

La première possibilité est de définir directement quels lieux doivent être détectés par le robot et comment ils doivent l'être. Des procédures sont alors écrites qui permettent de détecter spécifiquement chaque type de lieu. Le choix le plus courant est l'utilisation de couloirs, de portes et d'intersections [33, 69, 83, 125]. Lorsque ce choix est fait, un très petit nombre de lieux différents peuvent être détecté, ce qui rend le problème du *perceptual aliasing* omniprésent. Les systèmes concernés dépendent donc en général fortement des données proprioceptives pour parvenir à utiliser ces représentations.

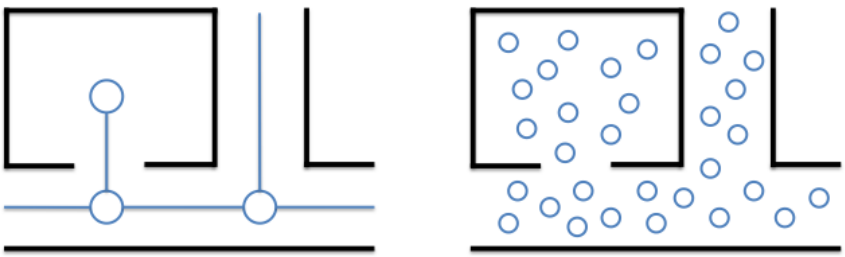


Figure 9.2: Exemples de cartes topologiques avec des noeuds définis à des positions canoniques et des liens métriques (à gauche) et avec des noeuds denses et des positions métriques associées (à droite).

Nœuds définis à des positions canoniques

Plutôt que de définir complètement les lieux que peut détecter le robot, le concepteur peut simplement définir dans quels types de situations le robot peut enregistrer un lieu, laissant le soin de définir chaque lieu précisément au moment de la découverte du lieu (figure 9.2, à gauche). Par exemple, le concepteur peut doter le robot de la capacité générale de détecter des portes. Lorsque le robot détectera une porte, il enregistrera un nouveau nœud dans la carte, mais ce nœud sera défini par la situation précise dans laquelle il se trouve quand il rencontre cette porte. Il pourra, par exemple, enregistrer la couleur de la porte, le numéro qui est inscrit dessus ou une image de l'environnement vu depuis cette position. Cette méthode de définition des nœuds a été proposée par Kuipers et Byun [82] sous le nom de *distinctive places*, puis utilisée sous une forme différente par Engelson et McDermott [43] et par Kortenkamp et Weymouth [80] sous le nom de *gateways*.

Nœuds définis de manière non supervisée

La troisième méthode pour définir les nœuds d'un carte topologique consiste à les définir comme des zones où les perceptions sont approximativement constantes (figure 9.2, à droite). Cela est obtenu en général par la catégorisation non supervisée des perceptions [9, 36, 51, 54, 84, 93, 98, 105, 141, 142]. Ces perceptions sont donc regroupées en catégories contenant des données similaires, sans que ces catégories soient spécifiées par un concepteur humain. Chaque catégorie correspond alors à un ou plusieurs nœuds de la carte. Le nœud correspondant à une catégorie étant unique dans le cas où il n'y a pas de perceptual aliasing. Cette méthode est bien adaptée à des robots autonomes car la catégorisation ne nécessite aucun superviseur, ni aucune définition a priori des données correspondant à un nœud. A ce titre, elle est utilisée dans tous les systèmes de navigation qui s'inspirent des comportements de navigation des animaux [6, 12, 25, 124, 137].

Pour mettre en œuvre une telle approche, il faut définir un critère qui permette de décider quand un nouveau lieu a été atteint. Le choix le plus évident est de comparer constamment la situation courante à celle du précédent nœud reconnu. Lorsque la différence est suffisamment importante, on considère qu'un nouveau lieu a été atteint. Cette méthode est utilisée par certains modèles [51, 54, 84, 98, 105], mais requiert que les perceptions soient comparées en temps réel, ce qui peut-être difficile pour certains capteurs (les caméras, par exemple). D'autres modèles considèrent donc plus simplement qu'un nouveau nœud a été atteint lorsque la distance parcourue depuis la dernière reconnaissance est assez grande [6, 137, 142, 145].

Définition des arêtes

Les arêtes reliant les nœuds permettent de mémoriser des données sur les relations de voisinage entre lieux représentés par les nœuds. Ces données sont en général obtenues grâce aux informations proprioceptives. Elles peuvent être plus ou moins précises et représentées sous diverses formes.

Relation d'adjacence

La première information que porte une arête est une information d'adjacence entre les deux lieux représentés par les nœuds qu'elle connecte. Cette relation peut être bidirectionnelle ou non. L'existence d'une arête signifie donc que le robot peut passer directement d'un lieu à l'autre, sans passer par un lieu intermédiaire. Si certains modèles ne mémorisent que cette information d'adjacence [51, 55, 69, 79, 106, 141], cette information est prise en compte dans tous les modèles, même si des informations supplémentaires sont enregistrées dans les arêtes.

Relations métriques

Des informations métriques sur la position relative des lieux peuvent être mémorisées dans les arêtes. Ces informations portent en général sur la position relative des lieux reliés par l'arête [43, 66, 82, 83, 105, 125, 127, 142] (figure 9.2, à gauche). Elles sont fournies et quantifiées par les données proprioceptives lorsque le robot se déplace d'un lieu à l'autre. Cette méthode présente l'avantage de limiter l'accumulation de l'erreur des données proprioceptives, puisque ces données ne sont utilisées que sur la distance reliant un nœud à un autre. Cette distance

est en général assez courte pour éviter une accumulation d'erreurs trop importante. Les cartes topologiques utilisant de telles informations métriques sont appelées par certains auteurs cartes *diktiométriques* [43], ou carte topo-métriques [13].

Association de positions aux nœuds

Dans le but d'intégrer les données proprioceptives à une carte topologique, il est également possible d'associer une position à chacun des nœuds (figure 9.2, à droite). Cette position se mesure dans l'espace dans lequel s'expriment les données proprioceptives et correspond à la position des différents lieux dans l'environnement. Ce type de carte se rapproche fortement des cartes métriques, à la différence que seuls les lieux visités par le robot, et non les objets perçus par le robot, sont mémorisés. L'inconvénient, par rapport à l'approche précédente, est qu'il est nécessaire de corriger les informations proprioceptives car elles ne sont plus utilisées localement. Chaque nœud ayant une position dans un cadre de référence global, il est possible de se contenter de cette information, sans ajouter de liens entre les nœuds [108, 6, 12]. Toutefois, certains modèles utilisent également des liens pour mémoriser l'information d'adjacence [98, 137, 84, 146, 36, 142, 33]. Comme l'information de position de chaque nœud est absolue, ce type de carte peut être appelé carte *diktiométrique absolue*.

Relation implicite

Dans certains cas, il est possible de retrouver les relations de position entre les lieux au vu des seules perceptions qui les représentent. Cela est possible, par exemple, lorsque les lieux sont définis par la configuration d'amers distants qui peuvent être perçus par le robot lorsqu'il se trouve à cette position. Un certain nombre d'amers communs, visibles depuis deux lieux différents permettront d'avoir des informations sur la position relative de ces lieux. L'existence d'amers communs peut donc être utilisée comme lien implicite [93, 124, 25].

9.2 Cartes métriques

9.2.1 Description

Dans une carte métrique, l'environnement est représenté par un ensemble d'objets auxquels sont associées des positions dans un espace métrique, généralement en deux dimensions. Cet espace est, la plupart du temps, celui dans lequel s'exprime la position du robot estimée par les données proprioceptives. Les perceptions permettent, en utilisant un modèle métrique des capteurs, de détecter ces objets et d'estimer leur position par rapport au robot. La position de ces objets dans l'environnement est alors calculée en utilisant la position estimée du robot. La fusion des deux sources d'information au sein d'un même cadre de représentation est caractéristique des cartes métriques.

Les objets mémorisés dans la carte peuvent être très divers et seront détaillés dans la suite de cette section. Dans certaines implantations, ces objets correspondent aux obstacles que le robot pourra rencontrer dans son environnement. La carte de l'environnement correspond alors directement à l'espace libre, c'est-à-dire à l'espace dans lequel le robot peut se déplacer.

9.2.2 Avantages

L'avantage principal des cartes métriques est de permettre de représenter l'ensemble de l'environnement, et non un petit sous-ensemble de lieux comme le font les cartes topologiques. Cette représentation complète permet ainsi d'estimer avec précision et de manière continue la position du robot sur l'ensemble de son environnement. De plus, cette représentation complète ne se limite pas aux positions physiquement explorées, mais s'étend à toutes les zones que le robot a pu percevoir depuis les lieux qu'il a visités. Cette propriété peut permettre la construction d'une carte plus exhaustive de l'environnement en un temps plus court.

Un autre avantage des cartes métriques est lié au fait que la position du robot est définie de manière non ambiguë par ses coordonnées au sein de l'espace dans lequel la carte est représentée. Il s'ensuit une utilisation simple et directe de toutes les informations métriques fournies par les données proprioceptives ou les perceptions. Cela est un avantage par rapport aux cartes topologiques où les positions possibles du robot sont limitées aux nœuds présents dans la carte et sont donc relativement imprécises. Une telle représentation, dans laquelle chaque nœud peut couvrir une zone étendue de l'environnement, rend plus difficile l'utilisation des données métriques car la position relative de deux zones est moins bien définie.

La représentation de l'environnement indépendante de l'individu utilisée dans les cartes métriques apporte un certain nombre d'avantages supplémentaires. Comme nous l'avons mentionné à propos des cartes topologiques, une telle représentation permet une réutilisation plus facile d'une carte sur des robots différents, équipés de capteurs différents, l'essentiel de l'adaptation se déroulant au niveau des modèles métriques des capteurs. Ce type de représentation est aussi facilement interprétable par un humain, ce qui peut être important dans le cas où il doit intervenir dans les déplacements du robot.

Cette représentation peut de plus utiliser des concepts de plus haut niveau, tels que des objets, des obstacles ou des murs. Cela permet un apport de connaissance plus facile de la part des humains, par exemple pour imposer que les murs détectés soient perpendiculaires ou parallèles.

9.2.3 Inconvénients

Lors de l'utilisation de cartes métriques, les données proprioceptives ont en général une importance supérieure à celle qu'elles ont dans l'utilisation d'un carte topologique. Par conséquent, une odométrie plus fiable peut être requise. Le niveau de fiabilité nécessaire peut être atteint en imposant des limitations sur l'environnement du robot. Par exemple, il est possible d'imposer que tous les couloirs soient orthogonaux, afin de pouvoir corriger efficacement la dérive de l'estimation de la position.

Comme nous l'avons mentionné dans la section 3.2, un modèle métrique des capteurs peut être difficile à obtenir. Les problèmes liés au bruit des capteurs et à la difficulté de modéliser de manière fiable leur relation avec l'environnement constituent donc un point faible des cartes métriques.

Enfin, le calcul de chemin au sein des cartes métriques peut être plus complexe, car la planification se déroule dans un espace continu et non dans un espace préalablement discrétisé,

comme c'est le cas pour les cartes topologiques. De nombreuses méthodes recourent d'ailleurs à l'extraction d'une carte topologique depuis la carte métrique pour réaliser cette opération de planification [87].

9.2.4 Mise en œuvre

Deux méthodes principales sont utilisées pour mémoriser des informations sous forme de carte métrique. La première méthode consiste à extraire explicitement des objets des perceptions et à les enregistrer dans la carte avec leur position estimée. Les objets peuvent être de types très variés et se situer à différents niveaux d'abstraction. La seconde méthode s'attache à représenter directement l'espace libre accessible au robot et les zones d'obstacles qu'il ne peut pas franchir, sans avoir recours à l'identification d'objets individuels.

Représentation d'objets

Points

Les objets les plus simples qui peuvent être utilisés sont des points [93, 111, 44] que l'on appelle dans ce cas *amers* (terme de marine désignant des points de repère remarquables). Ces points correspondent à des objets de l'environnement de taille suffisamment petite, ou situés suffisamment loin du robot, pour pouvoir être considérés comme ponctuels (figure 9.3, à gauche). Ils possèdent l'inconvénient que la perception d'un point de l'environnement ne suffit pas à déterminer de manière unique la position du robot. Ce type de points de repère est par conséquent relativement pauvre et contraint à la détection de plusieurs objets pour assurer une localisation précise. De plus, reconnaître un tel point de manière non ambiguë est souvent difficile et requiert une bonne capacité de discrimination de la part des capteurs. Cependant, certains modèles ne requièrent pas cette identification et utilisent des points indistinguables.

Certains modèles ont recours à des ensembles de points disséminés sur la surface des objets de l'environnement [94, 63, 134] (figure 9.3, à droite). Ces points sont en général obtenus par des télémètres laser, qui permettent d'en recueillir un grand nombre avec une résolution spatiale élevée. Les objets sont ainsi définis par la configuration d'ensembles de points, et non plus par des points uniques. Cette méthode présente donc l'avantage de ne pas recourir à l'identification individuelle de chaque point.

Points orientés

Afin d'obtenir plus d'information sur la position du robot par la perception d'un seul objet, il est possible de doter chaque objet ponctuel d'une orientation. La perception d'un tel point orienté permet alors d'estimer la position du robot de manière unique. Un tel type de point peut correspondre à un point de référence sur un objet non ponctuel de l'environnement [68, 129], par exemple l'angle d'un obstacle, perçu grâce à un télémètre laser [18].

Frontière des objets

Les frontières des différents objets et obstacles de l'environnement peuvent être directement représentées par des objets géométriques de plus haut niveau que des points. Des lignes ou

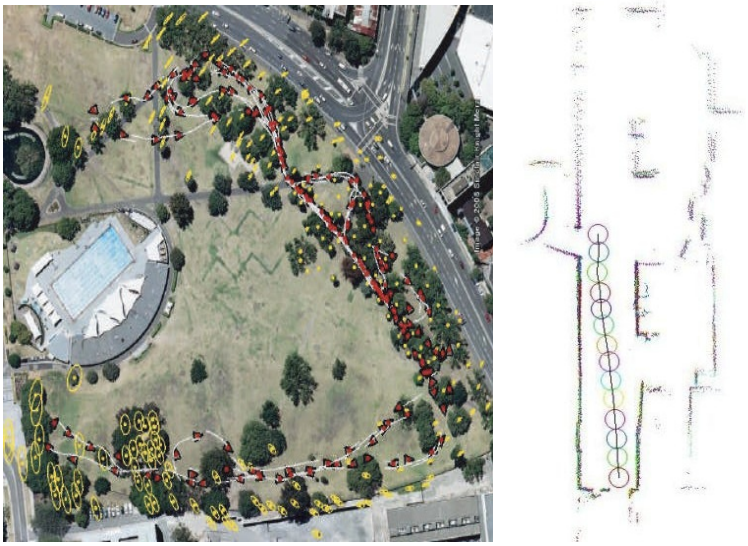


Figure 9.3: Exemples de cartes métriques à base de points isolés (on parle alors d'amers, à gauche, repris de [109]) et de carte à base de scans lasers (chaque cercle est le centre d'un scan laser regroupant les points de la même couleur, à droite).

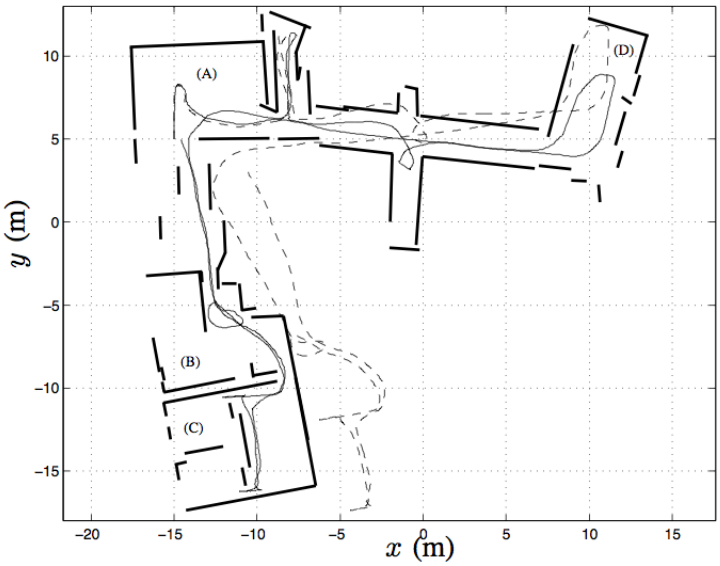


Figure 9.4: Exemple de carte à base de segments détectés par un télémètre laser (repris de [52]).

des polygones sont très souvent utilisés (figure 9.4). Ces objets sont extraits d'ensemble de points perçus par des capteurs à ultrasons [39, 53] ou des télémètres laser [103, 41, 28, 52]. Des cylindres et des plans, détectés par des capteurs à ultrasons sont aussi utilisés [92], ainsi que des structures de plus haut niveau, comme des plans en trois dimension, détectés par stéréo-vision [8].

Représentation de l'incertitude

Dans la plupart des systèmes, la manière dont est représentée et gérée l'incertitude est cruciale. L'incertitude concernant les objets mémorisés dans la carte est de deux types. Le premier concerne l'incertitude sur les paramètres des objets, par exemple sur leur position dans l'environnement. Ce type d'incertitude provient des erreurs de localisation du robot lors de la perception d'un objet, ou d'un bruit au niveau du capteur. Il est, dans la majorité des cas, représenté de manière probabiliste, notamment par la variance de paramètres considérés [129, 8, 103, 92, 68, 44, 28]. Toutefois, d'autres méthodes peuvent être utilisées, par exemple des intervalles [43] ou des ensembles flous [53].

Le second type d'incertitude se place à un niveau plus fondamental. Il porte sur la qualité de la correspondance entre la carte et l'environnement. Il mesure avec quelle confiance un objet présent dans la carte correspond effectivement à un objet de l'environnement. En effet, il peut arriver que des erreurs de perception fassent apparaître des objets qui n'existent pas dans l'environnement. Cette incertitude est caractéristique des environnements dynamiques, dans lesquels des objets sont susceptibles de se déplacer, d'apparaître ou de disparaître. Elle est gérée, pour une grande partie, au niveau des capteurs, les procédures permettant la détection d'objet à mémoriser étant conçues pour ignorer au maximum les éléments instables de l'environnement. Au niveau de la carte, la plupart des modèles traitent ce problème au moment de la mise à jour. Il est par exemple possible de supprimer les objets qui auraient dû être perçus, mais qui restent introuvables par le robot. Certain modèles toutefois modélisent explicitement cette incertitude au moyen d'un paramètre de crédibilité [92]. Ce paramètre permet une plus grande tolérance aux accidents de perception en mesurant la fiabilité d'objets comme point de repère.

Représentation de l'espace libre

Un des premiers modèles pour ce type de représentation est celui de la *grille d'occupation* [101, 131, 147]. Dans ce modèle, l'environnement est entièrement discrétisé suivant une grille régulière avec une résolution spatiale très fine (cf. figure 9.5). Une probabilité d'occupation est associée à chaque élément de cette grille. Cette probabilité mesure la confiance dans le fait que l'espace correspondant dans l'environnement est effectivement occupé par un obstacle. L'avantage d'une telle représentation est qu'elle utilise directement les valeurs des capteurs de distance afin de mettre à jour les probabilités d'occupation des cellules. Elle permet donc de supprimer la phase d'extraction d'objets qui est souvent coûteuse en temps de calcul et source de bruit.

Les grilles d'occupation utilisent cependant une quantité de mémoire importante, qui croît proportionnellement à la surface de l'environnement. Pour s'affranchir de ce problème, certains modèles font appel à des discrétisations irrégulières de l'espace [5] ou à des discrétisations hiérarchiques. De telles discrétisations permettent de s'adapter à la complexité de l'environnement, en

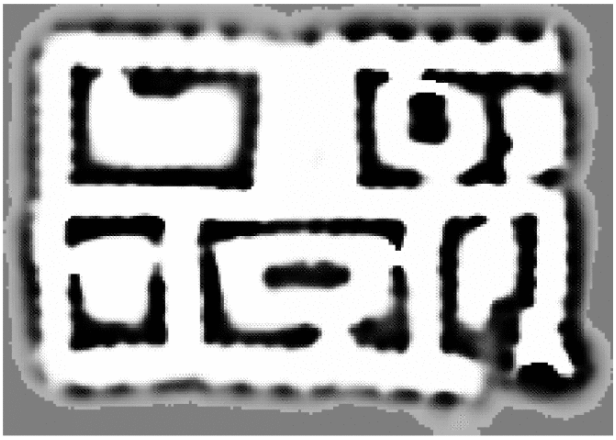


Figure 9.5: Un exemple de grille d'occupation utilisée pour représenter un environnement. Les zones sombres indiquent une forte probabilité de présence d'un obstacle (Repris de [131]).

représentant de manière grossière les grands espaces libres et plus finement les contours des obstacles.

9.3 Représentations hybrides et hiérarchiques

Au delà des deux grandes catégories topologiques et métriques, il existe toute une gamme de représentations hybrides mélangeant ces deux approches.

Nous avons déjà mentionné un premier type de représentations pouvant être considérées comme hybrides : les représentations topo-métriques qui sont des cartes topologiques contenant des informations métriques sur les arêtes du graphe (figure 9.2, à gauche). Ce type de représentation est par exemple bien adapté pour construire des cartes à partir de la vision : chaque nœud du graphe peut être associé à une image, relié à ses voisins par des informations obtenues par l'odométrie du robot ou par odométrie visuelle [13].

Les nœuds de la carte topo-métrique peuvent aussi contenir des informations plus complexes, telles que des cartes métriques locales (par exemple [136] et figure 9.6). L'intérêt de ces représentations est de contenir des cartes métriques précises pour des zones plus simples à cartographier et dans lesquelles la navigation devra être précise (notamment les pièces) et de ne représenter les couloirs (plus difficile à cartographier du fait de leur taille et du manque de points de repères) que comme liens topologiques entre pièces. Ces cartes présentent ainsi l'avantage de ne pas demander une localisation très précise sur une zone très étendue.

Enfin, au delà des cartes brutes construites par les méthodes que nous allons présenter dans ce cours, il devient de plus en plus important d'introduire différents niveaux d'information dans les cartes pour s'adapter aux différentes tâches d'un robot de service par exemple. En particulier, il commence à apparaitre des cartes contenant des informations *sémantiques*. Ces informations se situent à un niveau plus haut que l'espace libre ou les obstacles représenté dans les cartes brutes et peuvent concerner par exemple les pièces détectées dans l'environnement, le type de ces pièces (cuisine, salon...) ou les objets présents dans l'environnement [97, 72, 112]. Ces

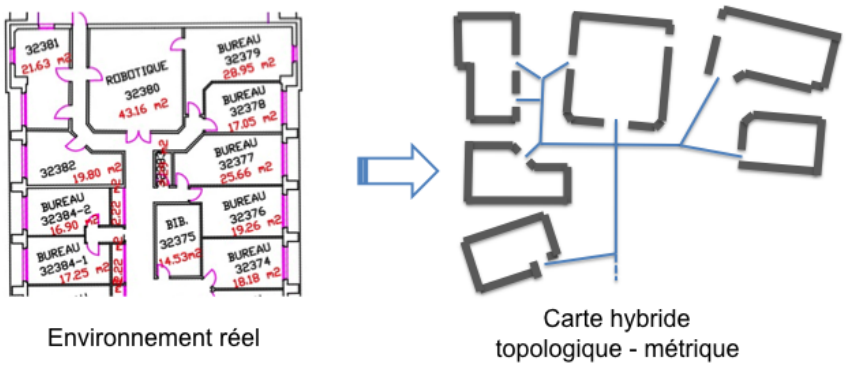


Figure 9.6: Un exemple de carte hybride mélangeant des cartes métriques locales pour les pièces et une carte topologique globale. Les positions relatives des cartes locales sont alors connues de manière imprécise car les couloirs ne sont pas cartographiés précisément, mais simplement suivis pour aller d'une porte à une autre.

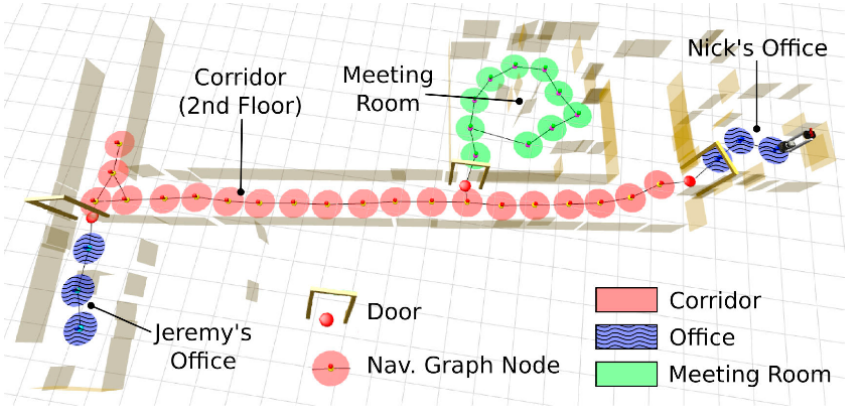


Figure 9.7: Un exemple de carte hiérarchique se basant sur une carte métrique dont est extrait une carte topologique. Les nœuds de cette carte topologique sont ensuite associés à une catégorie (repris de [112]).

types de représentations sont souvent hiérarchiques : sur la base d'une carte métrique, une carte topologique est construite, puis les nœuds sont classifiés suivant leur types et permettent de mémoriser les objets associés (figure 9.7). Les informations sémantiques peuvent être utiles dans de nombreuses situations, par exemple pour chercher un objet (une assiette sera plus probablement dans la cuisine), ou pour la navigation : la connaissance du type d'obstacle peut ainsi permettre d'envisager de pousser certains objets tels que les chaises.

Nous ne détaillerons pas dans ce cours la construction de ce types de cartes car elle font en général appel à l'intégration de très nombreuses méthodes différentes et sont l'objet de recherches très actives.

Chapter 10

Localisation

Ce chapitre présente les principales méthodes de localisation. On pourra trouver une description succincte d'un grand nombre d'autres méthodes dans [45]. La présentation est réalisée selon une classification personnelle en trois grandes catégories qui me semble utile à la compréhension, mais qui n'est pas forcement utilisée couramment dans la littérature.

10.1 Différentes capacités de localisation

Il existe trois types de capacités regroupées sous le terme «localisation», de complexités différentes.

- Le *suivi de position*, est la capacité de mettre à jour une estimation existante de la position au vu de données proprioceptives ou de perceptions nouvellement acquises. Dans le cas des données proprioceptives, cette mise à jour concerne un déplacement du robot et va en général diminuer la précision de l'estimation courante de la position, à cause de l'erreur sur la mesure. Dans le cas de perceptions, au contraire, cette mise à jour va en général permettre d'améliorer cette estimation grâce au lien avec l'environnement fourni par ces données. L'utilisation de cet ancrage dans l'environnement est fondamental pour assurer que l'estimation de la position reflète correctement la position du robot dans l'environnement réel. Cette mise à jour intégrant les deux types de données permet de combiner les avantages inhérents aux deux types d'information afin d'estimer au mieux la position du robot. En pratique, toutefois, le suivi de position est problématique car il repose sur une estimation initiale de la position qui doit souvent être fournie par une source extérieure. De plus, si la position estimée s'écarte trop de la position réelle du robot, il peut très bien être impossible de parvenir à corriger l'erreur et de retrouver la position réelle, ce qui conduit à une dérive de l'algorithme.

- la *localisation globale*, est plus générale et permet de retrouver la position du robot sans qu'aucune estimation initiale ne soit fournie. Cette capacité est très importante du point de vue de l'autonomie, car elle permet au robot de trouver sa position initiale, dans toutes les conditions, sans intervention extérieure. Elle permet, par exemple, de couper l'alimentation

d'un robot à des fins de maintenance, puis de remettre ce robot dans une position quelconque de l'environnement sans se soucier d'initialiser correctement son estimation de la position.

- la troisième capacité est la capacité à retrouver la position d'un robot *kidnappé*, c'est à dire d'un robot dont on a une estimation de la position, mais dont l'estimation est fausse car il a été déplacé, sans que le système de localisation n'en soit informé. Par rapport à la localisation globale, ce cas présente la difficulté supplémentaire de parvenir à détecter que la position actuellement suivie n'est plus correcte. Cette phase est délicate car il faut distinguer entre les cas où les perceptions sont simplement temporairement bruitées, sans que le robot ait été déplacé, et les cas où le robot a réellement été déplacé.

Les capacités de suivi de position et de localisation globale ont des propriétés duales. Comme le note Piasecki [110], dans le contexte d'une carte métrique, le suivi de position est une méthode locale, continue, qui effectue régulièrement de petites corrections à l'estimation de la position du robot. Cette méthode effectue de telles corrections en se basant sur des objets de l'environnement et la manière dont ils ont été perçus par le robot. L'identification de ces objets est de plus simplifiée grâce à l'estimation initiale de la position qui permet, en cas de perceptual aliasing, de décider quel est l'objet qui a été perçu parmi les différents objets correspondant aux perceptions.

Au contraire, la localisation globale est une méthode globale, discontinue, qui effectue exceptionnellement des corrections de grande ampleur de la position estimée. Sa première tâche, avant d'estimer une position, est de déterminer à quels objets de l'environnement correspond chacune des perceptions du robot. Cette tâche peut être très simple dans le cas où la carte ne contient que des amers différents, mais est en général assez complexe car plusieurs amers sont identiques à cause du *perceptual aliasing*.

Comme nous l'avons souligné, le suivi de position permet de fusionner et de tirer parti des informations proprioceptives et des perceptions disponibles pour le robot. Cette méthode ne conduit toutefois qu'à une estimation qui est localement la meilleure approximation possible de la position. En effet, la recherche est contrainte par l'estimation précédente de cette position. La position estimée sera donc celle qui est la plus en accord avec les données recueillies, *dans le voisinage de cette estimation précédente*. L'estimation résultante peut donc très bien ne pas correspondre à la position qui, sur l'ensemble de la carte, correspond le mieux aux données.

En principe, la localisation globale permet une telle estimation optimale. Au niveau de l'utilisation des données disponibles pour le robot, il existe de nouveau deux classes de méthodes de localisation globale :

- La première, qui ne fonctionne que dans des environnements où il n'existe aucun *perceptual aliasing*, fait appel uniquement aux perceptions disponibles en une position donnée. Nous l'appellerons dans ce cours *inférence directe de position*.

- La seconde, qui fonctionne dans tous les environnements, fusionne au contraire les informations proprioceptives et les perceptions, comme le fait le suivi de position. Toutefois, au

lieu de restreindre la recherche par une estimation précédente de la position, elle estime parmi toutes les positions possibles au sein de la carte celle qui correspond le mieux aux données présentes et passées recueillies par le robot. Les méthodes de cette catégorie reposent, d'une façon ou d'une autre, sur le *suivi de plusieurs hypothèses* de position, ce qui permet de généraliser le suivi de position en utilisant de manière plus efficace les informations disponibles.

Dans la suite de ce chapitre, nous allons présenter plus précisément les différentes méthodes de localisation. Nous allons d'abord voir comment il est possible d'estimer la position d'un robot au vu des seules perceptions (section 10.2). Puis, dans le cas de systèmes perceptifs soumis au *perceptual aliasing*, nous verrons comment il est possible d'intégrer les informations proprioceptives, de manière locale (section 10.3) puis globale (section 10.4), afin de lever les ambiguïtés restantes.

10.2 Estimation de la position par les perceptions

Dans cette section, nous résumons les différentes méthodes qui peuvent être utilisées pour estimer la position d'un robot à l'aide des seules perceptions. Dans le cas où l'environnement est exempt de *perceptual aliasing* , cette étape suffit à déterminer la position du robot de manière unique. Cette méthode est alors la première méthode de localisation globale mentionnée précédemment. Dans le cas où le *perceptual aliasing* est présent, ces méthodes sont également utilisées mais elles serviront à repérer plusieurs positions possibles pour le robot au sein de l'environnement. Le suivi de position ou la seconde méthode de localisation globale qui seront présentés dans les paragraphes suivants doivent alors être utilisés en sus pour sélectionner la position correcte.

10.2.1 Cartes topologiques

Dans le cas des cartes topologiques, estimer la position à partir des seules perceptions est extrêmement simple. En effet, parmi tous les lieux représentés dans la carte, la position du robot est celle d'un des nœuds qui correspond le mieux aux perceptions courantes. La recherche de ces nœuds passe donc par la comparaison des perceptions du robot avec les perceptions mémorisées dans chacun des nœuds de la carte. Les nœuds qui sont identiques ou suffisamment similaires sont alors reconnus comme positions possibles du robot.

En l'absence de *perceptual aliasing*, tous les nœuds de la carte correspondent à des situations différentes. Cette étape est alors suffisante pour la localisation complète du robot car le nœud reconnu est unique. Différents systèmes perceptifs ont été utilisés pour implanter de tels modèles. Certains auteurs utilisent des images panoramiques de l'environnement pour définir les nœuds de la carte [80, 51]. D'autre modèles utilisent les directions ou les distances d'amers ponctuels tous discernables, soit en simulation [25, 93, 139, 124], soit sur des robots réels [9, 54].

L'utilisation d'images panoramiques permet par exemple de réaliser un système de localisation à partir d'une méthode d'indexation d'images [128]. Il suffit en effet d'avoir une base de données indexant les images des différents nœuds de la carte, puis, pour se localiser, de rechercher dans cette base l'image la plus proche de l'image courante. Cette image nous donnera le nœud correspondant à la position courante. L'indexation peut par exemple utiliser une analyse en composantes principales[1] qui va déterminer une base sur laquelle il sera possible de projeter chaque image. Les coordonnées de chaque images dans cette base fournissent ainsi un représentation de faible dimension de chaque nœud de la carte. Pour la localisation, il suffit de projeter l'image courante sur la base et de chercher l'image ayant les coordonnées les plus proches.

Lorsqu'une position dans un espace métrique est associée à chacun des nœuds de la carte, la localisation permet en outre de déterminer la position métrique du robot. Cette position peut simplement être la position du nœud reconnu, mais il est souvent possible d'obtenir une précision supplémentaire. En effet, au lieu de tenir simplement compte du nœud le plus conforme aux perceptions courantes, il est possible de tenir compte de chacun des nœuds, selon son degré de similarité avec ces perceptions. La méthode mise en œuvre dans de tels modèles pour réaliser cette estimation de position s'appelle dans la terminologie des neurosciences le *codage par population de vecteurs* [56]. Cette méthode consiste à estimer la position du robot par la moyenne des positions des différents nœuds, pondérées par le degré de similarité de chaque nœud avec les perceptions du robot. Cette méthode donne une estimation précise de la position du robot, mais suppose une relative continuité de l'environnement. Elle suppose en effet que des lieux similaires seront proches les uns des autres pour que la moyenne des positions ait un sens. Les perceptions doivent donc varier de manière relativement continue avec la position.

Lorsque les modèles permettent la gestion du *perceptual aliasing* (par une des méthodes décrites dans les paragraphes suivants), les lieux peuvent également être définis par des images panoramiques de leur environnement [6, 40, 66, 141, 113, 142], ou par la configuration des positions d'amers distants [12, 137]. Mais, puisque le *perceptual aliasing* sera géré par ailleurs, des définitions plus simples des nœuds peuvent également être adoptées, au prix d'une moins grande discrimination. Certains modèles utilisent ainsi les valeurs brutes de capteurs de distance [105, 98, 82, 69, 66, 108], ou la configuration des murs autour du robot afin de détecter des angles de couloirs ou des embranchements [83, 33, 125, 27, 106, 127, 130].

10.2.2 Cartes métriques

Dans le cas des cartes métriques, diverses méthodes d'estimation de la position existent. Lorsque les perceptions sont constituées d'amers ponctuels, une méthode de triangulation peut être utilisée [14, 59, 93, 96]. Cette méthode repose sur la mesure de la direction et de la distance d'amers ponctuels connus. La perception de trois amers de ce type permet en effet de définir la position du robot de manière unique. Un simple calcul mathématique permet donc d'estimer cette position à partir des positions des amers. Ce calcul peut également être approché par des réseaux de neurones, [111], ou par des méthodes heuristiques qui permettent une meilleure résistance au bruit [144]. Lorsque cette méthode est utilisée avec des cartes ne comportant pas de *perceptual*

[1] voir par exemple http://fr.wikipedia.org/wiki/Analyse_en_composantes_principales

aliasing, chaque amer est unique et cette méthode permet d'estimer directement de manière non ambiguë la position du robot. En cas de *perceptual aliasing*, certains amers ne peuvent être distingués et il faut tenir compte de l'estimation précédente de la position afin de pouvoir identifier correctement les différents amers et estimer correctement la position.

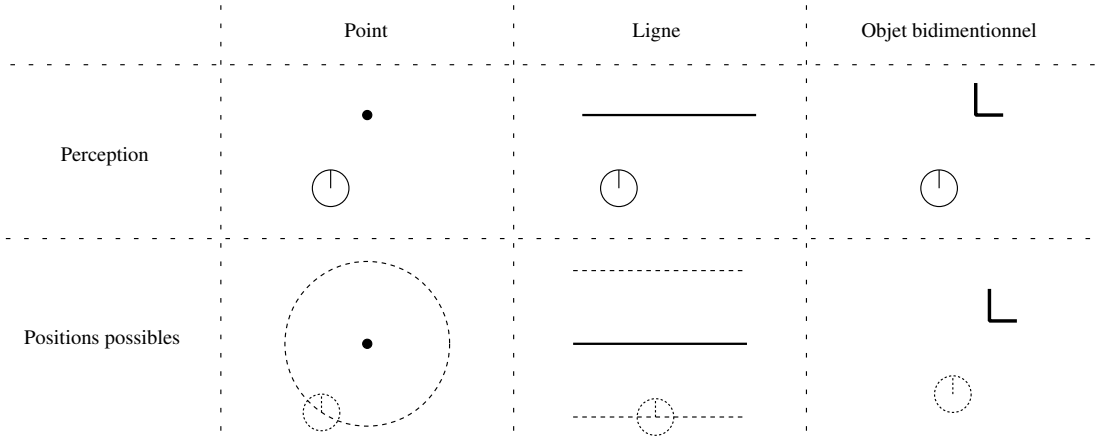


Figure 10.1: Les amers que peut repérer un robot fournissent plus ou moins d'information sur sa position en 2 dimensions. La détection d'un amer ponctuel permet de savoir que le robot se trouve sur un cercle entourant l'amer. Un amer rectiligne permet de connaître la distance du robot perpendiculairement à cet amer, mais pas sa position le long de cet amer. Enfin, un amer ayant une étendue spatiale en deux dimensions permet de définir la position du robot de manière unique.

Certains types d'objets fournissent plus d'information que des amers ponctuels, sans toutefois permettre une estimation non ambiguë de la position. Par exemple, c'est le cas des murs dont la perception fournit une information sur la distance du robot à ce mur, mais pas sur sa position le long de ce mur (cf. figure 10.1). Certains modèles utilisent de tels types d'objets, qui permettent d'affiner une estimation précédente de la position, mais pas d'estimer directement cette position [28, 73, 117].

Lorsque les objets mémorisés dans la carte ont une certaine étendue spatiale en deux dimensions, il est par contre possible d'utiliser la perception d'un seul objet afin d'estimer directement la position du robot. Les amers utilisés peuvent alors être des objets tridimensionnels détectés par une caméra [126], les angles des obstacles détectés par un télémètre laser [7, 18, 57, 73] ou un capteur à ultrason [92, 117], des segments détectés en utilisant une caméra [8] ou un télémètre laser [28, 31, 103].

D'autre modèles, enfin, n'estiment pas directement la position du robot au vu des perceptions, mais reposent sur la comparaison d'une carte métrique locale avec la carte métrique globale (cf. figure 10.2). La carte métrique locale est construite soit à partir des seules perceptions courantes, soit à partir des données proprioceptives et des perceptions recueillies sur un court laps de temps. Le problème est alors de trouver la portion de carte globale qui correspond le mieux à la carte locale. Cette méthode est très souvent utilisée avec les grilles d'occupation [107, 119, 122, 131], ainsi qu'avec des données brutes de télémètres laser [94, 65, 41]. Le *polygone de visibilité*, qui

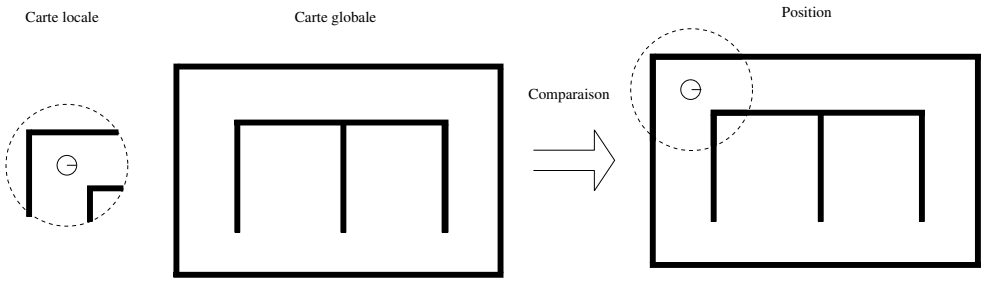


Figure 10.2: Pour estimer la position d'un robot, il est possible de construire une carte locale représentant l'environnement proche de la position courante. La comparaison de cette carte locale et de la carte globale de l'environnement permet alors de trouver la position.

entoure la zone d'espace libre visible depuis la position courante du robot peut aussi être utilisé [62, 75]. Comme nous le verrons dans la section suivante, ces méthodes sont souvent utilisées sur un espace de recherche restreint par une estimation initiale de la position. Elles peuvent cependant être utilisées pour la localisation globale [107, 62, 75].

10.2.3 Corrélation de cartes

Dans cette section, nous allons détailler une méthode de corrélation de cartes qui permet de chercher, pour deux cartes de dimensions réduites, la transformation (translation+rotation) qui permet la meilleure superposition. Cette transformation permet alors de corriger l'estimation de position du robot. Cette méthode peut être utilisée avec différents types de cartes, dont les grilles d'occupation, mais donne des résultats particulièrement efficaces avec des données issues d'un télémètre laser (Figure 10.3).

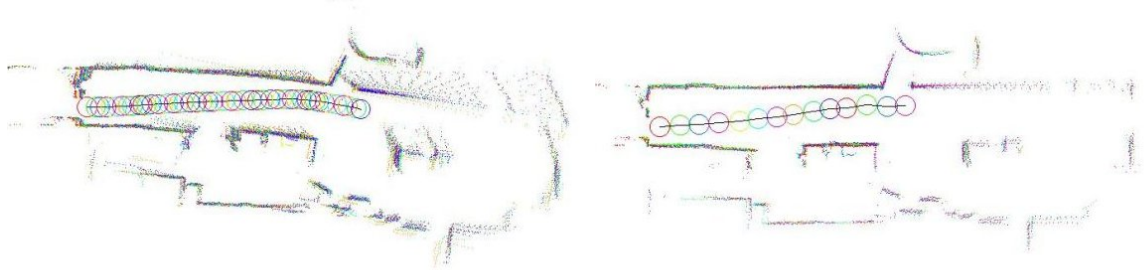


Figure 10.3: Exemple de résultat de corrélation de mesures laser. La partie gauche montre les mesures affichées à la position mesurée par l'odométrie. La partie droite montre le résultat de la corrélation.

Il existe de nombreuses méthodes de mise en correspondance de scans lasers, telle que la méthode Iterated Closest Point (ICP) [95] ou des méthodes basées sur l'approche RANdom SAmple Consensus (RANSAC) [76] que nous ne présenterons pas toutes ici. La méthode que nous avons choisi de présenter [116] est une méthode simple qui est relativement robuste et

résistante au bruit. Elle est basée sur les histogrammes des directions des tangentes au scan laser.

La première étape consiste à calculer pour chaque point du scan la droite tangente en utilisant la méthode des moindres carrés. On cherche pour cela la droite qui fournit la plus faible erreur quadratique sur un ensemble comprenant quelques points avant et quelques points après le point courant (Figure 10.4).

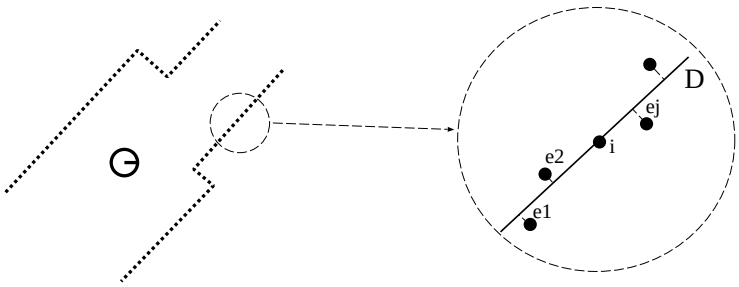


Figure 10.4: Illustration de la méthode de calcul de la tangente en chaque point du scan. Pour chaque point i, on cherche les paramètres de la droite qui donnent la plus faible erreur quadratique $\sum e_j^2$.

On construit ensuite l'histogramme des directions de ces tangentes, après avoir filtré les points pour lesquels la qualité de l'approximation des tangentes est trop faible. Cet histogramme contient alors des pics lorsque des ensembles de points correspondent à un mur rectiligne de l'environnement. Après avoir construit ces histogrammes de direction pour les deux scans, on cherche le décalage de ces histogrammes qui fournit la meilleure corrélation. Dans l'hypothèse où les deux scans ont été perçus en des points de l'environnement assez proches et représentent donc a peu près la même zone, ce décalage correspond à la rotation qui aligne les deux scans (Figure 10.5 gauche).

Une fois la direction recalée, pour corriger la translation, on projette les points selon la direction principale des tangentes. On construit ensuite l'histogramme du nombre de points projetés sur la perpendiculaire à cette direction et, en cherchant le maximum de corrélation entre les histogrammes correspondant aux deux scans, on corrige le décalage en translation selon cette direction (Figure 10.5 droite). On recommence ensuite cette procédure dans la direction perpendiculaire.

Cette méthode fonctionne bien dès qu'il y a des structures rectilignes dans l'environnement qui conduisent à des histogrammes contenant des pics pour lesquels le recalage par corrélation fonctionne bien. Ils faut toutefois prendre un certain nombre de précautions qui ne sont pas détaillées ici, notamment faire un filtrage intelligent des scan afin de ne garder que les points qui ne correspondent ni à du bruit, ni à des éléments dynamiques. Il faut également veiller à ne réaliser ce recalage que pour des scans qui ont été perçus à des positions effectivement proches, sous peine de recueillir des résultats très fantaisistes. Pour s'assurer de ce point, il est possible de vérifier la qualité de la corrélation des histogrammes afin de vérifier a posteriori que les deux scans représentaient des portions similaires de l'environnement.

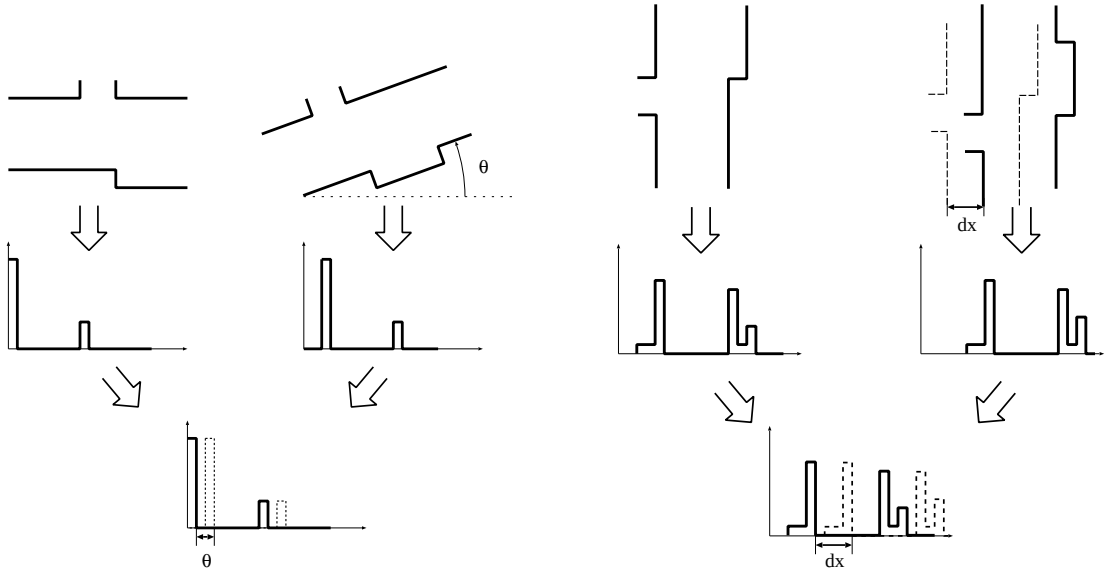


Figure 10.5: Illustration de la méthode de recalage par corrélation d'histogrammes. Recalage en rotation (à gauche) et en translation (à droite).

10.2.4 Limitations de l'estimation de la position par les perceptions

L'hypothèse d'un environnement sans *perceptual aliasing* est relativement forte, car beaucoup de capteurs en robotique sont limités et bruités. De plus, les environnements intérieurs, de type bureaux, peuvent être très réguliers et présenter de nombreuses zones apparemment similaires pour le robot. Toutefois, les environnement courants contiennent souvent suffisamment d'informations accessibles à des capteurs précis et efficaces. Un être humain, par exemple, n'a aucun mal à se repérer dans un immeuble de bureaux, en lisant les numéros écrits sur les portes (trouver son chemin jusqu'à la sortie est un autre problème !). Il est donc théoriquement possible de concevoir des systèmes suffisamment discriminants pour être capables de se repérer grâce aux seules perceptions et une carte précise. Il est de plus possible d'aménager l'environnement afin de simplifier la tâche de perception pour le robot (comme le montre l'exemple des numéros de porte). Cette solution nuit toutefois à l'autonomie du robot puisqu'il est alors limité aux environnements bien définis qui ont été préparés à l'avance. Le rejet de cette dernière solution et la difficulté de réaliser des capteurs suffisamment discriminant conduit donc la plupart des systèmes de navigation robotique à prendre en compte le *perceptual aliasing* et à utiliser les données proprioceptives pour déterminer leur position de manière unique. Nous allons décrire les différentes méthodes qui peuvent être utilisées dans la suite de ce chapitre.

10.3 Suivi d'une hypothèse unique

Lorsque les perceptions ne suffisent pas pour estimer la position de manière unique, une seconde source d'estimation de la position du robot est nécessaire pour lever l'ambiguïté. Cette seconde

estimation provient, d'une part de la position déterminée lors de la précédente phase de localisation et, d'autre part, des données proprioceptives recueillies depuis cet instant. Les méthodes présentées dans cette section utilisent cette seconde estimation pour sélectionner ou calculer, à chaque instant, la position qui est la plus cohérente vis-à-vis de cette estimation. Les positions estimées grâce aux perceptions qui ne sont pas compatibles avec la position précédente sont simplement ignorées.

10.3.1 Cartes topologiques

Dans une carte topologique, sélectionner le nœud correct parmi les nœuds correspondant aux perceptions peut reposer simplement sur l'adjacence avec le nœud précédent. Dans ce cas, le nœud sélectionné est celui qui est connecté au nœud représentant la position précédente. Cette information est toutefois rarement suffisante et les relations métriques mémorisées dans les arêtes entre nœuds sont souvent utilisées en complément. Le nœud sélectionné est donc celui dont la position relative par rapport au nœud précédent correspond le mieux aux données proprioceptives [82, 83, 33, 105]. Lorsqu'une position métrique est associée à chaque nœud, c'est le nœud dont la position est la plus proche de la position estimée par l'odométrie qui est sélectionné [12, 84, 145].

Certains modèles fonctionnent dans le sens opposé. Au lieu d'utiliser les données proprioceptives pour sélectionner un nœud parmi les nœuds possibles, ils utilisent ces données pour restreindre l'ensemble des nœuds possibles et utilisent ensuite les perceptions pour sélectionner le nœud correct parmi ceux-ci. Les perceptions sont, par exemple, utilisées pour choisir un nœud parmi tous les nœuds adjacents au nœud précédent [141], ou parmi les nœuds suffisamment proches de la position estimée par l'odométrie [142].

Enfin, certains modèles intègrent les deux étapes en une seule en calculant la probabilité que chaque nœud représente la position courante. Cette probabilité intègre, d'une part, la similarité du nœud avec les perceptions courantes, et d'autre part sa proximité avec la position estimée par l'odométrie. Le nœud ayant la plus forte probabilité peut alors être reconnu [98], ou la position peut être estimée par codage par population de vecteurs en utilisant les probabilités calculées [6, 137].

10.3.2 Cartes métriques

Dans une carte métrique, l'estimation initiale de la position est utilisée pour restreindre l'espace de recherche de la position correspondant aux perceptions. Dans le cas où la carte contient des objets, une estimation de la position permet de simplifier le problème de l'appariement entre les objets perçus et ceux de la carte. En effet, dans le cas où les senseurs sont soumis à un fort *perceptual aliasing*, de nombreux objets identiques, situés à des positions différentes, sont présents dans la carte. Lorsque le robot perçoit un objet, déterminer quel objet a été perçu exige d'examiner un grand nombre de possibilités. L'estimation de la position du robot permet donc d'estimer la position des objets perçus et donc de déterminer à quels objets de la carte ils correspondent. Ce choix se fait en général en appariant simplement chaque objet perçu à l'objet

mémorisé identique le plus proche [8, 28, 31, 39, 57, 92, 103, 129, 144]. Une fois l'appariement effectué, les objets sont identifiés sans ambiguïté et permettent donc d'estimer la position de manière unique.

Lorsque la position est déterminée par la mise en correspondance d'une carte locale et d'une carte globale, la position estimée est utilisée pour restreindre la recherche de la position donnant la meilleure correspondance entre les deux cartes [119, 122, 131, 147]. La recherche du maximum de correspondance est simplement effectuée sur une zone limitée autour de la position estimée précédemment. La zone étant de faible étendue, le problème de *perceptual aliasing* se pose moins et la recherche conduit en général à une position unique.

Lorsque la position correspondant aux perceptions a été identifiée de manière unique, elle peut être considérée directement comme la nouvelle estimation de la position du robot [147, 57, 144]. Cependant la plupart des modèles considèrent que cette estimation est entachée d'erreur, de la même manière que l'estimation initiale provenant de l'odométrie. La nouvelle position du robot est donc en général une combinaison de ces deux positions. La plupart des modèles [8, 129, 103, 31, 92, 119, 14, 18, 94, 28] utilisent un *filtre de Kalman* [99] pour réaliser cette combinaison. Ce filtre permet de calculer une estimation optimale de la position du robot, connaissant les deux positions et leurs covariances respectives. Il constitue une méthode classique de localisation et est décrit en détails dans la section 10.3.3. D'autres méthodes sont également utilisables pour combiner ces deux informations, par exemple la minimisation d'une fonction de coût reliée à ces deux positions [131], ou l'utilisation de la méthode des *moindres carrés récursifs* [16].

10.3.3 Le filtrage de Kalman pour la localisation

Principe

Le filtre de Kalman [99] permet d'estimer l'état d'un système à partir d'une prédiction bruitée de son évolution et de mesures bruitées de cet état. C'est un filtre récursif optimal, qui suppose que le système considéré est linéaire et les bruits blancs (de moyenne nulle). Pour la localisation en robotique mobile, l'état du système est la position du robot, la prédiction de l'évolution proviendra des données odométriques et les mesures proviendront des perceptions, qui permettent de calculer la position grâce à la carte. Dans la suite, nous présentons succinctement la description mathématique du filtre avant de donner un exemple d'application.

Le filtre donne à chaque instant une estimation $\hat{X}_t$ de la valeur de l'état X_t du système, ainsi qu'une estimation de la précision de cette estimation sous forme de sa matrice de covariance P_t[2].

L'évolution de l'état du système est modélisée par l'équation linéaire suivante :

$$X_t = A.X_{t-1} + B.u_t + \varepsilon_{evo} \tag{10.1}$$

où A et B sont des matrices, u_t est l'odométrie relevée par le robot ou le vecteur des commandes qui lui sont données et ε_{evo} est le bruit sur l'estimation de l'état, supposé d'espérance nulle et de variance $Q = E\{\varepsilon_{evo}\varepsilon_{evo}^T\}$.

[2] Si on note le bruit comme une variable aléatoire N de moyenne nulle, avec $\hat{X}_t = X_t + N$, la matrice de covariance est données par $P_t = E\{N.N^T\} = E\{(X-\hat{X})(X-\hat{X})^T\}$. En pratique, N et X sont inconnus, mais le filtre de Kalman fournit directement une estimation de P_t

Une mesure Y_t effectuée sur l'état du système sera donnée par l'équation linéaire:

$$Y_t = H.X_t + \varepsilon_{obs} \tag{10.2}$$

ou H est la matrice d'observation et ε_{obs} le bruit de mesure, supposé de moyenne nulle et de variance $P_Y = E\{\varepsilon_{obs}\varepsilon_{obs}^T\}$.

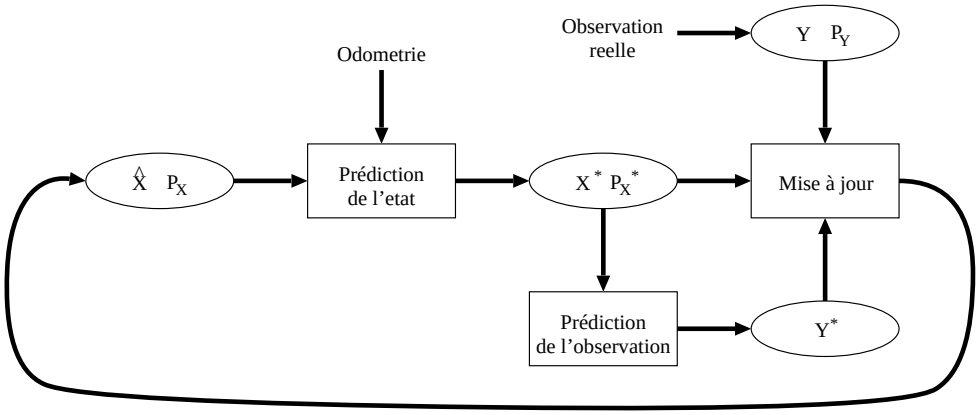


Figure 10.6: Schéma de fonctionnement du filtre de Kalman.

Le fonctionnement du filtre se déroule en quatre étapes (Figure 10.6) :

- **Prédiction de l'état** à l'instant courant X_t^*, ainsi que de sa covariance P_t^* à partir du modèle d'évolution, de l'estimation au pas de temps précédent et de la commande depuis cet instant:

$$X_t^* = A.\hat{X}_{t-1} + B.u_t \tag{10.3}$$

  La covariance est également prédite par la formule:

$$P_t^* = A.\hat{P}_{t-1}.A^T + B.Q.B^T \tag{10.4}$$

- **Prédiction de l'observation** à partir du modèle d'observation et de l'estimation de l'état:

$$Y_t^* = H.X_t^* \tag{10.5}$$

- **Observation** de l'état: on obtient, grâce au système perceptif, une mesure Y_t, dont on estime le bruit P_Y grâce au modèle du processus de perception.

- **Correction de l'état prédit** par mise à jour proportionnellement à l'erreur entre l'observation prédite et l'observation réalisée:

$$\hat{X}_t = X_t^* + K(Y_t - Y_t^*) \tag{10.6}$$

$$\hat{P}_t = P_t^* - KHP_t^* \tag{10.7}$$

ou K est le gain de Kalman, calculé pour minimiser l'erreur d'estimation au sens des moindres carrés et donné par la formule :

$$K = P_t^* H^T . (H . P_t^* . H^T + P_Y)^{-1} \tag{10.8}$$

Ces quatre étapes sont utilisées à chaque nouvelle information de déplacement et de perception, afin de mettre à jour l'estimation de l'état du système.

Application dans le cas mono variable

Pour montrer le fonctionnement intuitif de ce filtre, présentons son application dans un cas trivial : le cas où l'état du système est décrit par une variable scalaire $X = x$, de variance $P_t = {\sigma_x}^2$. Si on suppose de plus que l'observation permet d'obtenir directement la valeur de l'état: $Y_t = y = x$ avec une variance $P_Y = {\sigma_y}^2$, le gain du filtre s'écrit simplement:

$$K = \frac{{\sigma_x^*}^2}{{\sigma_x^*}^2 + {\sigma_y}^2}$$

et l'équation de mise à jour devient:

$$\hat{x} = x^* + \frac{{\sigma_x^*}^2}{{\sigma_x^*}^2 + {\sigma_y}^2}(y - x^*) \tag{10.9}$$

$$= \frac{{\sigma_x^*}^2 y + {\sigma_y}^2 x}{{\sigma_x^*}^2 + {\sigma_y}^2} \tag{10.10}$$

La mise à jour revient donc à faire une moyenne pondérée par la variance de la prédiction et de l'observation. Cette moyenne donne plus d'importance à la valeur ayant la variance la plus faible et donc la plus fiable (Figure 10.7).

Intuitivement, le filtre de Kalman va donc accorder plus d'importance aux valeurs pour lesquelles l'incertitude est la plus faible et les privilégier lors de la mise à jour. En pratique ces variances sont souvent basées sur des estimations empiriques (notamment en robotique pour l'odométrie et les capteurs). Il faut donc faire très attention à ne pas sous-estimer ces incertitudes de mesure, car, dans ce cas, le filtre de Kalman convergerait vers ces mesures, ce qui peu conduire à une divergence du filtre si ces mesures ne sont pas réellement aussi fiable que l'estimation de covariance le laisse penser.

Le filtrage de Kalman étendu

Le filtre présenté dans la section précédente suppose des équations d'évolution et d'observation linéaires, ce qui n'est pas le cas en robotique mobile dès que l'on représente la direction du

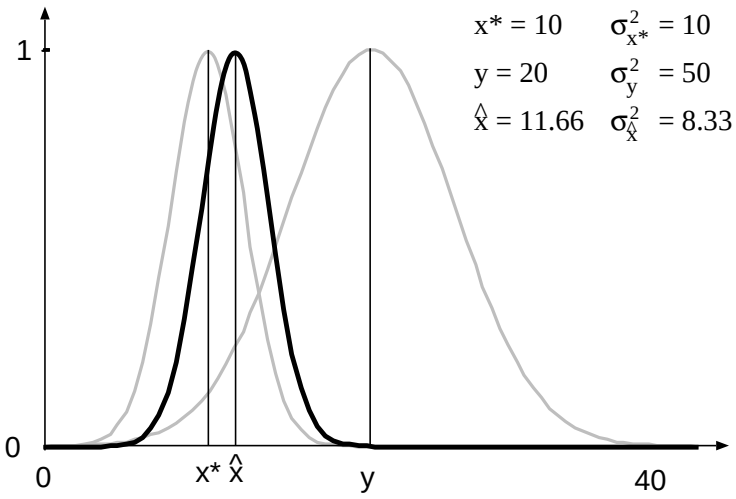


Figure 10.7: Illustration du filtre de Kalman dans le cas mono variable. La valeur estimée est la moyenne des valeurs prédites et observées pondérées par leurs variances. Ici, l'estimation $\hat{x}$ est plus proche de x^* qui une variance plus faible.

robot. Pour étendre le filtrage aux systèmes non-linéaires, le filtre de Kalman étendu linéarise simplement les équations grâce à un développement de Taylor.

Partant des équations non linéaires suivante:

$$X_t = f(X_{t-1}, u_t) + \varepsilon_{evo} \tag{10.11}$$
$$Y_t = h(X_t) + \varepsilon_{obs} \tag{10.12}$$

on utilise les matrices Jacobiennes A, B et H définies par :

$$A_{ij} = \frac{\partial f_i}{\partial x_j} \tag{10.13}$$
$$B_{ij} = \frac{\partial f_i}{\partial u_j} \tag{10.14}$$
$$H_{ij} = \frac{\partial h_i}{\partial x_j} \tag{10.15}$$

Avec ces deux matrices Jacobiennes, le principe du filtre de Kalman reste exactement le même, en remplaçant simplement les équations du filtre de Kalman original par les équations suivantes :

$$X_t^* = f(X_{t-1}, u_t) \tag{10.16}$$
$$P_t^* = A.\hat{P}_{t-1}.A^T + B.Q.B^T \tag{10.17}$$
$$Y_t^* = h(X_t^*) \tag{10.18}$$
$$K = P_t^* H^T.(H.P_t^*.H^T + P_Y)^{-1} \tag{10.19}$$
$$\hat{X}_t = X_t^* + K(Y_t - Y_t^*) \tag{10.20}$$
$$\hat{P}_t = P_t^* - KHP_t^* \tag{10.21}$$

Application à la localisation sans perceptual aliasing

Supposons, à titre d'exemple, un robot dont on peut commander la vitesse de translation v et de rotation ω. L'état que l'on cherche à estimer est simplement sa position dans le plan: $X_t = (x_t, y_t, \theta_t)$. Le vecteur de commande est $u_t = (v_t, \omega_t)$, ce qui conduit à l'équation d'évolution du système:

$$f(X_t, u_t) = \begin{bmatrix} x_t + v_t.dt.cos(\theta_t) \\ y_t + v_t.dt.sin(\theta_t) \\ \theta_t + \omega_t.dt \end{bmatrix}$$

Nous supposons de plus que le bruit entachant cette estimation est indépendant pour chaque variable et proportionnel aux vitesses:

$$Q_t = \begin{bmatrix} \sigma_T.v_t & 0 & 0 \\ 0 & \sigma_T.v_t & 0 \\ 0 & 0 & \sigma_R.\omega_t \end{bmatrix}$$

Supposons enfin que le système de perception permette de mesurer directement la position, par référence à la carte. L'équation d'observation sera simplement:

$$h(X_t) = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix}$$

et nous estimons un bruit constant sur cette mesure:

$$P_y = \begin{bmatrix} \sigma_O & 0 & 0 \\ 0 & \sigma_O & 0 \\ 0 & 0 & \sigma_{O_\theta} \end{bmatrix}$$

Les matrices Jacobiennes correspondant à ces équations, obtenues en dérivant f et h sont donc:

$$A = \begin{bmatrix} 1 & 0 & -v_t.dt.sin(\theta) \\ 0 & 1 & v_t.dt.cos(\theta) \\ 0 & 0 & 1 \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Compte tenu du fait qu'ici $H = I$, l'algorithme du filtre de Kalman étendu se simplifie :

$$\begin{aligned} X_t^* &= f(X_{t-1}, u_t) \\ P_t^* &= A.\hat{P}_{t-1}.A^T + B.Q.B^T \\ Y_t^* &= X_t^* \\ K &= P_t^*(P_t^* + P_Y)^{-1} \\ \hat{X}_t &= X_t^* + K(Y - X_t^*) \\ \hat{P}_t &= P_t^* - KP_t^* \end{aligned}$$

Pour l'initialisation de l'algorithme, nous supposons connaître une estimation de la position du robot:

$$\hat{X}_0 = \begin{bmatrix} x_0 \\ y_0 \\ \theta_0 \end{bmatrix}$$

$$\hat{P}_0 = \begin{bmatrix} \sigma_{x_0} & 0 & 0 \\ 0 & \sigma_{y_0} & 0 \\ 0 & 0 & \sigma_{\theta_0} \end{bmatrix}$$

et nous appliquons les équations de mise à jour pour chaque nouveau déplacement ou chaque nouvelle perception.

Application à la localisation avec perceptual aliasing

En cas de perceptual aliasing, plusieurs valeurs de la mesure proviennent de la phase d'observation. Il faut donc choisir parmi ces valeurs la valeur correspondant à la position réelle du robot qui sera utilisée pour la mise à jour.

Lorsque l'observation donne une mesure de la position, il est possible de sélectionner simplement la mesure la plus proche de la position prédite pour le robot. Dans le cas général, il est préférable d'utiliser la *distance de Mahalanobis* , qui est une mesure de distance normalisée par la covariance. Cette mesure permet par exemple de privilégier une mesure plus lointaine mais moins précise qui aura en fait une probabilité plus grande de correspondre à la mesure prédite (Figure 10.8).

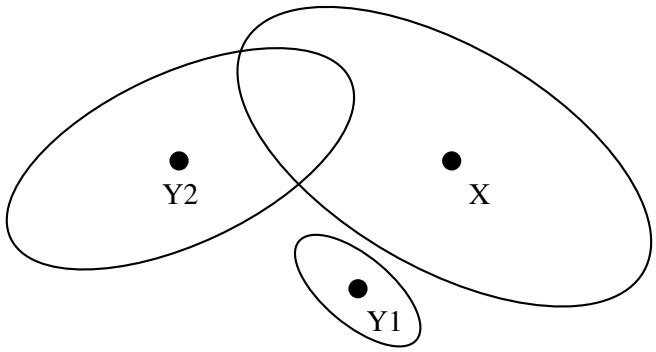


Figure 10.8: Illustration de l'intérêt de la distance de Mahalanobis. La mesure Y1 est plus proche de X en distance euclidienne mais les incertitudes font que ces mesures sont incompatibles. La distance de Mahalanobis sera plus faible pour Y2 et X du fait que les incertitudes montrent que ces mesures peuvent provenir de la même variable.

Pour deux valeurs X et Y de covariances P_X et P_Y, cette distance vaut:

$$d^2 = \frac{1}{2}(X-Y)^T(P_X+P_Y)^{-1}(X-Y) \tag{10.22}$$

ce qui se traduit dans le cas scalaire par une simple pondération par les variances:

$$d^2 = \frac{(x-y)^2}{2(\sigma_x^2+\sigma_y^2)}$$

Dans le cas du filtrage de Kalman, cette distance est utilisée entre l'observation prédite et les différentes observations faites sur le système:

$$\begin{aligned} d^2 &= \frac{1}{2}(Y^*-Y)^T(P_{Y^*}+P_Y)^{-1}(Y^*-Y) \\ &= \frac{1}{2}(H.X^*-Y)^T(H.P_t^*.H^T+P_Y)^{-1}(H.X^*-Y) \end{aligned}$$

à partir des distances de Mahalanobis des différentes observations à l'observation prédite, il est possible de sélectionner l'observation la plus proche ou de choisir un seuil qui permettra de déterminer si une des observations correspond bien à l'état courant. Si une des observation est en dessous de ce seuil, elle est utilisée pour la mise à jour du filtre, sinon, on considère que l'état n'a pas pu être mesuré et on ne fait pas de correction de la prédiction.

L'une des principales faiblesses du filtre de Kalman pour la localisation provient précisement de cette phase d'identification (d'appariement) des éléments perçus. En effet, en cas de mauvais choix dû à une mauvaise prédiction de la position du robot ou a une erreur de perception, l'erreur d'estimation de la position sera confirmée, voir augmentée. Un tel processus conduit rapidement à une divergence du filtre et à une perte de la position du robot.

Le filtre de Kalman sans parfum (unscented)

La linéarisation utilisée par le filtre de Kalman étendu peut poser problème lorsque le modèle est fortement non-linéaire ou lorsque les déplacements ou les erreurs de perception sont grandes. Dans certain cas, l'estimation de la covariance après transformation peut être très mauvaise (Figure 10.9). Pour ces cas là, il existe une autre manière de linéariser les équations qui conduit au filtre de Kalman "sans parfum" (Unscented) [74].

Au lieu de linéariser l'équation non-linéaire autour de la moyenne de la variable, ce filtre utilise plusieurs point d'échantillonnage à partir de la gaussienne de départ, calcule leurs correspondants via l'équation non-linéaire et estime la variance de la gaussienne à partir de ces correspondants (Figure 10.9). Cette stratégie permet une bien meilleure prédiction de l'état, que ce soit pour l'évolution ou l'observation et permet donc d'étendre le domaine de fonctionnement du filtre de Kalman pour traiter des problèmes avec des mises à jour de plus grande ampleur. Elle est cependant plus couteuse en calcul que le simple EKF.

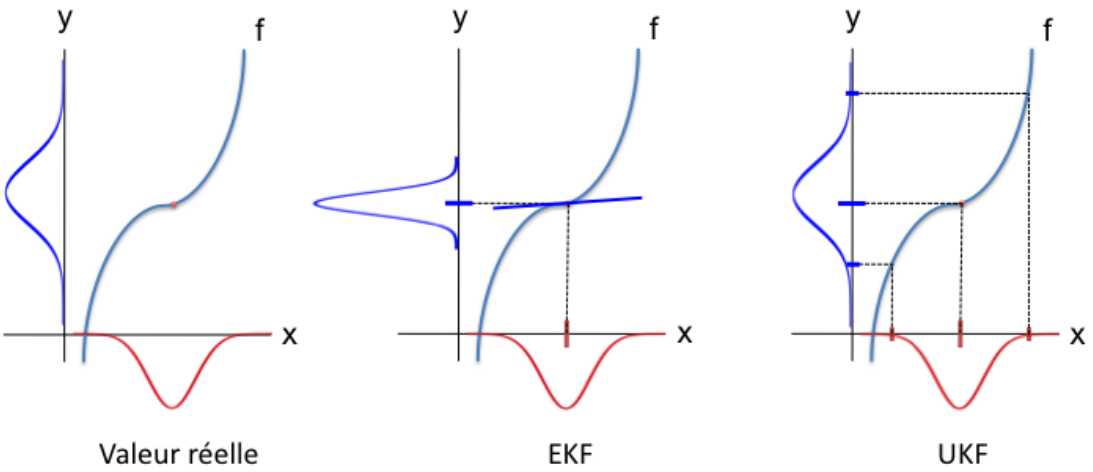


Figure 10.9: Principe de fonctionnement du filtre de Kalman sans parfum, illustré sur un exemple mono dimensionnel. Le filtre étendu (EKF) peut conduire à de fortes erreurs pour des fonctions non-linéaires car il n'utilise que la moyenne et la dérivée en ce point. Le filtre sans parfum (UKF) utilise plusieurs points pour estimer les paramètres de la gaussienne transformée.

10.3.4 Limitations du suivi de position

La limitation principale de toutes ces méthodes de suivi de position est qu'elles ne garantissent une bonne estimation de la position que localement, autour de l'estimation initiale de la position. En pratique, si cette estimation initiale est trop éloignée de la position réelle, ces modèles ne pourront pas estimer correctement la position du robot (cf. figure 10.10). Ces modèles ne garantissent donc pas que la position calculée soit la position de la carte qui corresponde globalement le mieux aux données recueillies par le robot et donc la position réelle la plus probable.

Ce problème prend toute son importance lorsque l'estimation de la position est perturbée à la suite d'informations proprioceptives ou de perceptions erronées. En effet, de telles informations erronées peuvent faire diverger l'estimation de la position de telle manière que le système soit par la suite incapable de retrouver une estimation correcte de la position.

La position correcte du robot pourra cependant être retrouvée par l'une des méthodes de localisation globale décrites dans la section précédente. Cette position pourra ensuite être utilisée comme nouvelle position initiale dans le processus de suivi de position. Toutefois, il est également possible d'utiliser une des méthodes de localisation globale décrites dans la fin de ce chapitre qui permettent de ne plus dépendre d'une estimation initiale correcte de la position. Cette seconde solution, qui ne requiert pas l'utilisation séparée de deux méthodes de localisation sera en général plus robuste, mais parfois moins précise.

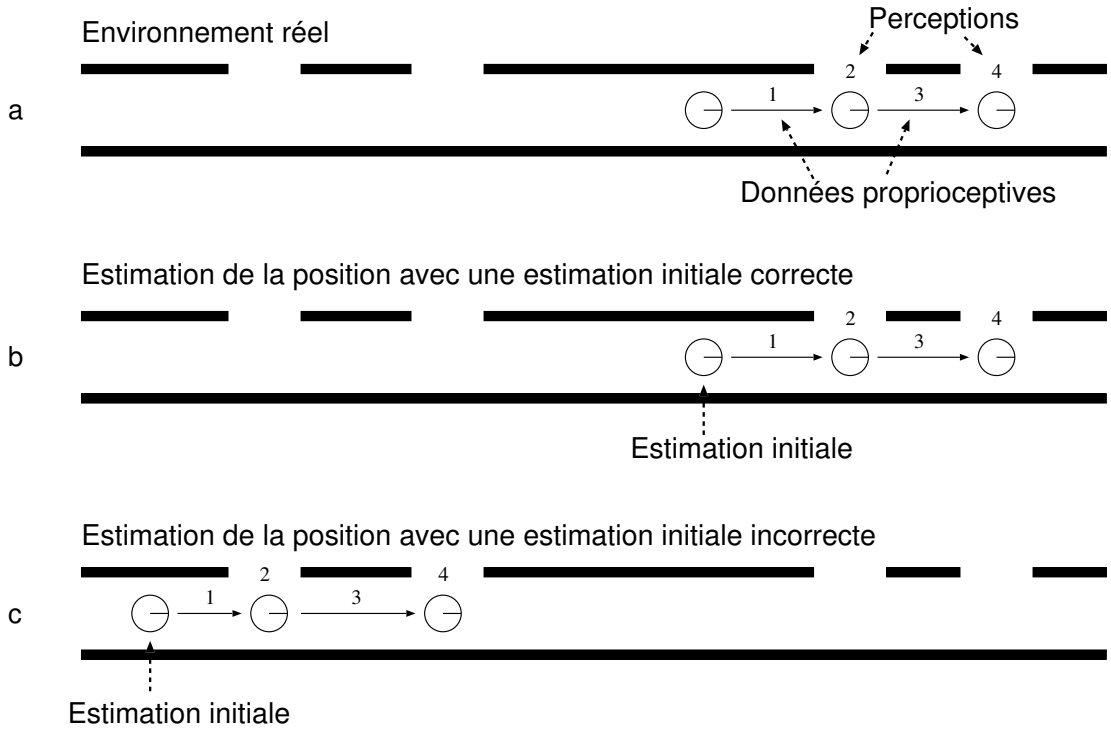


Figure 10.10: La position estimée par le suivi de position dépend fortement de l'estimation initiale de la position. Dans cet exemple, le robot mesure son déplacement dans un couloir (1 et 3) dans lequel il détecte des portes (2 et 4) sans être capable de les reconnaître individuellement (Partie a). Le système de localisation va estimer la position du robot en intégrant ces données. Si l'estimation initiale de la position est proche de la position correcte, le système de localisation sera capable d'estimer précisément la position réelle du robot (Partie b). Toutefois, si l'estimation initiale de la position est trop éloignée de la position réelle, le système fournira une estimation de la position qui n'est que localement optimale et ne correspondra pas à la position réelle (Partie c).

10.4 Suivi de plusieurs hypothèses

La localisation globale, lorsque le robot est soumis au *perceptual aliasing*, ne peut se faire qu'en utilisant de manière optimale les informations proprioceptives et les perceptions. Contrairement au suivi de position qui utilise l'estimation précédente de la position pour sélectionner l'une des positions caractérisées par les perceptions et ignorer les autres, il faut tenir compte à chaque étape de toutes ces positions possibles. Ces positions conduisent à des hypothèses qui peuvent être mises à jour en fonction des données proprioceptives et qu'il faut comparer, afin de choisir la plus pertinente à chaque étape (cf. figure 10.11).

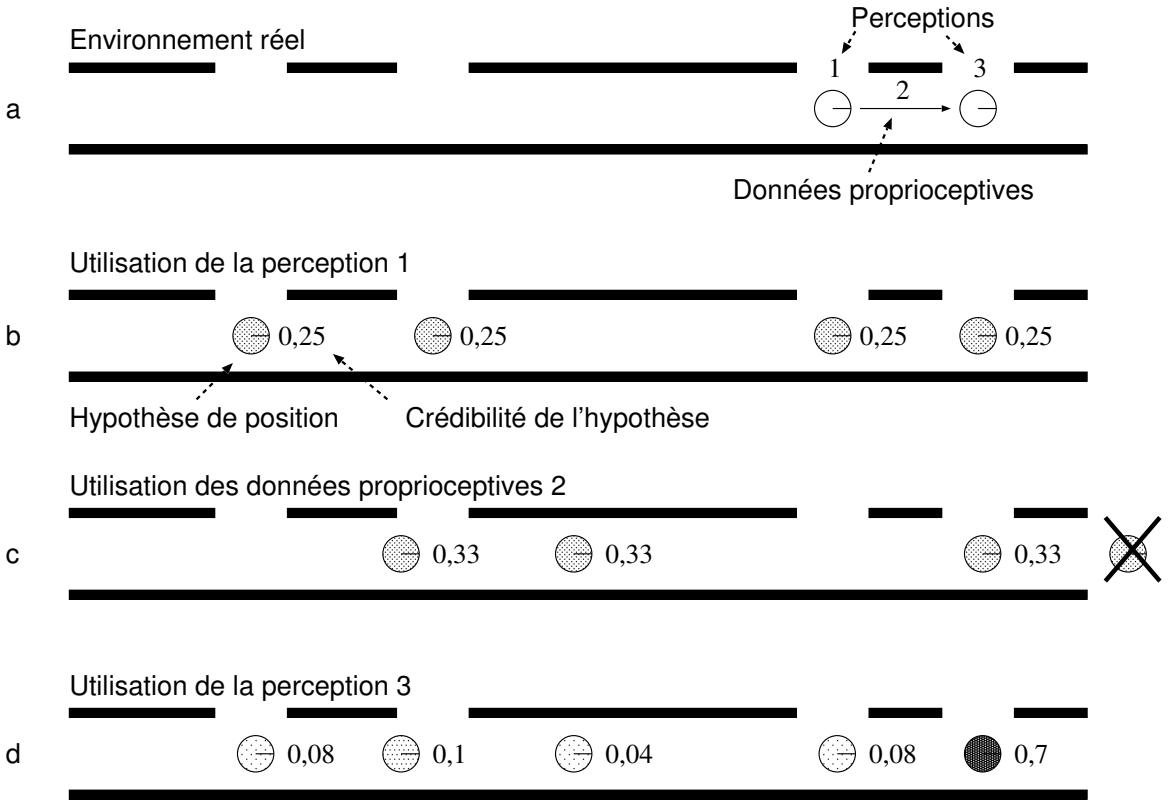


Figure 10.11: Le suivi de plusieurs hypothèses permet de déterminer la position au sein de la carte qui globalement correspond le mieux aux données recueillies par le robot. Dans cet exemple, le robot est capable d'estimer son déplacement dans un couloir (2) dans lequel il détecte des portes (1 et 3) sans être capable de les reconnaître individuellement (Partie a). La perception d'une porte sans aucune estimation préalable de la position permet simplement de créer plusieurs hypothèses de position pouvant correspondre à cette perception. Il est, à ce stade impossible de décider quelle hypothèse est correcte (Partie b). L'intégration des données proprioceptives permet de mettre à jour la position de chacune des hypothèses, mais ne permet pas de les discriminer (Partie c). Des nouvelles perceptions permettent d'estimer la crédibilité relative de chacune des hypothèses en rendant plus crédibles les hypothèses dont la position correspond aux perceptions courantes et moins crédible les autres. (Partie d). L'hypothèse ayant alors la plus forte crédibilité correspond à la position qui rend le mieux compte des données recueillies par le robot.

10.4.1 Suivi explicite de plusieurs hypothèses

Ce suivi de plusieurs hypothèses peut être réalisé de manière explicite, en gérant une liste des hypothèses en question. Lorsque des données proprioceptives sont disponibles, chaque hypothèse est simplement mise à jour de manière à refléter le déplacement du robot (cf. figure 10.11c). Lorsque de nouvelles perceptions sont disponibles, l'ensemble des positions de la carte susceptibles de correspondre à ces perceptions est déterminé. Cet ensemble est ensuite comparé à l'ensemble des hypothèses. Si une hypothèse correspond à une position perçue, cette hypothèse est alors mise à jour en utilisant les perceptions par une méthode similaire à celle permettant le suivi de position, par exemple un filtre de Kalman. Les positions perçues qui ne correspondent à aucune hypothèse sont utilisées pour créer de nouvelles hypothèses associées à la position correspondante. La crédibilité de chacune des hypothèses est ensuite évaluée, généralement en fonction de la proximité de l'hypothèse avec une position correspondant aux perceptions du robot (cf. figure 10.11d). Ainsi, une hypothèse verra sa crédibilité augmenter si elle est proche d'une des positions correspondant aux perceptions du robot, elle la verra diminuer dans le cas contraire. De nouvelles hypothèses peuvent également être ajoutées pour les positions correspondant aux perceptions qui ne correspondent à aucune hypothèse existante.

De tels modèles ont été implémentés en utilisant des cartes topologiques [37] où les différentes hypothèses correspondent à différents noeuds de la carte. Il existe également des modèles de ce type utilisant des cartes métriques [110, 73]. Ces derniers modèles utilisent en général plusieurs filtres de Kalman en parallèle et permettent de résoudre en grande partie les problèmes de divergence du filtre de Kalman employé seul. Cette technique est connue sous le nom de *multi hypothesis tracking (MHT)* [73].

10.4.2 Le filtrage Bayésien

Les méthodes de localisation multi-hypothèses peuvent être vues dans un cadre plus large, celui du *filtrage Bayésien*. Cette méthode de filtrage permet d'intégrer de manière similaire les deux types d'informations (odométrie et perceptions), mais ne gère en général pas explicitement les hypothèses de position. Les différentes hypothèses sont ici remplacées par une distribution de probabilité de présence du robot sur l'ensemble des positions possibles de la carte. Cette représentation permet donc de considérer chacune des positions au sein de la carte comme une position possible du robot dont il faut évaluer la probabilité. Nous présentons d'abord le cadre général qui permet la gestion de cette distribution de probabilité, avant de voir comment il peut être utilisé en pratique dans des cas discrets ou continus.

Le filtrage Bayésien regroupe un ensemble de méthodes d'estimation d'état utilisant les probabilités et plus particulièrement la loi de Bayes:

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \tag{10.23}$$

Dans le cadre de la localisation en robotique mobile, X est en général la position et Y une perception de l'environnement. Cette loi nous permet donc d'estimer la probabilité $P(X|Y)$ des

positions, connaissant une perception, c'est à dire précisément ce que nous cherchons à calculer pour localiser un robot. Pour ce calcul, nous aurons besoin de $P(Y|X)$, la probabilité d'une perception connaissant la position, qui peut être calculée grâce à la carte de l'environnement et au modèle du capteur utilisé. Nous aurons également besoin d'une estimation des probabilités des positions $P(X)$ avant cette perception, ainsi que de la probabilité globale $P(Y)$ de cette perception. Dans un filtre Bayésien, cette formule est utilisée de manière récursive et $P(X)$ est donc simplement l'estimation précédente de la probabilité des positions. Quant à $P(Y)$, il peut être remplacé par un artifice de calcul lors de l'utilisation de la formule. En effet, par la loi des probabilités marginales :

$$P(Y) = \sum_X P(Y|X)P(X)$$

ce qui permet d'utiliser la loi de Bayes pour calculer les probabilités $P(X|Y)$ de la manière suivante:

$$\begin{aligned} \forall X, temp_{X|Y} &= P(Y|X)P(X) \\ P(Y) &= \sum_X temp_{X|Y} \\ \forall X, P(X|Y) &= \frac{temp_{X|Y}}{P(Y)} \end{aligned}$$

Dans cette équation, le terme $P(X)$ est la probabilité a priori (*prior* en anglais), $P(X|Y)$ est la probabilité a posteriori (*posterior* en anglais). La puissance de cette équation réside dans le fait qu'elle permet de transposer une quantité simple à évaluer, $P(Y|X)$, en une quantité plus difficile à estimer et qui nous intéresse, $P(X|Y)$. La *vraisemblance* $P(Y|X)$ est simple à évaluer car elle est le produit d'un *raisonnement causal*: connaissant une carte, un modèle de capteur et une position, on peut facilement prévoir les mesures que devraient renvoyer ce capteur. $P(X|Y)$, pour sa part, est le fruit d'un *raisonnement de diagnostic* et il est difficile à évaluer car une perception Y ne permet pas de définir simplement une position, mais peut correspondre à plusieurs, notamment dans le cas du perceptual aliasing[3].

Nous venons de voir comment la loi de Bayes permet de mettre à jour une probabilité de position en fonction d'une perception. Pour la localisation d'un robot mobile, il faut également pouvoir intégrer l'effet d'un déplacement sur une distribution de probabilité. Cela se fait aussi très simplement grâce à l'équation suivante (loi des probabilités marginales):

$$P(X|U) = \sum_{X'} P(X|U,X')P(X') \tag{10.24}$$

Dans cette équation, $P(X|U,X')$ est un modèle du déplacement du robot, qui donne la probabilité d'une position X si le robot exécute l'action U dans la position X'. Ce modèle ne dépend que

[3]Ceci est très général et tout a fait intuitif. Dans le domaine médical, par exemple, il est simple de prévoir de la fièvre (l'observation) si l'on sait que l'on a la grippe (l'état). Le raisonnement inverse est plus difficile car la fièvre peut correspondre à plusieurs maladies.

du robot et correspond souvent au modèle d'odométrie que nous avons vu au début du cours. Comme précédemment, la probabilité a priori, $P(X')$, est le fruit d'une estimation à l'étape précédente.

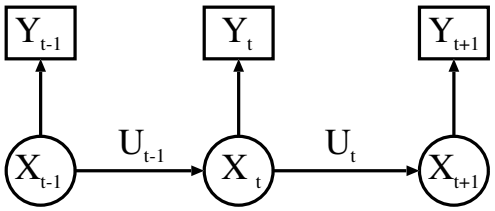


Figure 10.12: Illustration des dépendances considérées pour la localisation d'un robot à partir d'une suite de perceptions et de déplacements. Une flèche indique que la valeur à la pointe dépend de la valeur à l'origine.

Pour localiser un robot, nous cherchons évidement à estimer la position à partir de nombreux déplacements et de nombreuses observations: $P(x_t|u_1, y_2, ..., u_{t-1}, x_{t-1})$. Pour pouvoir réaliser les calculs de manière récursive, l'hypothèse de Markov est utilisée: on suppose que les perceptions ne dépendent que de l'état courant et que la position après un déplacement ne dépend que de la position précédente. Ceci est illustré par le réseau bayésien de la figure 10.12 qui montre les dépendances entre variables et correspond aux simplifications suivantes:

$$\begin{aligned} P(y_t|x_t, u_1, y_2, ..., u_{t-1}) &= P(y_t|x_t) \\ P(x_t|u_1, y_2, ..., u_{t-1}, x_{t-1}) &= P(x_t|u_{t-1}, x_{t-1}) \end{aligned}$$

Partant de ces différents éléments, le filtrage Bayésien permet donc d'estimer de manière récursive l'état d'un système à partir d'une estimation de son évolution et de mesures sur cet état. Pour pouvoir appliquer ce filtrage, nous avons besoin des éléments suivants, qui sont tous connus ou qui peuvent être définis grâce aux modèles du robot, des capteurs et de l'environnement :

- Un modèle d'observation (de capteur) $P(y|x)$ qui donne, pour une position x donnée, la probabilité de la mesure y.
- Un modèle d'évolution (d'action) $P(x|u, x')$ qui donne la probabilité que le robot arrive en x si il exécute l'action u en x'.
- Une suite d'actions et de perceptions $u_1, y_2, ..., u_{t-1}, y_t$.
- Une estimation initiale de la position $P_0(x)$, qui peut, par exemple, être uniforme dans le cas de la localisation globale ou qui peut être une répartition gaussienne si nous connaissons la position initiale du robot.

Le filtre permet d'estimer la position en fonction des données mesurées: $P(x_t|u_1, y_2, ..., u_{t-1}, y_t)$, ce que nous noterons par la suite (en version continue) $Bel(x_t)$ (de l'anglais *Belief State*). L'équation de mise à jour récursive permet alors d'estimer $Bel(x_t)$ en fonction de $Bel(x_{t-1})$. Cette équation se dérive de la manière suivante, en utilisant les loies présentées ci-dessus:

$$\begin{aligned}
Bel(x_t) &= P(x_t|u_1, y_2, ..., u_{t-1}, y_t) \\
(Bayes) &= \eta P(y_t|x_t, u_1, y_2, ..., u_{t-1}) P(x_t|u_1, y_2, ..., u_{t-1}) \\
(Markov) &= \eta P(y_t|x_t) P(x_t|u_1, y_2, ..., u_{t-1}) \\
(probtotales) &= \eta P(y_t|x_t) \int P(x_t|u_1, y_2, ..., u_{t-1}, x_{t-1}) P(x_{t-1}|u_1, y_2, ..., u_{t-1}) dx_{t-1} \\
(Markov) &= \eta P(y_t|x_t) \int P(x_t|u_{t-1}, x_{t-1}) P(x_{t-1}|u_1, y_2, ..., u_{t-1}) dx_{t-1} \\
&= \eta P(y_t|x_t) \int P(x_t|u_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1}
\end{aligned}$$

Pour résumer, l'estimation de l'état par un filtre Bayésien correspond à l'utilisation de l'équation de mise à jour suivante, pour une perception y_t et un déplacement u_{t-1}:

$$Bel(x_t) = \eta P(y_t|x_t) \int P(x_t|u_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1} \tag{10.25}$$

Le filtrage Bayésien regroupe en fait un grand nombre d'approches connues sous des noms différents qui se différentient par la manière dont la distribution de probabilité $Bel(x_t)$ est représentée (figure 10.13). Le filtre de Kalman est par exemple une implantation de ce filtre avec des distributions de probabilités gaussiennes et des modèles linéaires. Le filtre de kalman multi-hypothèses correspond à une représentation sous forme de somme de gaussiennes. Nous allons voir dans la fin de ce chapitre deux autres méthodes qui utilisent soit une représentation discrète soit une représentation sous forme d'un ensemble d'échantillon. Dans ce dernier cas, le filtre s'appelle le *filtre particulaire*.

Représenter la position du robot par une telle distribution de probabilités permet d'intégrer la totalité des informations recueillies au cours du temps. Elle est mise à jour, d'une part à chaque déplacement du robot, et donc à chaque nouvelle donnée proprioceptive, et d'autre part à chaque nouvelle perception. De manière imagée, les données proprioceptives permettent de déplacer les probabilités d'une position à une autre pour refléter le déplacement du robot. Les perceptions permettent de moduler les probabilités de chaque position. Ainsi, les positions pour lesquelles les perceptions prévues à l'aide de la carte sont similaires aux données perçues voient leurs probabilités augmenter, tandis que les autres voient leurs probabilités diminuer.

Lorsqu'on utilise une telle distribution de probabilité, la position du robot calculée est en général donnée par l'hypothèse ayant la plus forte probabilité [66, 69, 125, 127, 24, 49]. Cependant d'autres estimations telles que la moyenne des positions pondérées par leurs probabilités peuvent être utilisées [27], ou des versions intermédiaires telle que la moyenne des x% des meilleurs positions.

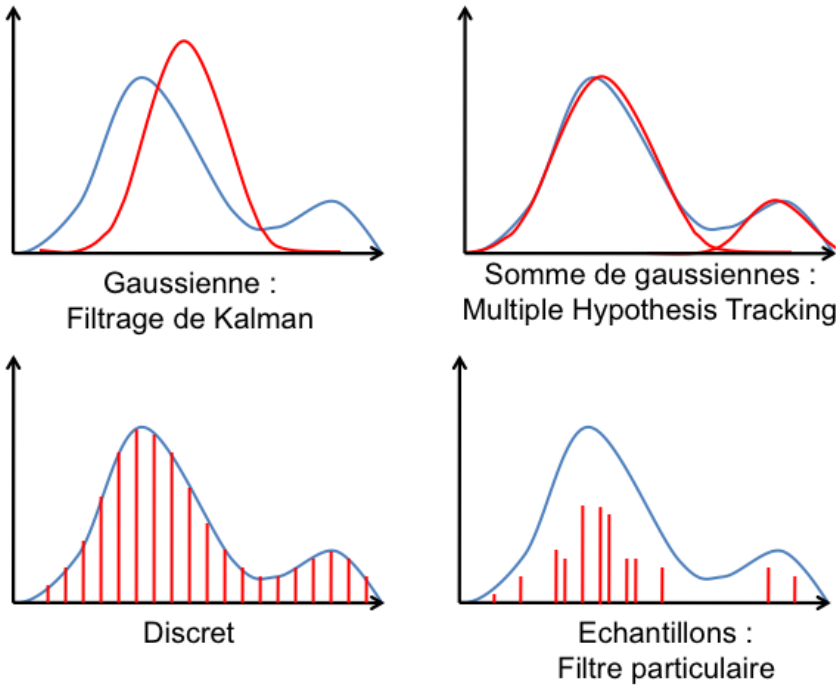


Figure 10.13: Le filtrage Bayésien correspond à différents algorithmes en fonction de la manière dont sont représentées les probabilités.

La distribution de probabilités obtenue dépend faiblement des conditions initiales et peut donc être initialisée à une distribution uniforme lorsqu'aucune information n'est disponible sur la position du robot. La position sera alors retrouvée, même si le robot est soumis à un très fort *perceptual aliasing*, assurant ainsi la localisation globale du robot dans tous les environnements. Ces méthodes sont extrêmement robustes en pratique et mettent en œuvre un système de localisation complètement autonome, ne dépendant d'aucune intervention extérieure [133].

Ces qualités reposent toutefois de manière importante sur le fait que la carte de l'environnement est complète. En effet, les systèmes de suivi de plusieurs hypothèses nécessitent une estimation correcte des probabilités des différentes positions possibles. Or une carte partielle de l'environnement rend une telle estimation à partir des perceptions difficile. Pour cette raison, ces systèmes sont en général utilisés pour la localisation sur des cartes construites dans une phase préalable.

10.4.3 Filtrage Bayésien dans le cas discret

Pour appliquer le filtrage Bayésien que nous avons présenté dans le cas continu, il faut choisir une manière de représenter les distributions de probabilités. Une première approche consiste à discrétiser l'environnement et à donner à chaque position discrète une probabilité approximant la valeur de la distribution continue. Cette approche, également nommée *Histogram Filter* (je ne connais pas la bonne traduction française...), a été utilisée à la fois avec des cartes topologiques

où les nœuds sont utilisés comme positions possibles, [27, 66, 69, 79, 106, 108, 125, 127, 130] et avec des cartes métriques, pour lesquelles il est possible de discrétiser l'ensemble des positions, à la manière des grilles d'occupation [24, 49, 133].

Algorithm 2 Procédure UpdatePerception($Bel(x_i)$,y);

1: $\eta = 0$
2: **for all** x_i **do**
3: $\quad Bel'(x_i) = P(y|x_i)Bel(x_i)$
4: $\quad \eta = \eta + Bel'(x_i)$
5: **end for**
6: **for all** x_i **do**
7: $\quad Bel'(x_i) = Bel'(x_i)/\eta$
8: **end for**
9: Return $Bel'(x_i)$

Algorithm 3 Procédure UpdateMouvement($Bel(x_i)$,u);

1: **for all** x_i **do**
2: $\quad Bel'(x_i) = \sum_{x_k} P(x_i|u,x_k)Bel'(x_k)$
3: **end for**
4: Return $Bel'(x_i)$

Quelle que soit la discrétisation choisie, l'algorithme reste le même. Il s'agit d'évaluer la probabilité $Bel(x_i)$ que le robot soit situé sur l'état x_i de la carte. Ceci est fait par deux procédures différentes selon que l'on cherche à intégrer une perception ou des données proprioceptives. Pour une perception, on utilisera la procédure de l'algorithme 2. Les probabilités $P(y|x_i)$ proviennent soit d'un modèle de capteur métrique pour les grilles d'occupation, soit d'un modèle comparant une perception avec les données mémorisées dans les nœuds pour la carte topologique. Pour un déplacement on utilisera l'algorithme 3. La figure 10.14, tirée de [48] montre un exemple d'évolution de la probabilité de position d'un robot.

L'implantation naïve de cet algorithme conduit à une mise à jour quadratique en fonction du nombre d'états (O(N^2)), ce qui peut être rapidement lourd à calculer. Cependant, le modèle probabiliste de déplacement $P(x|u,x')$ est en général nul dès que l'on s'éloigne de la position spécifiée par la commande u depuis l'état x'. Si l'on note p le nombre d'états pour lequel le modèle est non nul, on peut facilement écrire un algorithme en O(np), qui est donc linéaire en le nombre d'états de la carte.

10.4.4 Filtrage particulaire

Il est également possible d'utiliser une autre méthode pour représenter une distribution de probabilité continue sur l'espace de la carte [46] sans discrétisation de la carte. Cette méthode, le *filtrage particulaire* est l'une des plus efficaces pour la localisation.

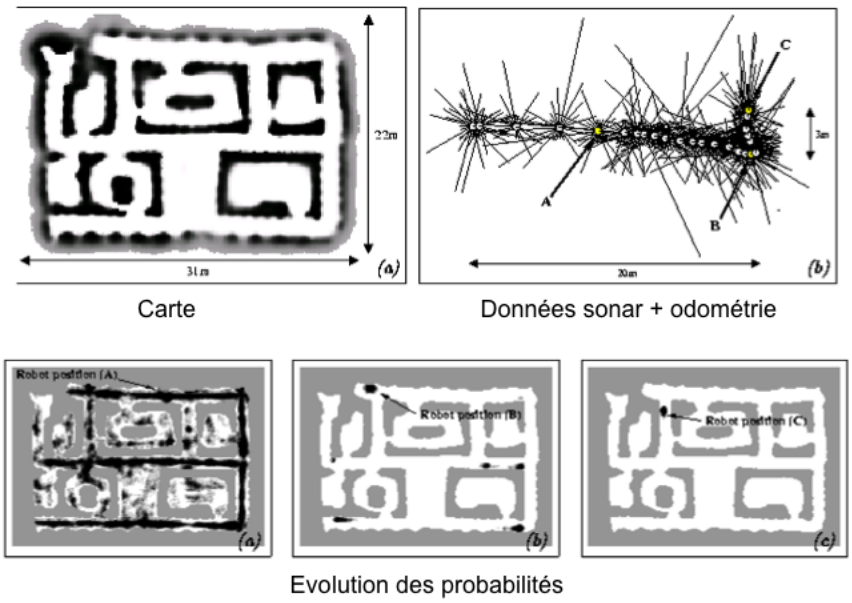


Figure 10.14: Exemple d'évolution de la probabilité de position au cours du temps. Figure tirée de [48].

Pour représenter la distribution de probabilité $Bel(x)$, cette méthode utilise un échantillonage en un ensemble de particules qui permet d'approcher la distribution. En effet, considérons que la position du robot est la position moyenne définie par la distribution de probabilité $Bel(x)$:

$$Position = \int_x x.Bel(x)dx$$

Supposons par ailleurs que l'on soit capable de générer un ensemble de N échantillons de position θ_i, que nous appellerons particules, selon la distribution de probabilité $Bel(x)$ (Figure 10.15 gauche). On peut alors approcher la position par:

$$Position \approx \frac{1}{N}\sum_i \theta_i$$

C'est cet ensemble de particules qui va permettre de représenter la distribution de probabilités de positions du robot. Tout le problème revient alors à être capable de générer récursivement un ensemble de particules réparties selon la loi $Bel(x)$.

Dans notre cas, il est évidement impossible a priori de générer de tels échantillons car $Bel(x)$ est une distribution inconnue que nous cherchons à évaluer. Pour pouvoir néanmoins approcher cette fonction, nous allons introduire une fonction auxiliaire $\pi(x)$, appelée *fonction d'importance* selon laquelle nous allons tirer nos particules. En effet, avec de telles particules, réparties selon une loi $\pi(x)$, il est également possible d'approcher $Bel(x)$. Pour cela, nous commençons par écrire:

$$Bel(x) = \frac{Bel(x)}{\pi(x)}\pi(x) = w(x)\pi(x)$$

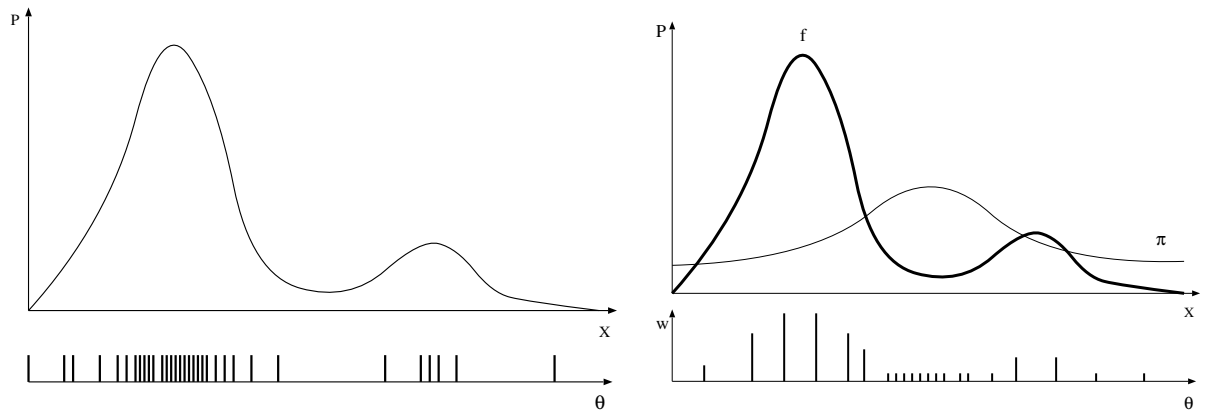


Figure 10.15: Gauche : Illustration de la méthode de représentation d'une distribution de probabilité par des particules. Les particules sont tirées aléatoirement selon la loi à représenter. Droite : Illustration de la méthode de représentation d'une distribution de probabilité par des particules tirées selon une loi d'importance. Les particules sont tirées aléatoirement selon la loi π et ont chacune un poids associé $w = f/\pi$.

avec

$$w(x) = \frac{Bel(x)}{\pi(x)}$$

et donc, si nous générons les échantillons aléatoires θ_i selon la *fonction d'importance* π (connue car nous l'avons choisie), nous pouvons estimer la position par:

$$Position = \int_x x.Bel(x)dx = \int_x x.w(x)\pi(x) \approx \frac{1}{N}\sum_i w(\theta_i)\theta_i$$

Ainsi la distribution $Bel(x)$ peut s'approcher sous la forme d'un ensemble de particules tirées selon une loi $\pi(x)$, chaque particule ayant un poids associé $w(x)$ (Figure 10.15 droite). La difficulté d'échantillonner selon $Bel(x)$ reste cachée dans la difficulté de calculer le poids de chaque particule $w(x)$, mais grâce à un choix judicieux de $\pi(x)$ nous pouvons arriver à rendre ce calcul possible en utilisant l'équation de mise à jour du filtrage Bayésien.

En effet, dans le cas qui nous intéresse:

$$Bel(x_t) = \eta P(y_t|x_t) \int P(x_t|u_{t-1},x_{t-1})Bel(x_{t-1})dx_{t-1} \tag{10.26}$$

en choisissant (intelligemment) $\pi(x) = \int P(x|u_{t-1},x_{t-1})Bel(x_{t-1})dx_{t-1}$, nous obtenons:

$$\begin{aligned} w(\theta) &= \frac{Bel_t(\theta)}{\pi(\theta)} \\ &= \frac{\eta P(y_t|\theta)\int P(\theta|u_{t-1},x_)Bel_{t-1}(x)dx}{\int P(\theta|u_{t-1},x_{t-1})Bel(x_{t-1})dx_{t-1}} \\ &= \eta P(y_t|\theta) \end{aligned}$$

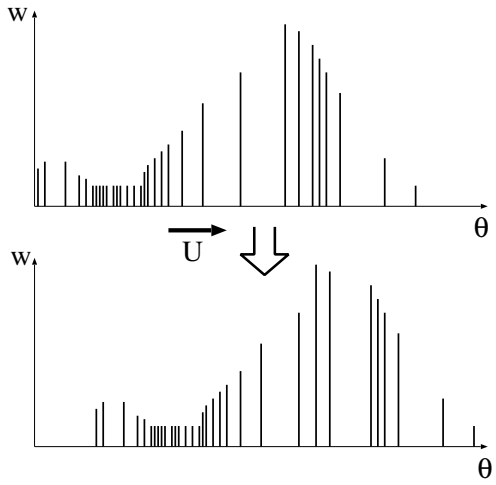


Figure 10.16: Illustration de la mise à jour de l'ensemble des particules en utilisant les déplacements.

Cette valeur est simple à calculer car elle ne dépend que de la carte et du modèle de capteurs pour la position d'un échantillon θ.

Le problème du calcul des poids est donc réglé avec ce choix de $\pi(x) = \int P(x|u_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$. Reste a voir comment il est possible d'échantillonner des particules selon cette loi. Comme nous utilisons une méthode récursive, $Bel(x_{t-1})$ est approchée sous la forme d'un ensemble de particules θ_{t-1}, chacune associée à un poids w_{t-1}. A partir de ces particules, l'échantillonnage selon $\pi(x)$, se déroule en deux étapes: une mise à jour selon le modèle d'évolution $P(x|u_{t-1}, x_{t-1})$, puis un ré-échantillonnage des particules pour correspondre à la loi $\pi(x)$.

Pour la première étape, les particules sont mises à jour à partir du modèle de déplacement en tirant de manière aléatoire, pour chaque particule θ_i, une particule δ_i selon la loi $P(x|u_{t-1}, \delta_i)$. L'ensemble des δ_i représente alors la distribution $\int P(x|u_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$, mais la fonction d'importance selon laquelle ils sont répartis n'est pas celle que nous voulons (Figure 10.16).

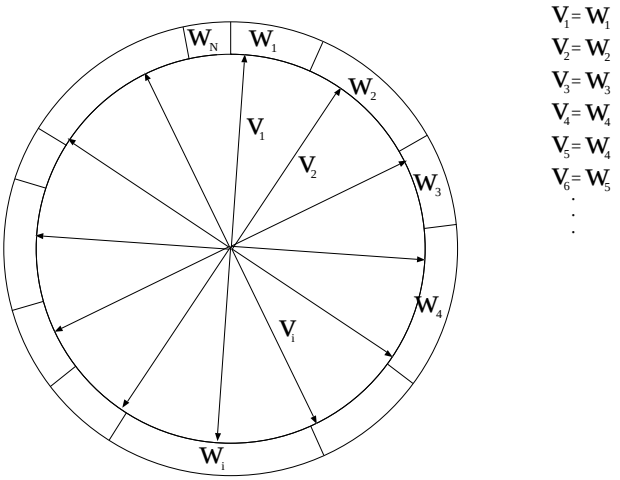


Figure 10.17: Illustration de la méthode d'échantillonage. L'ensemble de particules w est ré-échantillonné en l'ensemble v.

La seconde étape a pour but d'obtenir un échantillonnage selon la fonction d'importance $\int P(x|u_{t-1},x_{t-1})Bel(x_{t-1})dx_{t-1}$. Pour cela, il faut ré-échantillonner, ce qui peut se faire selon une méthode appelée *Stochastic universal sampling* [11]. Cette méthode consiste à choisir N particules δ_i selon des points équi-répartis sur un cercle sur lequel sont disposés l'ensemble initial des particules selon des secteurs angulaires de taille proportionnelle à leur poids (Figure 10.17). Cette méthode permet de multiplier les particules qui ont des poids élevés, et de réduire le nombre d'échantillons correspondant aux particules de poids faibles (Figure 10.18).

Algorithm 4 Procédure Filtrage Particulaire(θ_i^{t-1},w_i^{t-1},u,y);

for all i **do**
  Tirer une particule δ_i selon la loi $P(x|u,\theta_i^{t-1})$
end for
ré-échantillonner les particules δ_i selon les poids w_i^{t-1}.
$\eta = 0$
for all i **do**
  Calculer le poids $w_i^t = p(y|\delta_i)$
  $\eta = \eta + w_i^t$
end for
for all i **do**
  Normaliser les poids $w_i^t = w_i^t/\eta$
end for
Return $\theta_i^t = \delta_i$, w_i^t

En résumé, la localisation par filtrage particulaire s'effectue selon l'algorithme 4, et est illustré sur la figure 10.19. Intuitivement, l'algorithme permet de concentrer les particules dans les

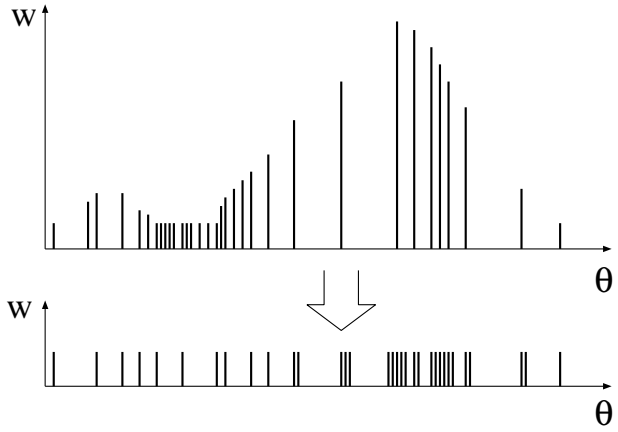


Figure 10.18: Illustration de l'effet de la méthode de ré-échantillonnage. Les particules ayant initialement un poids fort conduisent à des zones de particules plus denses. Les particules ayant initialement un poids faible conduisent à des zones de particules moins denses.

zones de plus fortes probabilités, où elles auront des poids plus fort. C'est un algorithme au final extrêmement simple et particulièrement robuste.

Cette version de base demande cependant un réglage relativement précis des paramètres, notamment des modèles probabilistes de perception et de l'odométrie pour donner des résultats satisfaisant.

Le nombre de particules utiles pour approcher correctement la probabilité de position peut également être relativement délicat à choisir. En effet, si il y a trop peu de particules, les zones de fortes probabilités risquent de ne pas être bien représentées et le ré-échantillonnage ne permettra pas de concentrer les particules autour de la position la plus probable, et le filtre ne convergera donc pas. On parle alors de *raréfaction des particules*.Ce problème est d'autant plus important pour la localisation globale, pour laquelle les particules sont initialisées aléatoirement sur tout l'environnement. Si il n'y a pas initialement de particule proche de la position réelle du robot, le filtre ne pourra pas la découvrir.

Augmenter le nombre de particules permet en général de résoudre le problème, mais augmente proportionnellement le temps de calcul, ce qui est rapidement problématique. Il existe donc de nombreuses variantes et améliorations de la méthode, qui permettent par exemple de sélectionner automatiquement le nombre de particules optimal pour fournir une approximation correcte de la position [50], ou de réaliser des mises à jour partielles pour respecter un temps de calcul limite [85].

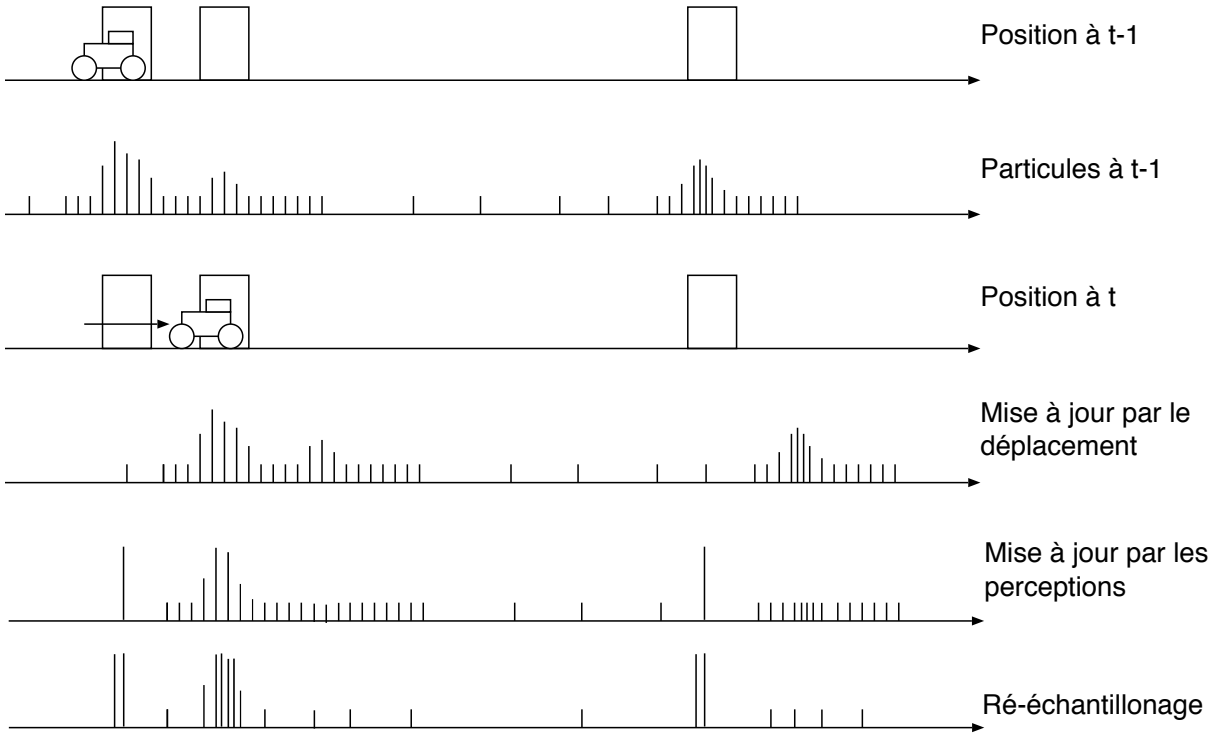


Figure 10.19: Illustration du fonctionnement de l'algorithme de filtrage particulaire. Les particules se concentrent autour des positions de plus forte probabilités.

10.5 Comparaison des méthodes de localisation

Le tableau de la figure 10.20 résume les grandes caractéristiques et compare les différentes méthodes de localisation présentées dans ce cours.

	Topologique hypothèse unique	Topologique hypothèses multiples	Métrique Filtre de Kalman	Métrique Multi Kalman	Métrique Filtres particulaires
Modèles de senseur	Quelconque	Quelconque	Gaussien	Gaussien	Quelconque
Capacité de localisation	Suivi	Global	Suivi	Global	Global
Consommation Mémoire	++	++	+	+	++
Consommation Calcul	+	++	+	++	++
Simplicité d'Implémentation	++	+	+	-	++
Précision	-	-	++	++	+
Robustesse	+	++	-	+	++

Figure 10.20: Comparaison des méthodes de localisation.

Les méthodes topologiques sont en général assez simples à implanter puisqu'il s'agit essentiellement de créer une procédure de comparaison de perceptions. Cependant, pour obtenir une robustesse correcte, cette procédure doit néanmoins être faite avec soin et peut devenir complexe. La localisation directe a partir des seules perceptions est difficile à envisager dans des environnements non triviaux. L'ajout de suivi multi-hypothèses dans ce cadre permet cependant d'améliorer la robustesse de manière assez simple. Les cartes topologiques conduisent cependant à une localisation relativement imprécise et demandent un certain nombre de comportements sensori-moteurs efficaces pour les déplacements entre lieux. Un des gros inconvénients des cartes topologiques, comme nous le verrons plus loin, se situe plutôt au niveau de leur création, qui peut être complexe à automatiser.

Concernant les méthodes métriques, le filtre de Kalman connaît un grand succès mais peut être difficile à régler et risque de diverger. L'utilisation de multi-Kalman permet d'améliorer sensiblement les choses et donne une très bonne précision, mais conduit à des implantations assez complexes et difficiles à maîtriser. Les filtres particulaires sont une bonne solution, en particulier pour leur robustesse et leur simplicité d'implantation, mais peuvent aussi être difficiles à régler et lourds en temps de calcul.

Chapter 11

Cartographie

Ce chapitre présente différentes méthodes de cartographie selon une classification personnelle en fonction de la capacité de ces méthodes à revenir ou non sur les données passées.

11.1 Les problèmes de la cartographie

11.1.1 Limitation des méthodes de localisation

Comme nous l'avons expliqué dans le chapitre 8 la cartographie est indissociable de la localisation, ce qui implique de disposer d'une méthode de localisation robuste pour espérer avoir une carte correcte. Ceci est le principal problème lors de la phase de cartographie, car une localisation correcte repose en général sur une bonne carte de l'environnement.

La tâche de cartographie est donc intrinsèquement plus complexe que celle de localisation. En effet, la localisation revient à rechercher, parmi les positions possibles représentées dans la carte, celle qui correspond le mieux à la position courante du robot. Cette recherche se déroule donc dans un espace fermé, car on postule que la position recherchée se trouve parmi les positions enregistrées dans la carte. Dans le cas de la cartographie, une difficulté importante provient du fait que l'estimation de la position du robot se déroule dans un espace ouvert, puisque le robot peut découvrir des zones encore inconnues. De plus cet espace est de faible dimension pour la localisation (3 en général), tandis que la construction de la carte se déroule dans un espace de beaucoup plus grande dimension. Par exemple, pour une carte de N amers 2D, il y aura $2N$ paramètres. Or que ce soit pour la cartographie ou la localisation, les données disponibles (odométrie et perceptions) restent les mêmes, et il faudra donc mieux les exploiter pour construire une carte.

L'incomplétude de la carte rend de plus la plupart des méthodes de localisation globale précédemment évoquées difficiles à utiliser car elles supposent une comparaison des probabilités des différentes hypothèses de position. Or avec une carte en cours de construction, le robot peut se situer dans une zone qui n'est pas encore cartographiée et il sera donc impossible d'évaluer la probabilité de cette position. La plupart des systèmes reposent donc sur une méthode de localisation qui réalise un suivi de position. En effet, si le robot atteint un lieu qui n'est pas

représenté dans la carte, il est possible, grâce à ces méthodes locales, de définir sa position par rapport à une position précédente connue au sein de la carte.

11.1.2 Fermetures de boucles

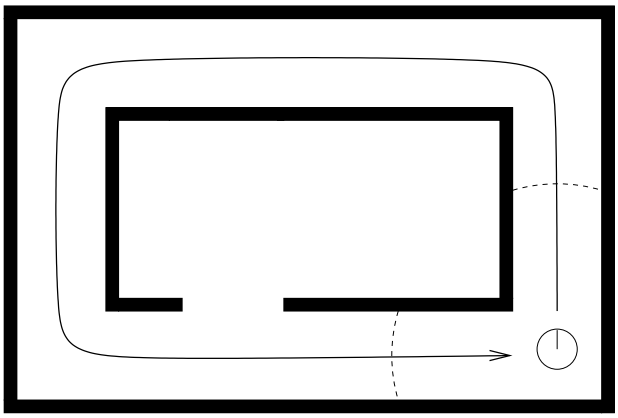


Figure 11.1: Un exemple simple d'environnement contenant un cycle.

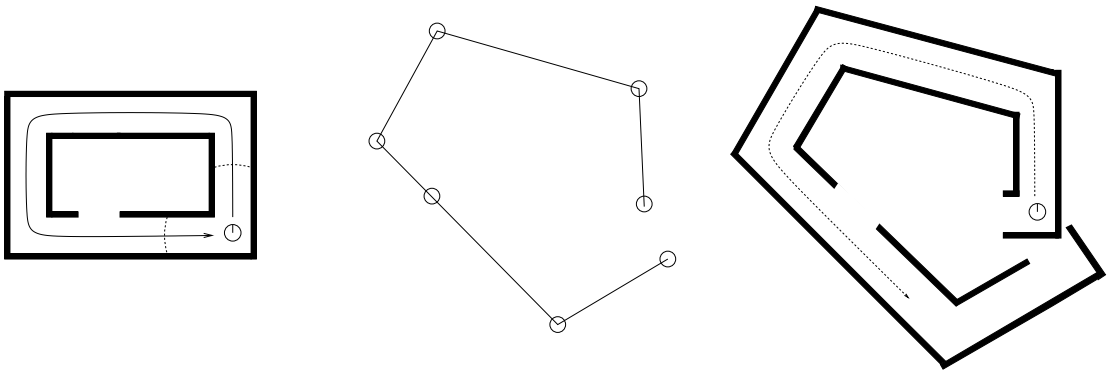
L'utilisation de systèmes de localisation effectuant un suivi de position lors de la cartographie pose cependant certains problèmes car, comme nous l'avons vu au chapitre précédent, ces méthodes peuvent diverger et conduire à une estimation erronée de la position sans possibilité de retrouver la position réelle du robot. Ce problème est particulièrement important au cours du processus de cartographie car les erreurs de localisation conduisent à des mises à jour erronées de la carte, ce qui peut conduire à des erreurs durables dans les futures tentatives de localisation.

Ces erreurs sont cruciales dans les environnements *cycliques*, c'est-à-dire dans des environnements contenant des boucles dont les différentes parties ne sont pas toutes visibles par les capteurs les unes à partir des autres (cf. figure 11.1). En effet, dans de tels environnements, les erreurs de cartographie durant le parcours du cycle peuvent empêcher la reconnaissance de la fermeture du cycle et conduire à des cartes à la topologie erronée (cf. figure 11.2). Il existe d'ailleurs des algorithmes spécialement adaptés à la détection de fermetures de boucles afin de pouvoir corriger ce type d'erreurs [1].

11.1.3 Cartographie incrémentale et retour en arrière

Un autre problème important de la cartographie est le choix de la représentation utilisée pour mémoriser les informations qui serviront par la suite au robot. Nous avons déjà présenté les réponses à ce problème en présentant les diverses structures de cartes, mais il faut garder à l'esprit que le choix de cette structure a une grande influence sur la qualité du processus de cartographie. Le choix de la représentation va déterminer la possibilité de faire ou non des mises à jour globales et efficaces lorsque de nouvelles informations sont disponibles.

On peut distinguer deux grandes catégories de méthodes de cartographie. La *cartographie incrémentale* [134] constitue une première méthode simple de construction de carte. Elle permet



Environnement réel Carte topologique Carte métrique

Figure 11.2: Lorsque la position du robot est estimée par une méthode de suivi de position, les erreurs s'accumulent pendant le parcours d'un cycle. Lors de la fermeture de ce cycle, le système peut alors être incapable de reconnaître la position initiale du robot, ce qui conduit a une carte incorrecte, dans le cas topologique (partie gauche) comme dans le cas métrique (partie droite).

simplement d'ajouter localement de nouvelles informations dans la carte à partir de l'estimation courante de la position du robot. Cependant, si cette estimation se révèle fausse a posteriori, il est impossible de revenir sur les modifications qui ont été effectuées. Cette limitation se révèle problématique dans les environnements contenant des cycles parce que la fermeture d'un tel cycle donne une information importante sur les erreurs des estimations précédentes de la position du robot. Cette information est ignorée par les systèmes de cartographie incrémentale et conduit, dans le cas d'environnements cycliques à des cartes dans lesquelles les erreurs vont se concentrer dans une petite zone (cf. figure 11.2 et 11.3).

La cartographie incrémentale constitue la méthode de base de nombreux systèmes utilisant des cartes topologiques. Cette méthode est également utilisée pour la création de grilles d'occupation que nous décriront plus loin.

La seconde catégorie de systèmes de cartographie regroupe les systèmes qui permettent d'intégrer des informations a posteriori sur les positions passées du robot (figure 11.3). Il faut noter que l'intégration de données a posteriori sur les positions passées est relativement simple dans les cartes topologiques grâce à la séparation des données proprioceptives et des perceptions que cette représentation implique. En effet, des erreurs dans l'estimation de la position du robot n'influent que sur les informations mémorisées dans les liens de la carte, et non sur les perceptions qui sont mémorisées dans les nœuds. Ainsi, revenir sur une modification passée de la carte, lorsque des nouvelles informations sur la position du robot sont disponibles, requiert simplement de modifier les informations mémorisées dans les liens et ne concerne pas les perceptions mémorisées. Dans le cas des cartes métriques, la modification d'une position passée va avoir des répercussions sur les perceptions (via le modèle métrique) et va donc rendre les modifications dues aux retours en arrière assez profondes. Pour permettre de tels retours en arrière, il faut alors utiliser un moyen de mémoriser les perceptions en les reliant à la position depuis laquelle elles ont été perçues. Cela peut se faire en mémorisant les perceptions indépendamment (des

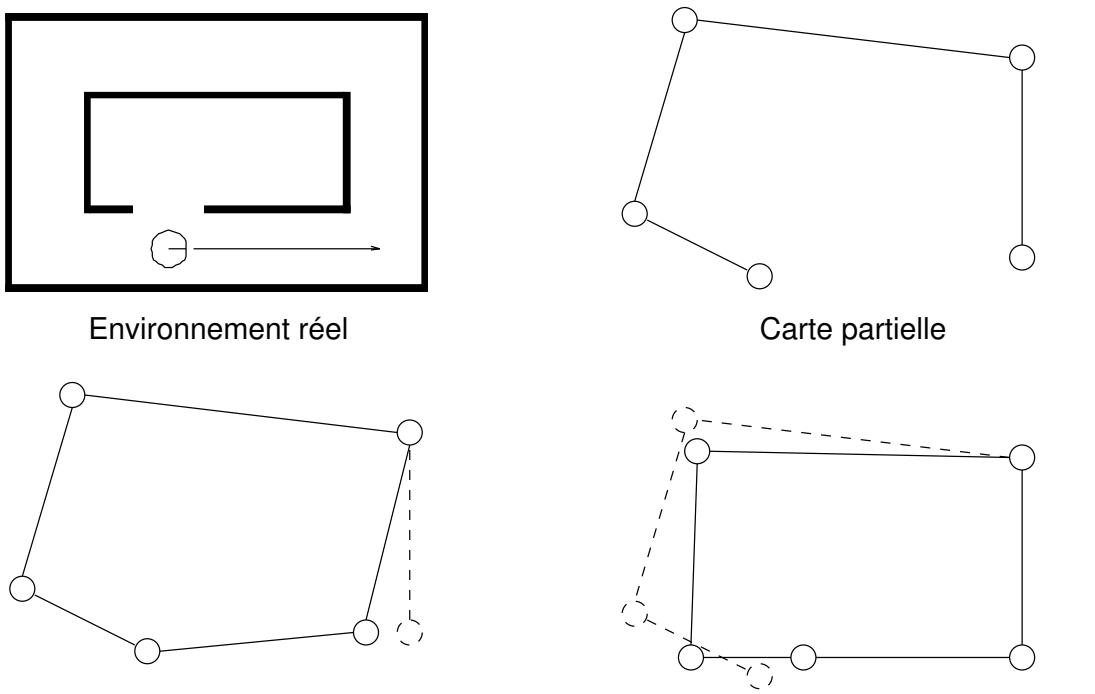


Figure 11.3: Illustration des capacités de cartographie sur une carte topologique. En bas à gauche, la cartographie incrémentale ne fera que des corrections locales lors d'une fermeture de boucles. En bas à droite, une méthode de cartographie avec retour en arrière pourra potentiellement corriger toute la carte pour réduire les erreurs.

scans lasers par exemple), mais n'est pas possible pour certaines représentations (notamment les grilles d'occupation).

11.2 Cartographie incrémentale

Les modèles de cartographie incrémentale utilisent donc une méthode de suivi de position pour estimer la position du robot par rapport à la carte existante. Au vu de cette position, si la carte ne représente pas le lieu ou les objets perçus par le robot, ceux-ci sont ajoutés à la carte. En revanche, si ce lieu est déjà représenté, la carte est adaptée en fonction des nouvelles perceptions.

11.2.1 Cartes Topologiques

Pour construire une carte topologique, nous commençons donc par estimer la position du robot. Dans les cartes topologiques sans *perceptual aliasing* [124, 25, 93, 51, 146, 55, 10], il suffit de comparer les perceptions courantes avec les données mémorisées dans chacun des nœuds de la carte. Si aucun des nœuds ne correspond suffisamment bien aux données courantes, cela signifie que le lieu n'est pas représenté dans la carte et qu'il devra donc être ajouté. Au contraire,

si la similitude des données courantes avec un nœud de la carte est suffisante, ce nœud sera reconnu comme la position courante du robot. Le choix entre ces deux alternatives est le point difficile du processus de cartographie. En effet, dans le cas de la simple localisation, il suffit de trouver le nœud correspondant le mieux à la situation courante pour trouver la position du robot. Dans le cadre de la cartographie, il faut de plus utiliser un seuil pour décider si le lieu le plus similaire est la position courante du robot ou non : si la similitude est supérieure à ce seuil, le nœud est reconnu, sinon un nouveau lieu est créé. Cette utilisation d'un seuil rend le processus de cartographie potentiellement plus instable que la localisation seule.

Lorsque le *perceptual aliasing* est pris en compte, percevoir un lieu différent de tous les lieux mémorisés dans la carte permet toujours de conclure que ce lieu est nouveau. Mais des perceptions qui correspondent à un lieu déjà mémorisé ne sont pas suffisantes pour déterminer si le lieu est nouveau ou s'il est connu car un lieu nouveau peut être similaire à un lieu déjà visité. La position précédente du robot doit donc être prise en compte pour déterminer si un lieu est nouveau ou s'il correspond à un nœud mémorisé. Si la position prédite par l'odométrie depuis le lieu précédent ne correspond pas au lieu déjà mémorisé, le lieu est considéré comme nouveau et ajouté à la carte [82, 43, 84, 145, 83, 142, 105, 12]. Certains modèles intègrent directement les informations perceptuelles et la position pour la reconnaissance des nœuds, se ramenant ainsi au cas où il n'y a pas de *perceptual aliasing* [6, 137, 33, 98].

Une fois le noeud reconnu ou créé, les perceptions sont utilisées pour corriger les données mémorisées dans ce nœud. Cela permet d'avoir une meilleure estimation des perceptions caractérisant le lieu grâce au filtrage du bruit sur ces données. Les données proprioceptives recueillies depuis le nœud précédent sont ensuite utilisées pour créer ou modifier l'arête qui joint le nœud précédent au nœud courant. Ce processus de cartographie est décrit par l'algorithme 5

Algorithm 5 Algorithme de cartographie topologique pour un déplacement u et des perceptions Y

1: **if** Il existe $noeud_i$ compatible avec u, Y et $Position_{t-1}$ **then**
2: $\quad Position_t = noeud_i$
3: **else**
4: $\quad Position_t$ = nouveau noeud
5: **end if**
6: Mettre à jour données de $Position_t$ avec Y
7: Mettre à jour la connexion $Position_{t-1}$ - $Position_t$ avec u

Comme nous l'avons mentionné précédemment, dans le cas où des informations métriques sont mémorisées entre les lieux, la carte obtenue peut alors se révéler incohérente. Dans les méthodes de cartographie incrémentale, la cohérence peut être assurée par l'association d'une position à chacun des nœuds de la carte [6, 137, 33, 98, 84], ou par une adaptation locale des valeurs des liens (figure 11.3). Dans le cas où ces valeurs seront ensuite simplement utilisées de manière locale[1], sans chercher à estimer les relations métriques entre lieux distants, le maintien de la cohérence peut être simplement négligé [82, 43].

[1] Par exemple pour guider un robot entre deux nœuds voisins via l'odométrie relative

11.2.2 Cartes métriques : corrélation de scan

Une première méthode simple de cartographie métrique incrémentale consiste à simplement utiliser une méthode de corrélation de scans lasers ou de cartes locales. Lorsque le robot progresse dans une zone encore non cartographiée, la corrélation est simplement effectuée entre la carte locale courante et la carte locale précédente. La nouvelle carte locale est ensuite ajoutée à la carte de l'environnement à sa position estimée (Figure 11.4).

Figure 11.4: Exemple de carte métrique créée par corrélation de scans. Chacun des cercles relié par une ligne indique le centre d'un scan ajouté à la carte (repris de [32]).

Lorsque le robot revient ensuite dans une zone déjà cartographiée après avoir parcouru une boucle, la corrélation peut être faite entre la carte locale courante et la portion de la carte globale la plus proche de la position actuelle (en non uniquement avec la carte locale précédente). Ceci permet de ré-estimer la position du robot dans la portion de carte déjà construite, mais ne corrige pas la carte le long de la boucle. Nous verrons plus loin (section 11.3.1) comment étendre ces méthodes pour corriger ces erreurs.

Cette méthode est en général suffisante lorsqu'elle est utilisées avec les données d'un télémètre laser et dans un environnement de taille limitée ne contenant pas de grands cycles. Lors de la fermeture de grands cycles, par contre, elle montre rapidement ses limites car les erreurs de localisation ne permettent pas de trouver la bonne portion de carte avec laquelle faire la corrélation. La méthode échoue alors, mais il est possible d'ajouter des procédures spécifiques qui font une recherche globale lorsqu'un cycle semble être fermé et permettent de rattraper ces échecs [64].

11.2.3 Cartes métriques : grilles d'occupation

Une autre méthode de cartographie incrémentale très populaire est la construction de grille d'occupation. Rappelons que les grilles d'occupation sont une représentation de l'environnement dans laquelle l'espace est discrétisé en cellules régulières et dont chaque cellule a une probabilité associée d'être occupée par un obstacle (Figure 11.5). Le construction de grille d'occupation est l'une des premières méthodes de cartographie a avoir été développée.

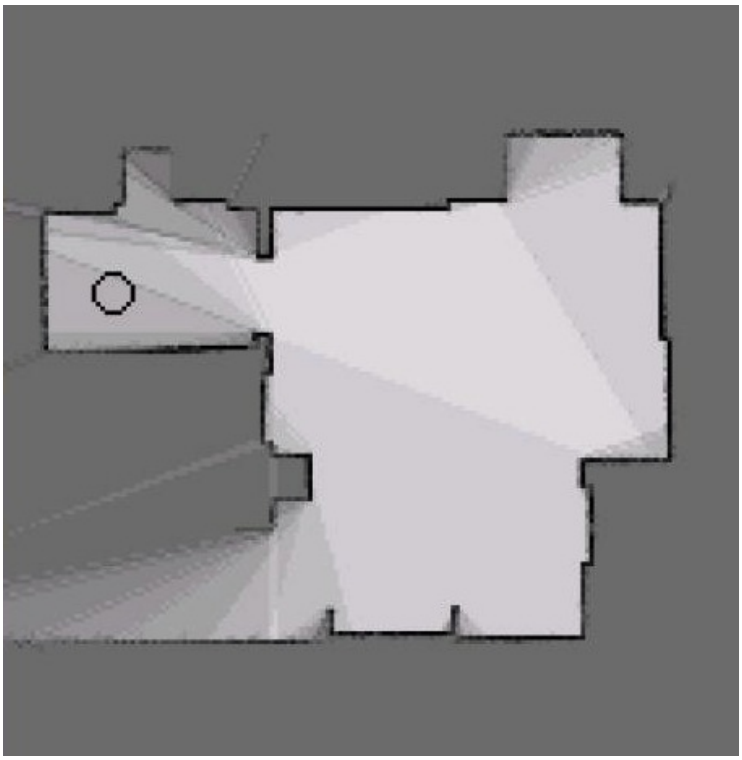


Figure 11.5: Exemple de grille d'occupation crée à partir de mesures d'un télémètre laser.

Dans les premiers travaux de leurs créateurs, Moravec et Elves [101, 131], la construction de grilles d'occupation supposait la position du robot connue. Dans des développements ultérieurs, toutefois, la grille en cours de construction est utilisée pour estimer la position du robot. On utilise pour cela des techniques de mise en correspondance de grilles similaires à celles présentées dans la section 10.2.3. Ces méthodes de recalage permettent de limiter la dérive de l'odométrie, mais ne sont en général pas suffisantes pour garantir une cartographie correcte dans les environnement cycliques. Pour résoudre ce problème, il est possible d'utiliser des hypothèses sur l'environnement, afin de permettre le recalage de l'odométrie durant la cartographie. L'hypothèse la plus couramment utilisée suppose que les murs de l'environnement sont rectilignes et orthogonaux [131, 83], ce qui permet de corriger facilement l'estimation de la direction du robot.

Après avoir estimé la position du robot, les valeurs des différentes cellules de la grille d'occupation sont mises à jour en fonction des perceptions. Pour cela, nous disposons d'un modèle des capteurs $P(occ_i|s)$ qui, pour une perception s donnée, nous fournit la probabilité

d'occupation des cellules dans le champ de vision du capteur en fonction de la valeur renvoyée par le capteur. Ce modèle probabiliste est en général similaire à ceux présentés dans la section 4.2.2 pour la distance au capteur et utilise une erreur gaussienne pour l'écart entre la direction de la cellule et la direction du capteur.

Nous cherchons donc à accumuler ces mesures pour estimer la probabilité pour la cellule i d'être occupée en fonctions de toutes mesures précédentes : $P(occ_i^T) = P(occ_i|s_1,...,s_T)$. Pour cela, nous utilisons, comme souvent, la formule de Bayes pour extraire la probabilité de la dernière mesure en fonction des autres variables :

$$P(occ_i^T) = \frac{P(s_T|occ_i,s_1,...,s_{T-1})P(occ_i|s_1,...,s_{T-1})}{P(s_T|s_1,...,s_{T-1})}$$

En faisant l'hypothèse que le monde est statique, qui se traduit par le fait que les mesures sont conditionnellement indépendantes si l'on connaît la valeur de la cellule de la carte, nous pouvons simplifier :

$$P(occ_i^T) = \frac{P(s_T|occ_i)P(occ_i^{T-1})}{P(s_T|s_1,...,s_{T-1})}$$

notre modèle de capteur nous donnant $P(occ_i|s_T)$, nous le faisons apparaître en utilisant Bayes une nouvelle fois :

$$P(occ_i^T) = \frac{P(occ_i|s_T)P(s_T)}{P(occ_i)}\frac{P(occ_i^{T-1})}{P(s_T|s_1,...,s_{T-1})}$$

Dans cette équation nous ne connaissons pas $P(s_T)$ ni $P(s_T|s_1,...,s_{T-1})$. Pour pouvoir estimer $P(occ_i^T)$, nous allons utiliser le fait que la valeur d'occupation est binaire. Nous commençons par calculer la probabilité que la cellule soit vide $P(\bar{occ}_i^T)$, ce qui se fait de la même manière :

$$P(\bar{occ}_i^T) = \frac{P(\bar{occ}_i|s_T)P(s_T)}{P(\bar{occ}_i)}\frac{P(\bar{occ}_i^{T-1})}{P(s_T|s_1,...,s_{T-1})}$$

et nous permet en utilisant le rapport des deux valeurs, de se débarrasser des termes gênants.

$$\frac{P(occ_i^T)}{P(\bar{occ}_i^T)} = \frac{P(occ_i|s_T)}{P(\bar{occ}_i|s_T)}\frac{P(\bar{occ}_i^{T-1})}{P(occ_i^{T-1})}\frac{P(\bar{occ}_i)}{P(occ_i)}$$

et en utilisant le fait que $P(\bar{occ}_i) = 1 - P(occ_i)$, nous obtenons :

$$\frac{P(occ_i^T)}{1-P(occ_i^T)} = \frac{P(occ_i|s_T)}{1-P(occ_i|s_T)}\frac{1-P(occ_i^{T-1})}{P(occ_i^{T-1})}\frac{1-P(occ_i)}{P(occ_i)}$$

il est alors possible d'extirper $P(occ_i^T)$ de cette expression, mais il est plus simple de chercher à estimer directement la valeur l_i définie par :

$$l_i^T = log\left(\frac{P(occ_i^T)}{1-P(occ_i^T)}\right)$$

qui se calcule simplement par :

$$l_i^T = log\left(\frac{P(occ_i|s_T)}{1-P(occ_i|s_T)}\right) + log\left(\frac{1-P(occ_i)}{P(occ_i)}\right) + l_i^{T-1} \tag{11.1}$$

Ceci nous fournit une règle simple de mise à jour incrémentale des valeurs l_i^T en fonction des valeurs précédentes et du modèle de capteur. La valeur $P(occ_i)$ est une valeur initiale de la probabilité d'occupation. Elle est en général choisie égale à 0,5 mais peut être plus faible ou plus forte si l'on souhaite intégrer un a priori sur le fait que l'environnement contient une densité plus ou moins grande d'obstacles. On retrouve ensuite la probabilité d'occupation par : $p(occ_i^T) = 1 - 1/e^{l_i^T}$

Il existe également des moyens plus simples de mise à jour des grilles, qui consistent simplement à estimer la probabilité d'occupation d'une cellule en fonction du nombre de perceptions de cette cellule qui ont eu lieu. Ainsi, si on note $occ(x,y)$ le nombre de fois où un obstacle a été détecté dans cette cellule et $vide(x,y)$ le nombre de fois où cette cellule est apparue vide car le faisceau du télémètre a traversé cette cellule, on peut simplement estimer la probabilité par :

$$P_{occ}(x,y) = \frac{occ(x,y)}{vide(x,y) + occ(x,y)}$$

Une variante encore plus simple pour créer une grille d'occupation est une mise à jour utilisant un simple décompte dans lequel on ajoute une certaine valeur fixe à la probabilité d'occupation d'une cellule si un obstacle y est détecté et on retranche une autre valeur fixe si la cellule a été traversée par le faisceau [78] (voir aussi la section 6.2). Cette méthode, nommée "histogrammic in motion mapping (HIMM)" présente cependant l'inconvénient de ne pas converger vers une valeur fixe lorsque le nombre de perceptions d'une cellule tend vers l'infini. Elle est de plus relativement sensible au bruit et suppose des réglages délicats de paramètres pour être adaptée à un robot spécifique.

11.2.4 Stratégies d'exploration

Pour limiter les erreurs de cartographie dans ce types de méthodes, il est possible d'utiliser une exploration active de l'environnement, plutôt que de passivement mémoriser les données recueillies par le robot. Dans le cadre des cartes topologiques, il est par exemple possible, lorsqu'un nœud a été reconnu, de chercher à atteindre un des nœuds voisins mémorisé dans la carte [82]. Si ce second nœud est correctement détecté, il permet de confirmer la détection du nœud précédent qui est alors mis à jour. D'autre modèles utilisent une exploration active de l'environnement pour diriger le robot vers les zones pour lesquelles l'incertitude de la carte est grande dans le but de la réduire [42]. Enfin certains modèles sont capables de générer des hypothèses sur

des portions non visitées de l'environnement qui sont ensuite vérifiées grâce à une procédure qui dirigera le robot dans les zones où de telles hypothèses ont été faites [83]. Des stratégies d'explorations peuvent également être utilisées afin de garantir un exploration rapide et exhaustive de l'environnement [131, 83, 147].

D'une manière générale, de telles procédures permettent donc, d'une part, de limiter les erreurs de cartographie en insistant sur les zones incertaines et en évitant que l'estimation de la position ne devienne trop mauvaise et, d'autre part, de garantir une exploration exhaustive de l'environnement.

11.3 Retour sur les modifications passées

Dans les méthodes de cartographie précédentes, les modifications apportées à la carte au cours du processus de cartographie se font donc en supposant que l'estimation de la position du robot au moment de cette modification est correcte. Or cette estimation se révèle en général fausse, ou entachée d'erreur, a posteriori. Dans ce cas, les modifications de la carte ont été effectuées de manière incorrecte, et il serait souhaitable de pouvoir revenir sur ces modifications pour prendre en compte les nouveaux indices sur les positions passées du robot. La plupart des modèles de cartographie utilisés actuellement, que nous présentons dans ce paragraphe, en sont capables.

11.3.1 Méthodes de relaxation

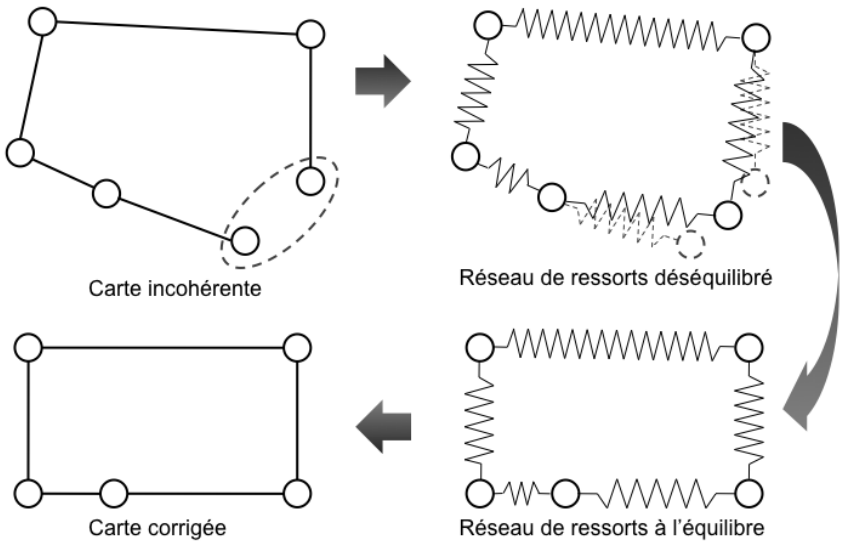


Figure 11.6: Illustration du principe des méthodes de relaxation.

Une première méthode pour prendre en compte la nouvelle information de position est de propager l'erreur tout le long du cycle parcouru par le robot. Cette méthode peut s'appliquer à des cartes topologiques ou à des cartes de scans (voir section 11.2.2). Intuitivement, cela

correspond à identifier la carte à un réseau de ressorts, dont les positions au repos correspondent aux positions relatives des nœuds (figure 11.6). L'erreur lors de la fermeture de boucle correspond alors à un déséquilibre dans ce réseau de ressort. Pour obtenir une carte globalement cohérente et qui respecte au mieux les positions relatives des nœuds, il suffit de calculer la position de repos du réseau de ressorts (d'ou le terme de méthodes de relaxation).

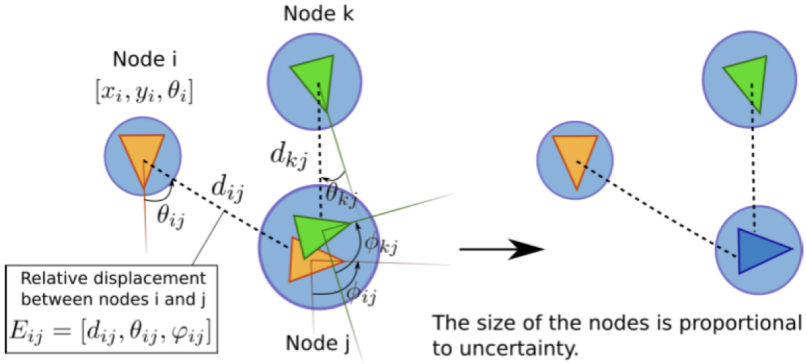


Figure 11.7: Principe de la méthode de relaxation itérative de [35, 2].

Il existe différents algorithmes de relaxation. Nous présentons ici un algorithme itératif très simple qui donne de bons résultats [35, 2]. Le principe de cet algorithme est de calculer la position de chaque nœud à partir de celle de ses voisins itérativement jusqu'à convergence des positions (figure 11.7). Ainsi pour chaque nœud i, les trois étapes suivantes sont appliquées :

- Estimation de la position du nœud i depuis chaque voisin j:

$$(x'_i)_j = x_j + d_{ji}\cos(\theta_{ji} + \theta_j) \quad (11.2)$$
$$(y'_i)_j = y_j + d_{ji}\sin(\theta_{ji} + \theta_j) \quad (11.3)$$
$$(\theta'_i)_j = \theta_j + \varphi_{ji} \quad (11.4)$$

  et estimation de la variance du nœud i depuis le nœud j:

$$(v'_i)_j = v_j + v_{ji}$$

- Estimation de la variance du nœud i par la moyenne harmonique des estimations depuis les voisins:

$$v_i = \frac{n_i}{\sum_j \frac{1}{(v'_i)_j}} \quad (11.5)$$
$$(11.6)$$

  où n_i est le nombre de voisins du nœud i.

- Estimation de la position du nœud comme la moyenne pondérée des estimations depuis ses voisins :

$$x_i = \frac{1}{n_i}\sum_j \frac{(x'_i)_j v_i}{(v'_i)_j} \tag{11.7}$$

$$y_i = \frac{1}{n_i}\sum_j \frac{(y'_i)_j v_i}{(v'_i)_j} \tag{11.8}$$

$$\theta_i = \arctan\left(\frac{\sum_j \frac{\sin((\theta'_i)_j)}{(v'_i)_j}}{\sum_j \frac{\cos((\theta'_i)_j)}{(v'_i)_j}}\right) \tag{11.9}$$

$$\tag{11.10}$$

La figure 11.8 montre un exemple de l'application de cet algorithme à une carte topo-métrique. Les nœuds entourés montrent les fermetures de boucles, c'est à dire les retours détectés à une position précédente qui sont utilisés pour contraindre le graphe.

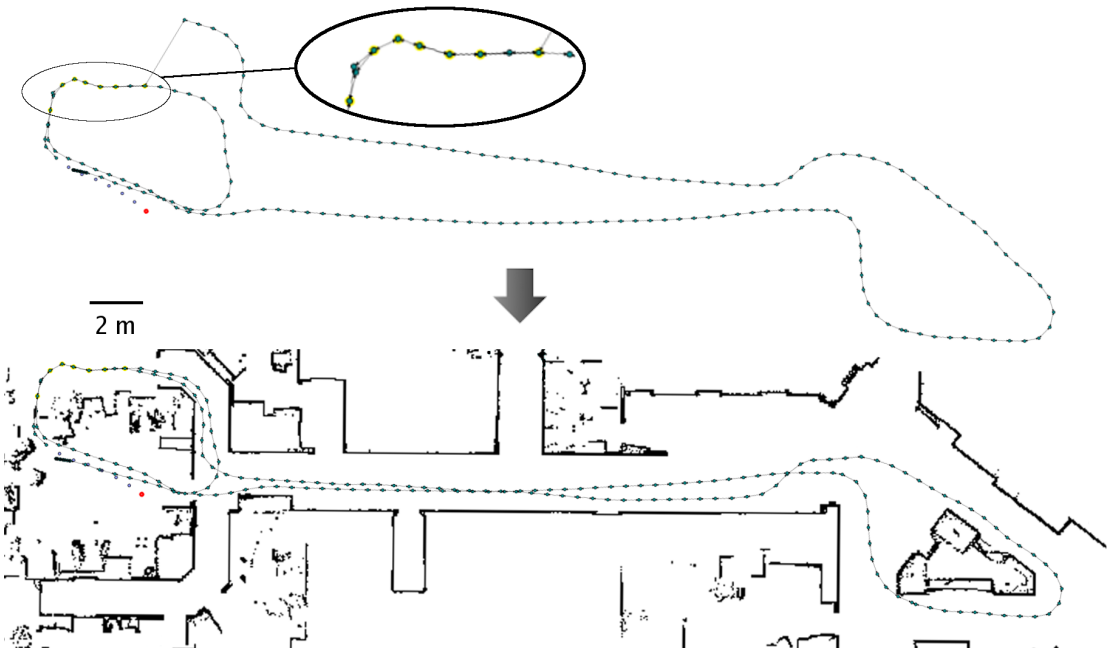


Figure 11.8: Exemple de carte topo-métrique obtenue par vision pour la détection de fermeture de boucle et relaxation, superposée à une carte métrique de référence obtenue par corrélation de scans lasers (tiré de [2]).

Cet algorithme donne de bons résultats, mais peut être assez lent à converger. En effet, si l'on applique la mise à jours à tous les nœuds dans un ordre fixe (l'ordre de création des nœuds par exemple), l'erreur qui est localisée sur une seule arête au départ ne sera propagée que sur un voisin supplémentaire à chaque tour. Pour accélérer la convergence, il est possible de simplement ré-ordonner la mise à jour des nœuds afin que l'erreur se diffuse plus rapidement.

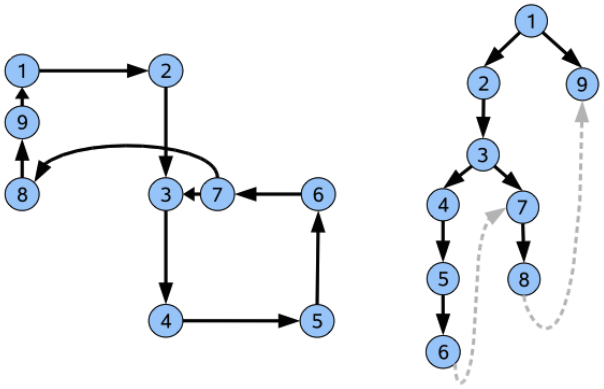


Figure 11.9: Illustration de la construction de l'arbre associé à un graphe de l'algorithme TORO (tiré de [61]). L'ordre de mise à jour optimal des nœuds pour la relaxation est donné par le parcours de l'arbre : 1, 2, 9, 3, 4, 7 ...

C'est ce que propose par exemple l'algorithme TORO (Tree-based netwORk Optimizer) [61] qui construit un arbre associé au graphe de la carte pour déterminer un ordre optimal de mise à jour. Cet arbre est construit simplement en ajoutant chaque nouveau nœud comme fils du plus ancien de ses voisins (figure 11.9). La méthode de mise à jour de la position des nœuds est ensuite appliquée dans l'ordre donné par le parcours descendant de l'arbre. Ainsi dans l'exemple de la figure 11.9, l'erreur[2] initialement concentrée sur l'arête entre les noeuds 1 et 9 sera propagée lors du premier parcours de la carte aux nœuds 9 et 2, puis 3, 4, 7. Avec une mise à jour dans l'ordre de création des nœuds, la propagation aurait simplement été faite aux nœuds 2, 3, 4... La première mise à jour n'aurait donc modifié que la position du nœuds 9 et aurait été nulle pour les autres.

11.3.2 Cartographie par filtrage de Kalman étendu

Le filtrage de Kalman étendu (section 10.3.3) peut aussi être utilisé pour la cartographie. C'est cette méthode qui était à l'origine présentée sous le terme SLAM [129, 91]. Le terme SLAM est depuis devenu plus générique et la cartographie par filtrage de Kalman étendu se retrouve souvent sous le nom "EKF SLAM".

Le filtrage est ici utilisé non seulement pour estimer la position du robot, mais aussi pour estimer la position des différents éléments enregistrés dans la carte. Cette méthode est en général utilisée avec des cartes représentant l'environnement sous forme d'objets géométriques simples tels que des points ou des segments. Elle va permettre d'utiliser les relations mesurées entre le robot et ces objets pour estimer leurs différentes positions.

Plus généralement, l'idée est donc d'utiliser un filtrage Bayésien estimant à la fois les positions du robot (x_t) et des objets de la carte (c_t):

[2] Définie comme l'écart entre la position relative des nœuds enregistrée dans l'arête et la position relative calculée par la différence de leur position absolue

$$Bel(x_t, c_t) = \eta p(y_t|x_t, c_t) \int\int p(x_t, c_t|x_{t-1}, c_{t-1}, u_{t-1}) Bel(x_{t-1}, c_{t-1}) dx_{t-1} dc_{t-1}$$

Cette équation peut être simplifiée en utilisant l'hypothèse de monde statique, qui est utilisée dans la plupart des modèles, et qui entraîne donc que la carte de l'environnement reste la même au cours du temps ($c_t = c_{t-1}$). Il faut bien distinguer cette hypothèse du fait que notre estimation de la probabilité des différentes cartes ($Bel(x_t, c_t)$) évolue au cours du temps, et donc que la carte que nous supposons représenter l'environnement (la plus probable en général) change au cours du temps. En utilisant cette hypothèse, l'équation du filtrage devient alors :

$$Bel(x_t, c_t) = \eta p(y_t|x_t, c_t) \int\int p(x_t|x_{t-1}, u_{t-1}) Bel(x_{t-1}, c_{t-1}) dx_{t-1} dc_{t-1}$$

Comme pour la localisation, ce filtrage peut être implémenté de différentes manières, par exemple en utilisant un filtre de Kalman ou un filtrage particulaire (voir section suivante). Il peut également être utilisé avec différentes formes de cartes, qui rendent son implantation plus ou moins aisée. La technique communément nommée EKF SLAM consiste en l'implantation de ce filtrage sous la forme d'un filtre de Kalman étendu, en utilisant une carte contenant des amers qui peuvent être des points ou des segments. C'est historiquement la première implantation de ce type de cartographie [129].

Détaillons a titre d'exemple une implantation de cette méthode utilisant une carte contenant des amers ponctuels. Dans un état où la carte contient N amers, on utilise un vecteur d'état contenant la position du robot et des différents amers. Le vecteur d'état est donc de dimension 2N+3 :

$$Bel(x_t, c_t) = \begin{bmatrix} x \\ y \\ \theta \\ x_{a1} \\ y_{a1} \\ \vdots \\ x_{aN} \\ y_{aN} \end{bmatrix}$$

La matrice de covariance associée permet de mémoriser les relations qui ont été perçues entre les amers et entre les amers et le robot. C'est l'utilisation de ces covariances lors de la mise à jour qui permettra un retour sur les modifications passées au sens ou elle permettra de propager toute nouvelle information de position du robot vers les éléments dont la position relative avec le robot est connue (càd les éléments avec lesquels la covariance est non nulle).

Supposons que l'on commande les vitesses de rotation et de translation du robot $u = (v, \omega)$, l'équation d'évolution ne modifiera que la position du robot et sera alors :

$$f(X_t,u) = \begin{bmatrix} x+v.dt.cos(\theta) \\ y+v.dt.sin(\theta) \\ \theta+\omega.dt \\ x_{a1} \\ y_{a1} \\ \vdots \\ x_{aN} \\ y_{aN} \end{bmatrix}$$

Ce qui donne la matrice jacobienne :

$$A = \begin{bmatrix} 1 & 0 & -sin(\theta) & 0 & \dots & 0 \\ 0 & 1 & cos(\theta) & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & \dots & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & \dots & 0 \end{bmatrix}$$

enfin si l'on suppose que lors de la perception de l'amer i, on mesure sa position relative dans le repère lié au robot en coordonnées polaires (r_i, ϕ_i), l'équation d'observation sera :

$$h_i(X_t) = \begin{bmatrix} \sqrt{(x-x_{ai})^2+(y-y_{ai})^2} \\ atan2(\frac{y_{ai}-y}{x_{ai}-x}) - \theta \end{bmatrix}$$

la matrice jacobienne de l'équation d'observation s'écrira alors :

$$H_i = \begin{bmatrix} \frac{(x-x_{ai})}{\sqrt{(x-x_{ai})^2+(y-y_{ai})^2}} & \frac{(y-y_{ai})}{\sqrt{(x-x_{ai})^2+(y-y_{ai})^2}} & 0 \\ \frac{(y-y_{ai})}{(x-x_{ai})^2+(y-y_{ai})^2} & -\frac{(x-x_{ai})}{(x-x_{ai})^2+(y-y_{ai})^2} & -1 \end{bmatrix}$$

Pour estimer à la fois la position des amers et du robot, il suffira alors de dérouler les équations du filtrage de Kalman étendu :

- **Prédiction de l'état** :

$$X_t^* = f(\hat{X}_{t-1}, u_t) \tag{11.11}$$

  et de la covariance:

$$P_t^* = A.\hat{P}_{t-1}.A^T + Q \tag{11.12}$$

- **Prédiction de l'observation** :

$$Y_t^* = h(X_t^*) \tag{11.13}$$

- **Observation** : on obtient une mesure Y, dont on estime le bruit P_Y grâce au modèle du processus de perception.

- **Correction de l'état prédit**:

$$\hat{X}_t = X_t^* + K(Y - Y_t^*) \tag{11.14}$$

$$\hat{P}_t = P_t^* - KHP_t^* \tag{11.15}$$

ou K est le gain de Kalman:

$$K = P_t^* H^T .(H.P_t^* .H^T + P_Y)^{-1} \tag{11.16}$$

Cette mise à jour suppose que l'amer perçu soit déjà dans la carte et donc dans le vecteur d'état. Si l'amer détecté par le robot n'est pas dans la carte, il est simplement ajouté à la fin du vecteur d'état et toutes les matrices sont agrandies de 2 lignes (et éventuellement colonnes) correspondantes.

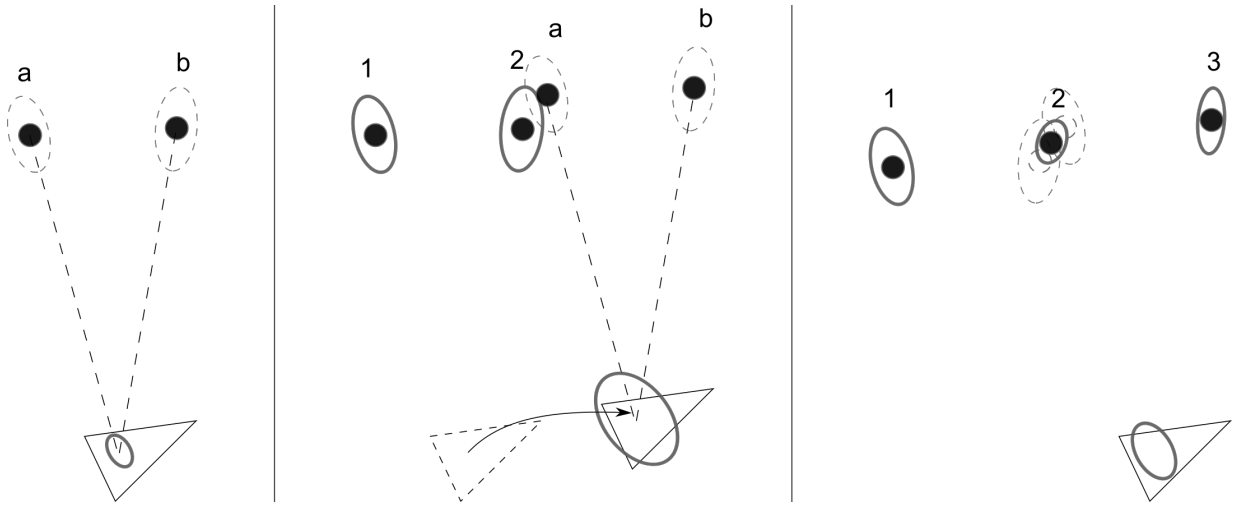


Figure 11.10: Illustration de l'algorithme de SLAM par filtrage de Kalman. Dans la partie A, le robot, dans sa position initiale, perçoit les amers 1 et 2 qui sont ajoutés à la carte. Dans la partie B, le robot s'est déplacé (on dispose d'une estimation bruitée de sa position par l'odométrie) et il perçoit les deux amers 3 et 4. Dans la partie C, on a considéré que les amers 2 et 3 étaient le même amer, ce qui a permis de mieux estimer sa position et de corriger l'estimation de la position du robot. L'amer 4 a été ajouté à la carte.

Il reste un problème important dans la méthode décrite car elle suppose que l'on soit capable d'identifier les amers, c'est à dire de savoir à quel amer i correspond un amer perçu. Or, en utilisant de simples points ou segments de droites, cela se révèle difficile à cause du très fort perceptual aliasing. Comme pour la localisation, il y a deux méthodes possibles pour résoudre ce problème. La première est de particulariser chaque amer, par exemple en utilisant une image de l'environnement autour de l'amer afin de pouvoir l'identifier individuellement. Cette méthode est rarement suffisante car avec l'augmentation de la taille de l'environnement on retombe en général sur le problème de perceptual aliasing. La seconde solution repose sur des amers indistinguables

et sélectionne l'amer correct en se basant sur sa position estimée. Pour cela, on estime la position absolue de l'amer perçu à partir de la position estimée du robot, puis on considère qu'il correspond à l'amer de la carte dont la position est la plus proche. Cette mise en correspondance peut également utiliser une distance de Mahalanobis (voir section 10.3.3), ou utiliser des critères supplémentaires qui caractérisent les amers. Si les amers sont des segments, par exemple, on peut chercher le segment le plus proche qui ait la même direction que le segment perçu.

L'algorithme 6 décrit finalement le processus complet de cette méthode de cartographie. Cette méthode est également illustrée graphiquement dans la figure 11.10.

Algorithm 6 Algorithme de SLAM pour un déplacement u et un amer perçu Y

Prédiction de l'état $X_t^* = f(\hat{X}_{t-1}, u_t)$
Estimation de la position et de la variance de l'amer perçu $x_Y, y_Y, \sigma_{x_Y}, \sigma_{y_Y}$
for all Amer i **do**
    Calcul de la distance de Mahalanobis $d_i^2 = \frac{1}{2}(Y - Y_i)^T (P_Y + P_{Y_i})^{-1} (Y - Y_i)$
end for
Sélection de l'amer j de la carte tel que d_j soit minimal
if $d_j <$ Seuil : l'amer est déjà dans la carte **then**
    Prédiction de l'observation $Y_{j_t}^* = h_j(X_t^*)$
    Correction de l'état prédit $\hat{X}_{t+1} = X_t^* + K(Y - Y_{j_t}^*)$
else
    ajout de l'amer perçu à la carte
end if

La principale faiblesse de cette méthode réside dans la phase d'appariement qui peut entraîner une divergence de l'algorithme en cas d'erreur. En effet, dans ce cas, non seulement la position de l'amer apparié par erreur sera mal estimée, mais cette mauvaise estimation sera propagée aux autres amers, via la matrice des covariances, ainsi qu'a la position du robot. Cette mauvaise estimation de la position du robot produira par la suite de nouvelles erreurs d'appariement qui se succéderont et entraîneront la divergence complète de l'algorithme.

Un autre inconvénient est la complexité des calculs requis, qui augmente en $O(N^2)$ avec le nombre d'amers de la carte. Ceci est lié à la taille de la matrice de covariances qui est $N \times N$. Or cette matrice mémorise les interrelations entre les positions des amers et du robot et est nécessaire à la bonne estimation de ces positions.

Pour réduire la complexité des calculs, il est cependant possible de réduire la taille de cette matrice en négligeant certaines interrelations entre amers. Ceci peut se faire par exemple en ne mémorisant les covariances qu'au sein de cartes locales, dont l'assemblage couvre l'ensemble de l'environnement. Diverses méthodes existent, utilisant différentes manières de découper la carte globale et de propager les informations entre cartes locales (par exemple [109]).

11.3.3 Fast SLAM

La méthode d'appariement des amers utilisée dans la méthode du SLAM EKF est à la source de la plupart des problèmes de cette méthode. Elle repose en effet sur l'hypothèse que l'estimation

de la position du robot est approximativement correcte pour fournir un appariement correct. On peut donc remarquer que si la position du robot est parfaitement connue, le problème se simplifie énormément car la mise en correspondance ne pose plus de problème et il reste simplement à estimer correctement la position des amers, ce qui est relativement trivial (sous l'hypothèse de la connaissance de la position du robot).

Cette remarque a inspiré la méthode de cartographie nommée Fast SLAM [100, 70, 60]. Dans cette méthode, on mémorise un grand nombre de trajectoires possibles, dont on évalue la probabilité, et pour ces trajectoires, on construit simplement la carte à partir des perceptions. Cet ensemble de trajectoires correspond en fait à un échantillonage de la distribution de probabilité sur ces trajectoires au moyen d'un filtre particulaire . Pour la construction de la carte à partir de l'une de ces trajectoires, on utilise le filtrage de Kalman, mais dans des conditions beaucoup plus favorables. En effet, comme la trajectoire est connue, les perceptions successives des différents amers deviennent indépendantes et il n'est plus nécessaire de mémoriser l'ensemble des covariances, mais seulement les variances des amers individuels. On passe ainsi d'une complexité en $O(N^2)$ à une complexité en $O(N)$ pour la partie filtrage de Kalman. Il faut cependant y ajouter la complexité du filtrage particulaire sur les trajectoires, mais globalement, l'algorithme est plus stable et moins gourmand en calculs que le SLAM original.

Cette méthode de séparation du problème de SLAM et un filtre particulaire pour la partie non gaussienne et en un filtre de Kalman pour une partie du sous problème est une méthode très générale connue sous le nom de *Rao-Blackwellisation* du filtre.

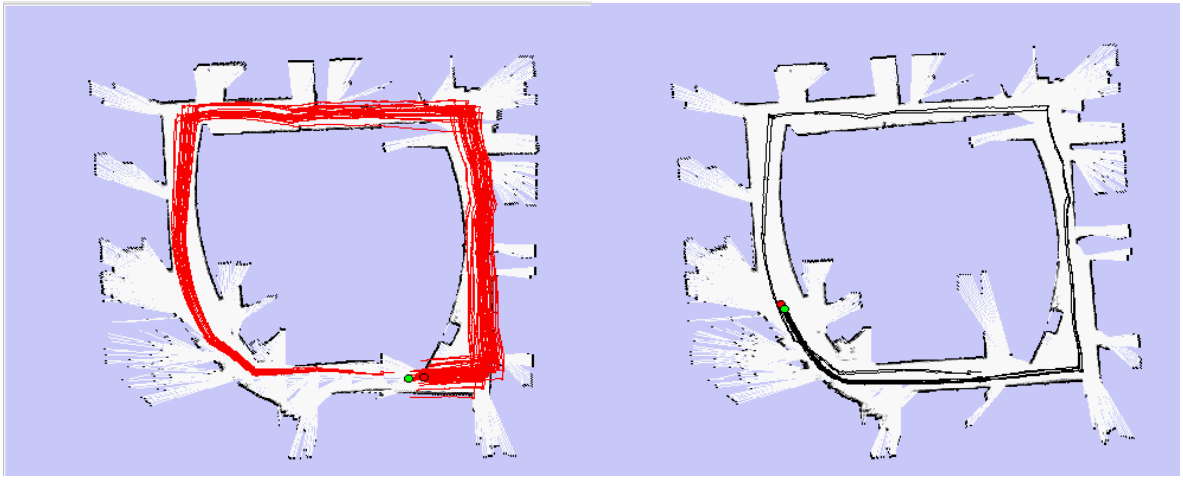


Figure 11.11: Avant la fermeture d'un cycle, un très grand nombre de trajectoires sont évaluées. Après fermeture, le filtre particulaire sur les trajectoires sélectionne automatiquement celle correspondant effectivement au cycle (image tirée de [70]).

La force de ces méthodes est surtout apparente lors de la fermeture de grands cycles. En effet, avant la fermeture, un grand nombre de trajectoires différentes sont mémorisées, ce qui permet d'avoir une forte probabilité que l'une d'elle corresponde à la trajectoire précise du cycle. Après la fermeture du cycle, le ré-échantillonage du filtre particulaire sélectionne naturellement les trajectoires correctes et permet ainsi une cartographie correcte du cycle (Figure 11.11). Ce

ré-échantillonage entraine cependant une forte perte d'information, puisque la plupart des trajectoires estimées avant la fermeture de boucle sont ignorées (c'est le problème classique de raréfaction des particules des méthodes de filtrage particulaire, voir section 10.4.4). Cette perte d'information devient problématique dans des environnements contenant plusieurs cycles, notamment dans le cas de cycles imbriqués.

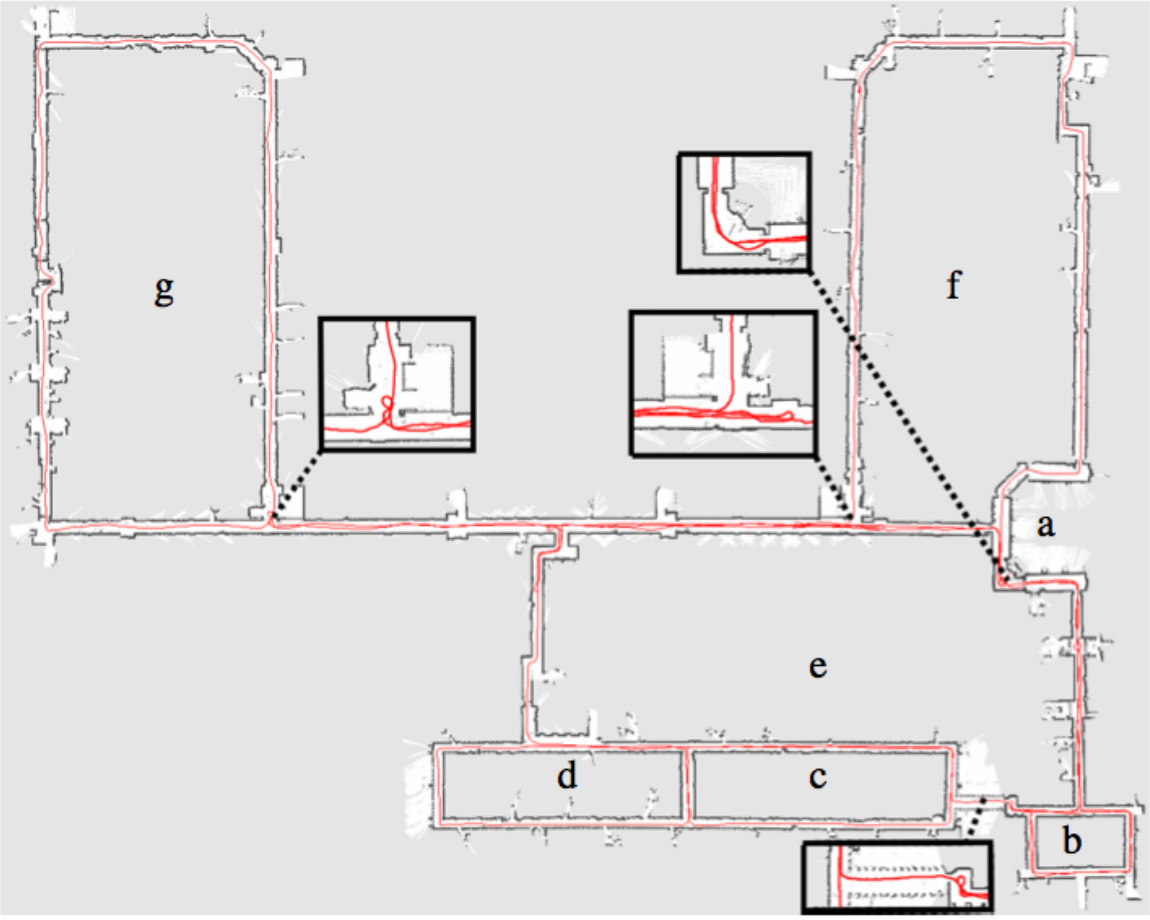


Figure 11.12: Exemple de carte construite par la méthode du FastSLAM. Cette carte contient de nombreux couloirs et plusieurs boucles et représente donc un problème difficile pour le SLAM (image tirée de [60]).

Une méthode de cartographie très similaire au FastSLAM permet de construire une carte métrique directement à partir des données laser [70]. Elle n'utilise pas de filtre de Kalman, mais repose sur un échantillonage des trajectoires possibles par un filtre particulaire et sur un algorithme rapide de corrélation de scans pour l'évaluation des probabilités des trajectoires. Cette méthode permet de construire de très grandes carte contenant plusieurs cycles (Figure 11.12).

11.4 Comparaison des méthodes de cartographie

Les méthodes de cartographie les plus simples sont les méthodes de cartographie incrémentales topologiques, elles montrent cependant rapidement leurs limites en terme de robustesse et demandent souvent une mise au point importante pour obtenir une capacité de reconnaissance de nœuds suffisamment fiables.

Concernant la cartographie métrique, les méthodes de corrélation de scans laser et de grille d'occupation sont également assez simples à mettre en oeuvre et sont suffisantes pour des environnements de taille restreinte, ne contenant notamment pas trop de cycles. Elles sont donc très populaires.

Les méthodes de SLAM seront en général plus efficaces, mais plus complexes à mettre en oeuvre. La version basique du SLAM utilisant un filtre de Kalman reste cependant assez simple, mais au prix d'un manque de robustesse en cas de mauvaises perceptions ou de mauvaise odométrie et d'une grande sensibilité à la qualité de l'association de données. Le FastSLAM quant à lui est plus complexe mais apporte de réels gains de robustesse, il est cependant limité par le problème de raréfaction des particules qui devient problématique pour les environnements de grande taille contenant plusieurs cycles.

11.5 Pour aller plus loin

Les archives de la "SLAM summer school" de 2002 contenant de nombreux articles et tutoriels :

http://www.cas.kth.se/SLAM/

Un site regroupant de nombreux algorithmes de SLAM en open-source :

http://www.openslam.org/

Le livre "Probabilitic Robotics" de Sebastian Thrun, Wolfram Burgard et Dieter Fox [135] détaille de manière très précise l'ensemble des algorithmes de SLAM.

Chapter 12

Planification

Connaissant une carte de l'environnement et la position du robot au sein de cette carte, il est possible de calculer une trajectoire pour rejoindre un but. Nous décrirons dans ce chapitre quelques méthodes simples pour la planification restreinte à des déplacements en 2D. Pour un aperçu plus large des techniques de planification, on pourra par exemple se reporter à [88] ou [90].

12.1 Espace des configurations

Dans le cadre de ce cours, nous considérons des robots capables de se déplacer dans un espace à 2 dimensions dont les commandes influent avec des relations simples sur la position dans cet espace (via la vitesse et la direction). Le calcul des déplacements peut donc se faire directement dans l'espace de la carte, en évitant seulement les positions qui conduiraient à des collisions avec les obstacles. Dans un cadre plus général, la planification est plutôt réalisée dans l'espace des degrés de liberté du robot appelé *espace des configurations* (qui peut être beaucoup plus grand que l'espace des mouvements, par exemple pour un bras manipulateur). Les obstacles de l'espace des déplacements sont alors traduits dans l'espace des configurations par des C obstacles, qui correspondent aux configurations des degrés de liberté qui vont faire percuter un obstacle au robot.

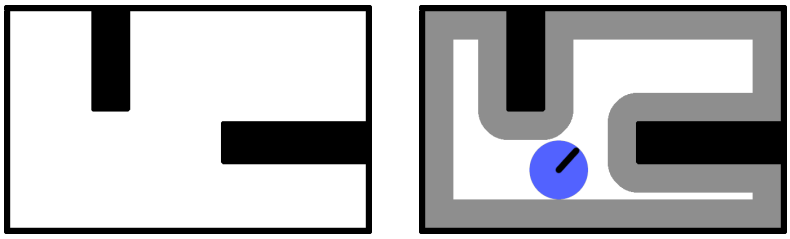


Figure 12.1: Illustration de l'espace des configurations dans le cas 2D. Gauche : Carte des obstacles. Droite : espace des configurations contenant les positions accessibles au robot sans percuter les obstacles (en blanc).

Dans le cas 2D pour un robot holonome , l'espace des configurations que nous utiliserons

sera donc simplement l'espace de travail auquel nous enlèverons toutes les positions conduisant à percuter un obstacle (figure 12.1), c'est à dire les obstacles euxmême, plus une marge de sécurité autour des obstacles correspondant au rayon du robot.

12.2 Discrétisation de l'espace de recherche

Les algorithmes de planification utilisent en général des méthodes de recherche de chemin dans des graphes. Il faut donc représenter la carte sous la forme d'un graphe. Les cartes topologiques fournissent directement ce graphe, mais dans le cas des cartes métriques, qui représentent l'espace de manière continue, ces techniques ne sont utilisables qu'après discrétisation de l'espace libre représenté dans la carte. Pour ce faire, certains modèles intègrent directement cette décomposition au niveau de la cartographie, en construisant une carte topologique parallèlement à la carte métrique [5, 29, 131, 22]. D'autres modèles font appel à des décompositions de l'espace libre spécifiques à la planification. Notons qu'il existe également des techniques, tels les champs de potentiel analogues à ceux décrits dans le chapitre sur la navigation réactive [87, 38, 104] qui permettent de calculer des chemins directement dans le domaine continu, sans phase préalable de discrétisation. Nous n'aborderons cependant pas ces techniques dans ce cours.

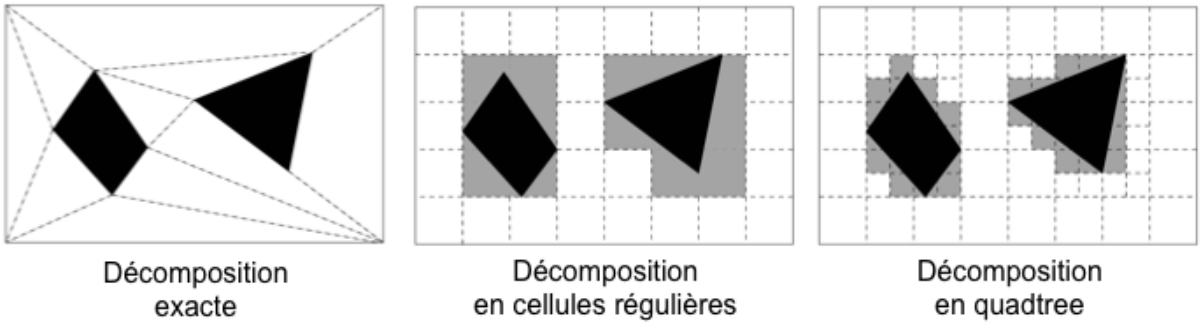


Figure 12.2: Exemples de décompositions en cellules de l'espace libre dans les cartes métriques.

Il existe deux catégories de méthodes pour discrétiser l'espace de recherche des cartes métriques. Les méthodes de la première catégorie font appel à des décompositions en cellules, de différents types, qui permettent de représenter la topologie de l'espace libre [87, 104] (cf. figure 12.2). La décomposition exacte permet de représenter l'ensemble de l'espace libre, à l'aide de cellules de formes irrégulières qui joignent les sommets des obstacles. La décomposition en cellules régulières pave l'espace libre de carrés, sur-estimant donc la surface des obstacles, ce qui peut être gênant si les cellules sont grandes. Ce type de représentation peut aussi correspondre en pratique à une grille d'occupation pour laquelle ce problème ne se pose pas. Enfin une représentation hiérarchique telle que le «quadtree» permet d'utiliser des cellules de taille variable en fonction de la complexité locale de l'environnement et de représenter donc finement l'espace libre tout en limitant l'occupation mémoire. Les cellules obtenues sont ensuite utilisées

de manière similaire aux nœuds des cartes topologiques dans le processus de planification, les cellules adjacentes étant considérées comme reliées par une arête.

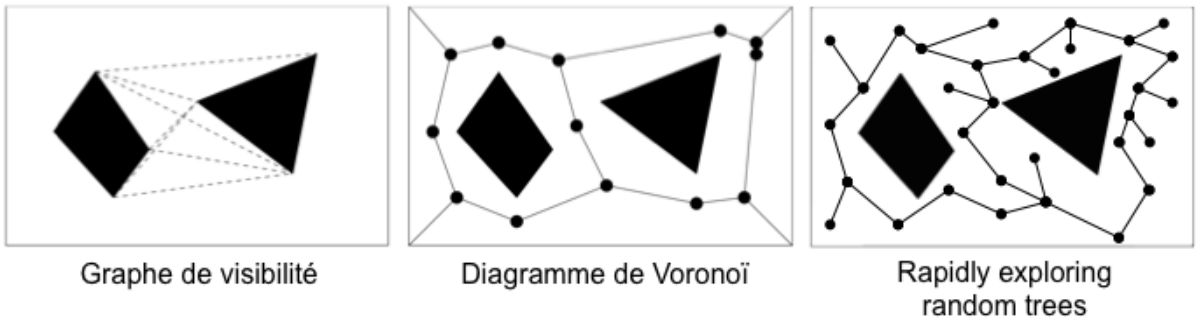


Figure 12.3: Exemples de décompositions en chemins pré-calculés dans les cartes métriques.

Les méthodes de la seconde catégorie font appel au pré-calcul de chemins entre des points répartis dans l'environnement [87, 88] (cf. figure 12.3). Le graphe de visibilité utilise les angles d'obstacles qui sont les points que le robot devra contourner pour éviter ces obstacles. Le diagramme de Voronoï utilise les points équidistants de plusieurs obstacles qui permettent de générer des chemins passant le plus loin possible des obstacles. La méthode "Rapidly exploring Random Trees" [89] quant à elle, construit un arbre aléatoirement en vérifiant que les branches créées ne rencontrent pas les obstacles. Cette méthode est très efficace car elle permet d'échantillonner l'espace sans le parcourir de manière exhaustive et peut aussi prendre en compte les contraintes de non-holonomie du robot. Les point sont ensuite utilisés comme les nœuds d'une carte topologique, tandis que les chemins pré-calculés reliant les nœuds seront utilisés comme les arêtes de cette carte.

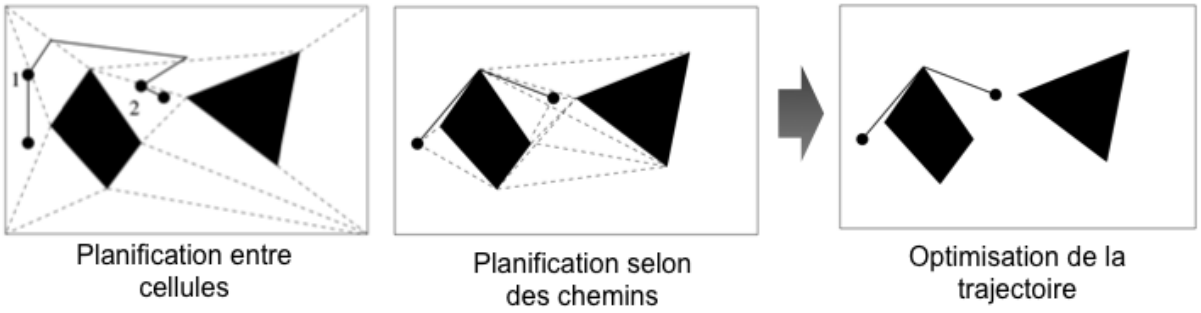


Figure 12.4: Exemple de planification de chemin dans une carte métrique. Deux portions de trajectoire sont calculées pour relier le point de départ et le but à des points de l'espace discrétisé (points 1 et 2 dans cet exemple). Un chemin est ensuite calculé dans l'espace discrétisé entre ces deux points. La trajectoire résultante peut ensuite être optimisée pour supprimer les effets de la discrétisation.

La planification du chemin entre deux points de l'environnement se réalise alors en deux étapes. La première étape permet de calculer un chemin direct entre, d'une part, le point de

départ et le point le plus proche dans l'espace discrétisé et, d'autre part, le point de l'espace discrétisé le plus proche du but et le but en question. La seconde étape permet ensuite de calculer un chemin entre ces deux points de l'espace discrétisé, en utilisant une des méthodes décrites dans le prochain paragraphe. Ces trois parties de trajectoires sont ensuite assemblées pour obtenir le chemin reliant le point de départ au but. Une phase d'optimisation supplémentaire peut être utilisée pour limiter les effets de la discrétisation et lisser la trajectoire (cf. figure 12.4). Dans le cas de la décomposition de l'espace libre en cellules, les points de l'espace discrétisé utilisés peuvent être les centres des cellules ou les milieux des côtés des cellules. Dans le cas de l'utilisation de chemins pré-calculés, ces points sont simplement les points de passage de ces chemins.

12.3 Recherche de chemin

A partir d'une carte métrique discrétisée ou d'une carte topologique représentée sous forme d'un graphe, il existe différentes méthodes pour calculer un chemin entre la cellule de départ et la cellule but. Nous distinguons ici les méthodes selon le type de plans qu'elles génèrent.

12.3.1 Deux types de plan

Le premier type de plan qui peut être généré contient une suite d'actions à effectuer par le robot, ou une suite de points à atteindre afin de rejoindre le but. Les algorithmes classiques de recherche dans les graphes, tels que l'algorithme de Dijkstra, $A^\star$, ou l'une de leurs nombreuses variantes, peuvent être utilisés pour calculer ce type de plan [93, 79, 81, 120, 106, 5, 38]. La taille raisonnable des cartes topologiques classiquement utilisées en robotique rend ces algorithmes suffisamment efficaces en pratique. Ce type de plan pose toutefois des problèmes lors de son exécution si le robot ne parvient pas à atteindre l'un des points du chemin calculé, ou s'il s'éloigne de la trajectoire et que sa position correspond à un nœud qui ne fait pas partie du chemin planifié. La solution à ces problèmes est alors de recommencer le processus de planification en prenant en compte la nouvelle position de départ (ce qui peut même se produire indéfiniment en cas de problème). Ce processus de replanification est souvent inutilement coûteux en calcul car un grand nombre des opérations nécessaires auront déjà été effectuées lors de la planification précédente.

Un second type de plan peut être utilisé, qui associe à chacune des positions possibles du robot au sein de la carte l'action qu'il doit effectuer pour atteindre son but. Ce type de plan est appelé *politique* (comme pour l'apprentissage par renforcement, section 7) ou *plan universel* [121]. Le résultat est alors une stratégie de déplacement similaire à la stratégie *d'action associée à un lieu* mentionnée dans la section 2.1. L'enchaînement de reconnaissances de positions et de réalisations des actions associées à ces positions permet donc de générer une route joignant le but. Ce type de plan présente l'avantage de permettre au robot d'atteindre le but, aussi longtemps qu'il possède une estimation correcte de sa position. En effet, le chemin précis rejoignant le but n'est pas spécifié et le robot peut donc s'écarter du chemin direct entre la position initiale et le but sans entraîner de replanification.

Une politique est plus lourde à calculer que les plans du type précédent car toutes les positions de la carte doivent être envisagées, sans utiliser les heuristiques des algorithmes précédents qui permettent de restreindre l'exploration de l'espace de recherche. Toutefois, cette augmentation est rapidement compensée si le robot s'écarte du chemin direct vers le but. Dans ce cas, en effet, la planification doit être reprise pour le premier type de plan, alors que c'est inutile pour une politique. Le calcul d'une politique reste donc en général praticable pour les cartes de taille limitée typiques de la robotique mobile.

12.3.2 Calcul de politique

Pour calculer une telle politique, une simple recherche en largeur dans le graphe en partant du but peut être utilisée. Cette méthode se retrouve sous le nom de *breadth first search*, *spreading activation* [98, 10] ou *wavefront propagation* [104]. Ces deux derniers noms viennent de l'analogie entre l'ordre de parcours du graphe et la manière dont un fluide progresserait s'il s'échappait du but pour se répandre dans le graphe.

Plutôt que d'associer directement une action à chaque état, le calcul d'une politique passe souvent par le calcul d'un potentiel associé à chaque état qui augmente en fonction de la distance nécessaire pour atteindre le but depuis l'état courant. A partir de ce potentiel, il est alors très simple de retrouver les actions à effectuer par une simple descente de gradient.

Pour calculer ce potentiel, des coûts élémentaires sont associés à chaque nœuds et à chaque lien entre les nœuds. Les coûts associés aux liens traduisent en général la distance entre nœuds, tandis que les coûts associés aux nœuds permettent de marquer des zones à éviter ou à favoriser pour les trajectoires du robot.

Algorithm 7 Algorithme de Dijkstra

```
1: S = ensemble vide; R = ensemble de tous les nœuds;
2: for all Noeuds i do
3:    d(i) = +∞;
4: end for
5: d(but) = 0;
6: while R n'est pas vide do
7:    u = Extract-Min-d(R)
8:    S = S union u
9:    for all Noeud v voisin de u do
10:      if d(v) > d(u) + w(u,v) + w(v) then
11:         d(v) = d(u) + w(u,v) + w(v)
12:      end if
13:   end for
14: end while
```

La méthode la plus simple pour calculer le potentiel est l'algorithme de Dijkstra (7), qui, partant du but fait à chaque étape la mise à jour du nœuds suivant de potentiel le plus faible. Cet algorithme suppose que tous les coûts utilisés sont positifs ou nuls.

Algorithm 8 Algorithme de Bellman-Ford

```
{%} Retourne FAUX si il y a un cycle de coût total négatif
for all Noeuds i do
    d(i) = +∞;
end for
d(but) = 0;
for i=1 jusqu'à Nombre de sommets do
    for all liens (u,v) du graphe do
        if d(v) > d(u) + w(u,v) + w(v) then
            d(v) = d(u) + w(u,v) + w(v)
        end if
    end for
end for
for all liens (u,v) du graphe do
    if d(v) > d(u) + w(u,v) + w(v) then
        return FAUX
    end if
end for
return VRAI
```

Une seconde méthode pour calculer ces potentiels lorsque certains coûts sont négatifs est l'utilisation de l'algorithme de programmation dynamique de Bellman-Ford (8), aussi connu sous le nom de *value itération*, notamment en apprentissage par renforcement [22, 131, 23]. Une itération de cet algorithme utilise quasiment la même méthode de mise a jour que l'algorithme de Dijkstra mais sans utiliser le même ordre dans les mises à jour des nœuds. En présence de coûts négatifs, cette itération doit être répétée autant de fois qu'il y a de nœuds pour garantir la convergence. De plus il peut arriver qu'un cycle du graphe ait un poids total négatif, ce qui ne permet pas de trouver de chemin de coût minimal. L'algorithme retourne FAUX dans ce cas.

La figure 12.5 permet de comparer l'ordre des mises à jour des valeurs de potentiels pour l'algorithme de Dijkstra et pour Value Iteration, qui aboutissent au même résultat.

12.3.3 Calcul d'un chemin

Si l'on ne souhaite que calculer un chemin, il est par exemple possible d'employer l'algorithme $A^\star$. Cet algorithme utilise exactement le même mécanisme que l'algorithme de Dijkstra, mais utilise une heuristique pour choisir le nœud suivant à explorer au lieu d'explorer systématiquement les nœuds voisins des nœuds déjà planifiés. Cette heuristique doit fournir une évaluation la plus rapide possible de la distance entre le nœud courant et le but (par exemple simplement la distance euclidienne en supposant qu'il n'y a pas d'obstacles). La mise en place d'une bonne heuristique assure de trouver très rapidement un chemin vers le but, mais ne garantit pas forcement l'optimalité (ce qui n'est souvent pas très important en pratique).

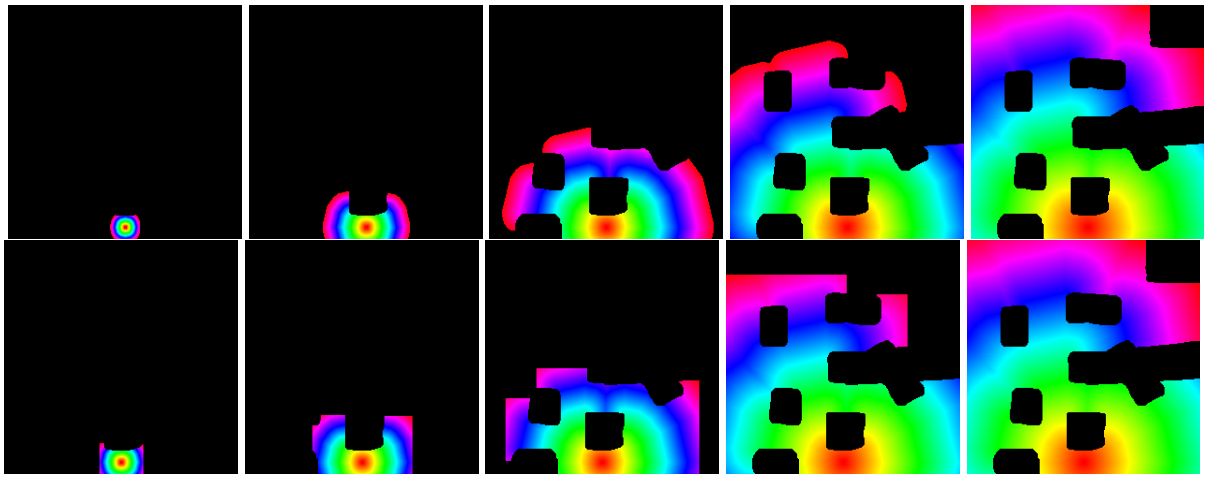


Figure 12.5: Propagation des valeurs de potentiel selon l'algorithme utilisé. Les poids associés aux noeuds (ici les cases d'une grille d'occupation) sont nuls, et les poids des liens sont la distance entre deux noeuds. La rangée du haut montre les résultats de l'algorithme de Dijkstra, celle du bas ceux de Value Iteration.

12.4 Exemples de politiques

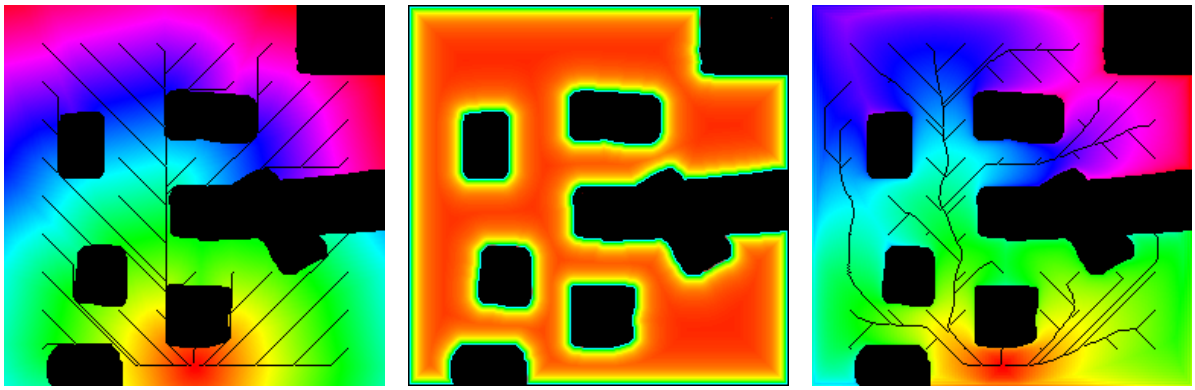


Figure 12.6: Gauche : Potentiel et exemple de trajectoires avec des poids de liens égaux à la distance entre noeuds. Centre : Poids associé aux noeuds fonction de la distance aux obstacles. Droite : Exemples de trajectoires obtenues en utilisant les poids de l'image du centre.

La figure 12.6 donne à gauche un exemple de champ de potentiel obtenu en utilisant un poids nul pour les nœuds et la distance entre nœuds comme poids sur les liens. Avec ce type de carte, ce choix pose le problème de générer des trajectoires très proche des obstacles, qui peuvent être dangereuses pour le robot.

Pour éviter ce problème, il est possible d'associer un poids dépendant de la distance à l'obstacle le plus proche à chacun des noeuds. Ceci a pour but de pénaliser les trajectoires proches des murs et de guider le robot selon l'axe des couloirs au lieu de longer un des murs. La

figure 12.6 montre à droite le poids associé aux nœuds et les trajectoires obtenues.

12.5 Choix de l'action avec une position incertaine

Lorsque la position estimée par le système de localisation est non ambiguë, l'utilisation d'une politique se résume simplement au choix de l'action associée avec la position courante. Toutefois, les systèmes réalisant le suivi de plusieurs hypothèses fournissent également une estimation de la probabilité de présence du robot en différentes positions. Il peut donc se révéler utile de tenir compte de ces probabilités pour sélectionner l'action à exécuter.

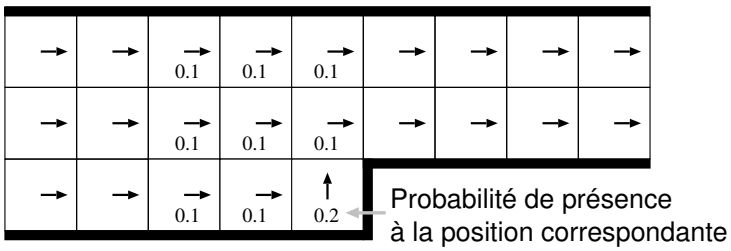


Figure 12.7: Exemple de l'intérêt d'une procédure de vote dans le cas où la situation du robot est incertaine. Si l'on choisit l'action associée à la position la plus probable, le robot ira en haut, ce qui correspondra à l'action correcte avec une probabilité $0,2$*. En utilisant une méthode de vote, l'action choisie sera d'aller à droite, ce qui sera correct avec une probabilité* $0,8$*.*

Différentes méthodes permettent de prendre ces probabilités en compte. Une première méthode consiste à utiliser une méthode de vote [127, 27]. Pour cela, une action est simplement associée à chacun des nœuds de la carte, en utilisant une des méthodes décrites au paragraphe précédent. Un score est alors calculé pour chaque action. Ce score est la somme des probabilités des nœuds auxquels chaque action est associée. L'action ayant le score le plus élevé est alors exécutée. Cette méthode est efficace dans les cas de grande ambiguïté dans la localisation, où la probabilité de la position la plus probable est seulement très légèrement supérieure aux autres. Dans ce cas, en effet, si la direction associée à la position la plus probable est incorrecte, cette méthode permet de l'ignorer et de choisir une direction associée à plusieurs autres hypothèses de position qui se révélera correcte dans un plus grand nombre de cas (cf. figure 12.7).

La méthode précédente tente de diriger le robot vers le but quelle que soit l'incertitude de l'estimation de sa position. Or si cette estimation est très incertaine, il est souvent plus judicieux de chercher d'abord à mieux l'estimer avant de chercher à rejoindre le but. Il est ainsi possible de mesurer la confiance dans l'estimation courante de la position pour choisir une action. Cette confiance peut être simplement mesurée par l'entropie de la distribution de probabilité [27, 132]. Ainsi, si l'entropie de la distribution de probabilité représentant la position est trop élevée, une action permettant de diminuer cette entropie sera sélectionnée. L'utilisation de telles stratégies permet par exemple d'éviter des zones dans lesquelles l'incertitude de localisation est plus grande (par exemple les larges espaces ouverts), et de privilégier les zones plus favorables à l'estimation

de la position (par exemples les zones où se trouvent des points de repère fiables). Cette méthode a été présentée sous le nom de "Coastal Navigation" [115].

12.6 Pour aller plus loin

Robot Motion Planning and Control de J.-P. Laumond. Lectures Notes in Control and Information Sciences 229. Springer, 1998 [88]. Disponible en ligne :

http://homepages.laas.fr/jpl/book.html

Planning algorithms de Steven M. LaValle, Cambridge University Press, 2006 [90]. Disponible en ligne :

http://planning.cs.uiuc.edu/

Index

Bibliography

[1] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer. A fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions On Robotics, Special Issue on Visual SLAM*, 2008. *Cité page 134*

[2] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer. Visual topological slam and global localization. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2009. *2 citations pages 143 et 144*

[3] R. Arkin. Towards the unification of navigational planning and reactive control. In *Proceedings of the AAAI Spring Symposium on Robot Navigation*, pages 1–5, 1989. *Cité page 23*

[4] Ronald Arkin. *Behavior-Based Robotics*. The MIP Press, 1998. *2 citations pages 21 et 60*

[5] A. Arleo, J. del R. Millán, and D. Floreano. Efficient learning of variable-resolution cognitive maps for autonomous indoor navigation. In *IEEE Transactions on Robotics and Automation*, volume 15, pages 990–1000, 1999. *3 citations pages 97, 154, et 156*

[6] A. Arleo and W. Gerstner. Spatial cognition and neuro-mimetic navigation: A model of hippocampal place cell activity. *Biological Cybernetics, Special Issue on Navigation in Biological and Artificial Systems*, 83:287–299, 2000. *5 citations pages 92, 93, 104, 109, et 137*

[7] A. Arsenio and M. I. Ribeiro. Absolute localization of mobile robots using natural landmarks. In *Proceedings of the International Conference on Electronics, Circuits and Systems*, 1998. *Cité page 105*

[8] N. Ayache and O. Faugeras. Maintaning representations of the environment of a mobile robot. *IEEE Transactions on Robotics and Automation*, 5(6):804 – 819, 1989. *3 citations pages 97, 105, et 110*

[9] I. A. Bachelder and A. M. Waxman. Mobile robot visual mapping and localization: A view-based neurocomputational architecture that emulates hippocampal place learning. *Neural Networks*, 7(6/7):1083–1099, 1994. *2 citations pages 92 et 103*

[10] I. A. Bachelder and A. M. Waxman. A view-based neurocomputational system for relational map-making and navigation in visual environments. *Robotics and Autonomous Systems*, 16:267–298, 1995. *2 citations pages 136 et 157*

[11] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21. Lawrence Erlbaum Associates (Hillsdale), 1987. *Cité page 129*

[12] K. Balakrishnan, O. Bousquet, and V. Honavar. Spatial learning and localization in rodents : A computation model of the hippocampus and its implications for mobile robots. *Adaptive Behavior*, 7(2):173–216, 1999. *5 citations pages 92, 93, 104, 109, et 137*

[13] S. Bazeille and D. Filliat. Incremental topo-metric slam using vision and robot odometry. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2011. *2 citations pages 93 et 98*

[14] M. Betke and K. Gurvits. Mobile robot localization using landmarks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-94)*, volume 2, pages 135–142, 1994. *2 citations pages 104 et 110*

[15] G. Blanc, Y. Mezouar, and P. Martinet. Indoor navigation of a wheeled mobile robot along visual routes. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005. *Cité page 58*

[16] D. Boley, E. Steinmetz, and K. Sutherland. Robot localization from landmarks using recursive total least squares. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-96)*, volume 4, pages 1381–1386, 1996. *Cité page 110*

[17] J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7:278–288, 1991. *Cité page 61*

[18] G. Borghi and D. Brugali. Autonomous map learning for a multi-sensor mobile robot using diktiometric representation and negotiation mechanism. In *Proceedings of the International Conference on Advanced Robotics (ICAR-95)*, 1995. *3 citations pages 95, 105, et 110*

[19] Valentino Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. The MIT Press, 1986. *2 citations pages 17 et 55*

[20] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 1(47):139–159, 1991. *Cité page 22*

[21] Rodney A. Brooks. How to Build Complete Creatures Rather than Isolated Cognitive Simulators. In *Architectures for Intelligence*, pages 225–239, 1991. *Cité page 22*

[22] J. Buhmann, W. Burgard, A. B. Cremers, D. Fox, T. Hofmann, F. Schneider, J. Strikos, and S. Thrun. The mobile robot rhino. *AI Magazine*, 16(1), 1995. *2 citations pages 154 et 158*

[23] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*. The MIT Press, 1998. *Cité page 158*

[24] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 896–901, 1996. *2 citations pages 123 et 125*

[25] N. Burgess, M. Recce, and J. O'Keefe. A model of hippocampal function. *Neural Networks*, 7:1065–1081, 1994. *4 citations pages 92, 93, 103, et 136*

[26] B. A. Cartwright and T. S. Collett. Landmark maps for honeybees. *Biol. Cybern.*, 57:85–93, 1987. *Cité page 17*

[27] A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996. *4 citations pages 104, 123, 125, et 160*

[28] J. A. Castellanos, J. M. M. Montiel, J. Neira, , and J. D. Tardos. The SPmap : A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15(5):948–953, 1999. *3 citations pages 97, 105, et 110*

[29] R. Chatila and J. Laumond. Position referencing and consistent world modelling for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-85)*, pages 138–170, 1985. *2 citations pages 87 et 154*

[30] F. Chaumette. *La commande des robots manipulateurs*, chapter Asservissement visuel. Traité IC2, Hermès, 2002. *Cité page 58*

[31] I. J. Cox. Blanche - an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, 7(2):193–204, 1991. *2 citations pages 105 et 110*

[32] A. Dalgalarrondo, D. Dufourd, and D. Filliat. Controlling the autonomy of a reconnaissance robot. In *SPIE Defense and Security 2004 Symposium. Unmanned Ground Vehicle Technology VI Conference*, 2004. *Cité page 138*

[33] G. Dedeoglu, M. Mataric, and G. S. Sukhatme. Incremental, online topological map building with a mobile robot. In *Proceedings of Mobile Robots XIV - SPIE*, pages 129–139, 1999. *5 citations pages 91, 93, 104, 109, et 137*

[34] A. Diosi, S. Segvic, A. Remazeilles, and F. Chaumette. Experimental evaluation of autonomous driving based on visual memory and image based visual servoing. *IEEE Trans. on Intelligent Transportation Systems*, 2011. *Cité page 58*

[35] T. Duckett, S. Marsland, and J. Shapiro. Learning globally consistent maps by relaxation. In *Proceedings of the International Conference on Robotics and Automation (ICRA'2000)*, pages 3841 – 3846, 2000. *Cité page 143*

[36] T. Duckett and U. Nehmzow. Experiments in evidence based localisation for a mobile robot. In D. Corne and J. L. Shapiro, editors, *Proceedings of the AISB 97 workshop on Spatial Reasoning in Animals and Robots*. Springer, 1997. *2 citations pages 92 et 93*

[37] T. Duckett and U. Nehmzow. Mobile robot self-localization and measurement of performance in middle scale environments. *Robotics and Autonomous Systems*, 1-2(24), 1998. *Cité page 120*

[38] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, 2000. *2 citations pages 154 et 156*

[39] G. Dudek and P. MacKenzie. Model-based map construction for robot localization. In *Proceedings of Vision Interface 1993*, 1993. *2 citations pages 97 et 110*

[40] S. Egerton and V. Callaghan. From mammals to machines: Towards a biologically inspired mapping model for autonomous mobile robots. In *Procceding of the 6th International Conference on Intelligent Autonomous Systems (IAS-6)*, 2000. *Cité page 104*

[41] T. Einsele. Real-time self-localization in unknown indoor environments using a panorama laser range finder. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-97)*, pages 697–703, 1997. *2 citations pages 97 et 105*

[42] S. P. Engelson. Continuous map learning for mobile robots. Extended Abstract for the 3rd French-Israeli Symposium on Robotics, 1995. *Cité page 141*

[43] S. P. Engelson and D. V. McDermott. Error correction in mobile robot map learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-92)*, 1992. *5 citations pages 91, 92, 93, 97, et 137*

[44] H. Feder, J. Leonard, and C. Smith. Adaptive mobile robot navigation and mapping. *International Journal of Robotics Research*, 18(7):650–668, 1999. *2 citations pages 95 et 97*

[45] D. Filliat and J.-A. Meyer. Map-based navigation in mobile robots - i. a review of localisation strategies. *Journal of Cognitive Systems Research, submitted for publication*, 2001. *Cité page 101*

[46] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization : Efficient position estimation for mobile robots. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*. AAAI, 1999. *Cité page 125*

[47] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1), 1997. *Cité page 61*

[48] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journalof Artificial Intelligence Research*, 11, 1999. *2 citations pages 125 et 126*

[49] D. Fox, W. Burgard, S. Thrun, and A. B. Cremers. Position estimation for mobile robots in dynamic environments. In *Proceedings of the Fifteenth National Conference on Articial Intelligence (AAAI-98)*, pages 983–988, 1998. *2 citations pages 123 et 125*

[50] Dieter Fox. Kld-sampling: Adaptive particle filters and mobile robot localization. In *In Advances in Neural Information Processing Systems (NIPS*, 2001. *Cité page 130*

[51] M. Franz, B. Scholkopf, P. Georg, H. Mallot, and H. Bulthoff. Learning view graphs for robot navigation. *Autonomous Robots*, 5:111–125, 1998. *3 citations pages 92, 103, et 136*

[52] A. Garulli, A. Giannitrapani, A. Rossi, and A. Vicino. Mobile robot slam for line-based environment representation. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 2041–2046. IEEE, 2005. *2 citations pages 96 et 97*

[53] J. Gasós and A. Martín. Mobile robot localization using fuzzy maps. In T. Martin and A. Ralescu, editors, *Fuzzy Logic in AI - Selected papers from the IJCAI'95 Workshop*, number 1188, pages 207–224. Springer-Verlag, 1997. *Cité page 97*

[54] P. Gaussier, C. Joulain, J.P. Banquet, S. Lepretre, and A. Revel. The visual homing problem : an example of robotics/biology cross-fertilisation. *Robotics and autonomous systems*, 30(1-2):155–180, 2000. *3 citations pages 17, 92, et 103*

[55] P. Gaussier, S. Leprêtre, C. Joulain, A. Revel, M. Quoy, and Banquet J. P. Animal and robot learning: experiments and models about visual navigation. In *Proceedings of the Seventh European Workshop on Learning Robots*, 1998. *2 citations pages 92 et 136*

[56] A. P. Georgopoulos, A. B. Schwartz, and R. E. Kettner. Neuronal population coding of movement direction. *Science*, (233):1416–1419, 1986. *Cité page 104*

[57] J. Gomes-Mota and M. I. Ribeiro. Mobile robot localisation on reconstructed 3d models. *Robotics and Autonomous Systems*, 31(1-2):17–30, 2000. *2 citations pages 105 et 110*

[58] S. Gourichon and J.-A. Meyer. Using colored snapshots for short-range guidance in mobile robots. *International Journal of Robotics and Automation, submitted for publication*, Special Issue on Biologically Inspired Robots, 2001. *2 citations pages 17 et 57*

[59] R. Greiner and R. Isukapalli. Learning to select useful landmarks. *IEEE Transactions on Systems, Man, and Cybernetics-Part B,Special Issue on Learning Autonomous Robots*, 26(3), 1996. *Cité page 104*

[60] G. Grisetti, C. Stachniss, and W. Burgard. Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2432–2437, 2005. *2 citations pages 150 et 151*

[61] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Nonlinear constraint network optimization for efficient map learning. *Trans. Intell. Transport. Sys.*, 10:428–439, September 2009. *Cité page 145*

[62] L. J. Guibas, R. Motwani, and P. Raghavan. The robot localization problem. *Algorithmic Foundations of Robotics*, pages 269–282, 1995. *Cité page 106*

[63] J. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proceedingsof the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA-2000)*, 2000. *Cité page 95*

[64] J. Gutmann and Kurt Konolige. Incremental mapping of large cyclic environments. In *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, page 318–325, Monterey, California, 1999. *Cité page 138*

[65] J. S. Gutmann and C. Schlegel. Amos: Comparison of scan matching approaches for self-localization in indoor environments,. In *Proceedings of the 1st Euromicro Workshop on Advanced Mobile Robots*. IEEE Computer Society Press, 1996. *Cité page 105*

[66] V. V. Hafner. Learning places in newly explored environments. In J. A. Meyer, A. Berthoz, D. Floreano, H. L. Roiblat, and S. W. Wilson, editors, *Sixth International Conference on simulation of adaptive behavior : From Animals to Animats (SAB-2000). Proceedings Supplement.*, pages 111–120. ISAB, 2000. *4 citations pages 92, 104, 123, et 125*

[67] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. *Cité page 47*

[68] P. Hébert, S. Betgé-Brezetz, and R. Chatila. Decoupling odometry and exteroceptive perception in building a global world map of a mobile robot: The use of local maps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-1996)*, pages 757–764, 1996. *2 citations pages 95 et 97*

[69] J. Hertzberg and F. Kirchner. Landmark-based autonomous navigation in sewerage pipes. In *Proceedings of the First Euromicro Workshop on Advanced Mobile Robots*. IEEE Computer Society Press, 1996. *5 citations pages 91, 92, 104, 123, et 125*

[70] D. Hähnel, D. Fox, W. Burgard, and S. Thrun. A highly efficient fastslam algorithm for generating cyclic maps of large-scale environments from raw laser range measurements. In *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, 2003. *2 citations pages 150 et 151*

[71] Daniel Ichbiah. *Robots, Génèse d'un peuple artificiel*. Minerva, 2005. *Cité page 12*

[72] I. Jebari, S. Bazeille, E. Battesti, H. Tekaya, M. Klein, A. Tapus, D. Filliat, C. Meyer, S. Ieng, R. Benosman, E. Cizeron, J.-C. Mamanna, and B. Pothier. Multi-sensor semantic mapping and exploration of indoor environments. In *Proceedings of the 3rd International Conference on Technologies for Practical Robot Applications (TePRA)*, 2011. *Cité page 98*

[73] P. Jensfelt and S. Kristensen. Active global localisation for a mobile robot using multiple hypothesis tracking. In *Proceedings of the IJCAI-99 Workshop on Reasoning with Uncertainty in Robot Navigation*, 1999. *2 citations pages 105 et 120*

[74] S. Julier and J. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL*, 1997. *Cité page 116*

[75] O. Karch and T. Wahl. Relocalization – theory and practice. *Discrete Applied Mathematics : Special Issue on Computational Geometry*, 93, 1999. *Cité page 106*

[76] Hee-Young Kim, Sung-On Lee, and Bum-Jae You. Robust laser scan matching in dynamic environments. In *Proceedings of the 2009 international conference on Robotics and biomimetics*, ROBIO'09, pages 2284–2289. IEEE Press, 2009. *Cité page 106*

[77] D. Kirsh. Today the earwig, tomorrow man? *Artificial Intelligence*, 47:161–184, 1991. *Cité page 22*

[78] Y. Koren and J. Borenstein. Histogramic in-motion mapping for mobile robot obstacle avoidance. *IEEE Transaction on Robotics and Automation*, 7(4):535–539, 1991. *Cité page 141*

[79] D. Kortenkamp, M. Huber, F. Koss, W. Belding, J. Lee, A. Wu, C. Bidlack, and S. Rogers. Mobile robot exploration and navigation of indoor spaces using sonar and vision. In *Proceedings of the AIAA/NASA Conference on Intelligent Robots in Field, Factory, Service, and Space (CIRFFSS 94)*, pages 509–519, 1994. *3 citations pages 92, 125, et 156*

[80] D. Kortenkamp and T. Weymouth. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 979–984, Seattle, WA, 1994. *2 citations pages 91 et 103*

[81] B. J. Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, (119):191–233, 2000. *Cité page 156*

[82] B. J. Kuipers and Y. T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8:47–63, 1991. *7 citations pages 87, 91, 92, 104, 109, 137, et 141*

[83] C. Kunz, T. Willeke, and I. Nourbakhsh. Automatic mapping of dynamic office environments. In *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-97)*, volume 2, pages 1681–1687, 1997. *7 citations pages 91, 92, 104, 109, 137, 139, et 142*

[84] A. Kurz. Alef : An autonomous vehicle which learns basic skills and construct maps for navigation. *Robotics and Autonomous Systems*, 14:172–183, 1995. *4 citations pages 92, 93, 109, et 137*

[85] C. Kwok, D. Fox, and M. Meila. Adaptive real-time particle filters for robot localization. In *Proc. of the IEEE International Conference on Robotics & Automation*, 2003. *Cité page 130*

[86] D. Lambrinos, R. Möller, T. Labhart, R. Pfeifer, and R. Wehner. A mobile robot employing insect strategies for navigation. *Robotics and Autonomous Systems, special issue: Biomimetic Robots*, 30:39–64, 2000. *Cité page 17*

[87] J.-C. Latombe. *Robot Motion Planning*. Boston: Kluwer Academic Publishers, Boston, 1991. *3 citations pages 95, 154, et 155*

[88] J.-P. Laumond. *Robot Motion Planning and Control*. Lectures Notes in Control and Information Sciences 229. Springer, 1998. *3 citations pages 153, 155, et 161*

[89] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998. *Cité page 155*

[90] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, May 2006. *2 citations pages 153 et 161*

[91] J. J. Leonard and H. F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. pages 1442–1447, 1991. *Cité page 145*

[92] J. J. Leonard, H. F. Durrant-Whyte, and I. J. Cox. Dynamic map building for an autonomous mobile robot. *International Journal of Robotics Research*, 11(4):89–96, 1992. *3 citations pages 97, 105, et 110*

[93] T. S. Levitt and D. T. Lawton. Qualitative navigation for mobile robots. *Artificial Intelligence*, 44:305–360, 1990. *8 citations pages 83, 92, 93, 95, 103, 104, 136, et 156*

[94] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997. *3 citations pages 95, 105, et 110*

[95] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Auton. Robots*, 4:333–349, October 1997. *Cité page 106*

[96] C. Madsen, C. Andersen, and J. rensen. A robustness analysis of triangulation-based robot self-positioning. In *Proceedings of the 5th Symposium for Intelligent Robotics Systems*, 1997. *Cité page 104*

[97] O. Martínez Mozos, R. Triebel, P. Jensfelt, A. Rottmann, and W. Burgard. Supervised semantic labeling of places using information extracted from sensor data. *Robotics and Autonomous Systems*, 55(5):391–402, May 2007. *Cité page 98*

[98] M. J. Mataric. Integration of representation into goal-driven behaviour-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, 1992. *6 citations pages 92, 93, 104, 109, 137, et 157*

[99] P. S. Maybeck. *Stochastic Models, Estimation and Control*. Academic Press, 1979. *Cité page 110*

[100] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI. *Cité page 150*

[101] H. Moravec and A. Elfes. High resolution maps from wide angular sensors. In *Proceedings of the IEEE International Conference On Robotics and Automation (ICRA-85)*, pages 116–121, St. Louis, 1985. IEEE Computer Society Press. *3 citations pages 87, 97, et 139*

[102] Hans Moravec. *ROBOT: mere machine to transcendent mind*. Oxford University Press, 1995. *Cité page 12*

[103] P. Moutarlier and R. Chatila. An experimental system for incremental environment modeling by an autonomous mobile robot. In *Experimental Robotics 1*, pages 327–346. Springer-Verlag, 1990. *3 citations pages 97, 105, et 110*

[104] R. R. Murphy. *Introduction to AI Robotics*. The MIT Press, 2000. *3 citations pages 23, 154, et 157*

[105] U. Nehmzow and C. Owen. Robot navigation in the real world: Experiments with manchester's fortytwo in unmodified, large environments,. *Robotics and Autonomous Systems*, 33(4):223–242, 2000. *4 citations pages 92, 104, 109, et 137*

[106] I. Nourbakhsh, R. Powers, and S. Birchfield. Dervish, an office navigating robot. *AI Magazine*, 16(2):53–60, 1995. *4 citations pages 92, 104, 125, et 156*

[107] C. F. Olson. Probabilistic self-localization for mobile robots. *IEEE Transactions on Robotics and Automation*, 16(1), 2000. *2 citations pages 105 et 106*

[108] S. Oore, G. Hinton, and G. Dudek. A mobile robot that learns its place. *Neural Computation*, 9:683–699, 1997. *3 citations pages 93, 104, et 125*

[109] D. Gálvez-López P. Piniés, L. M. Paz and J.D. Tardós. Ci-graph slam for 3d reconstruction of large and complex environments using a multicamera system. *International Journal of Field Robotics*, September/October 2010. *2 citations pages 96 et 149*

[110] M. Piasecki. Global localization for mobile robots by multiple hypothesis tracking. *Robotics and Autonomous Systems*, 16:93–104, 1995. *2 citations pages 102 et 120*

[111] T. J. Prescott. Spatial representation for navigation in animats. *Adaptive Behavior*, 4(2), 1995. *2 citations pages 95 et 104*

[112] Andrzej Pronobis. *Semantic Mapping with Mobile Robots*. PhD thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, June 2011. *2 citations pages 98 et 99*

[113] D. Radhakrishnan and I. Nourbakhsh. Topological localization by training a vision-based transition detector. In *Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-99)*, 1999. *Cité page 104*

[114] A. Remazeilles and F. Chaumette. Image-based robot navigation from an image memory. *Robotics and Autonomous Systems*, 55(4), 2007. *2 citations pages 17 et 18*

[115] Nicholas Roy and Sebastian Thrun. Coastal navigation with mobile robots. In *In Advances in Neural Processing Systems 12*, pages 1043–1049, 1999. *Cité page 161*

[116] Thomas Röfer. Building consistent laser scan maps. In *In Proc. of the 4th European Workshop on Advanced Mobile Robots (Eurobot 2001), volume 86 of Lund University Cognitive Studies, pages 83 ? 90*, pages 83–90, 2001. *Cité page 106*

[117] A. Saffiotti and L. P. Wesley. Perception-based self-localization using fuzzy locations. In *Reasoning with Uncertainty in Robotics*, volume 1093 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995. *Cité page 105*

[118] S. Schaal, C. G. Atkeson, and S. Vijayakumar. Real-time robot learning with locally weighted statistical learning. In *International conference on robotics and automation (icra2000)*, 2000. *Cité page 76*

[119] B. Schiele and J. Crowley. A comparison of position estimation techniques using occupancy grids. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-94)*, pages 1628–1634, 1994. *2 citations pages 105 et 110*

[120] B. Scholkopf and H. A. Mallot. View-based cognitive mapping and path planning. *Adaptive Behavior*, 3(3):311–348, 1995. *Cité page 156*

[121] M. J. Schoppers. Universal plans for reactive robots in unpredictable environments. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence (IJCAI 87)*, pages 1039–1046, Milan, Italy, 1987. *Cité page 156*

[122] A. C. Schultz and W. Adams. Continuous localization using evidence grids. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-98)*, pages 2833–2839, 1998. *2 citations pages 105 et 110*

[123] S. Segvic, A. Remazeilles, A. Diosi, and F. Chaumette. A mapping and localization framework for scalable appearance-based navigation. *Computer Vision and Image Understanding*, 113(2):172–187, February 2009. *Cité page 58*

[124] P. E. Sharp. Computer simulation of hippocampal place cells. *Psychobiology*, 19(2):103–115, 1991. *4 citations pages 92, 93, 103, et 136*

[125] H. Shatkay and L. P. Kaelbling. Learning topological maps with weak local odometric information. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, 1997. *5 citations pages 91, 92, 104, 123, et 125*

[126] R. Sim and G. Dudek. Learning visual landmarks for pose estimation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-1999)*, 1999. *Cité page 105*

[127] R. Simmons and S. Koenig. Probabilistic navigation in partially observable environments. In S. Mellish, editor, *Proccedings of IJCAI'95*, Montreal,Canada, 1995. Morgan Kaufman Publishing. *5 citations pages 92, 104, 123, 125, et 160*

[128] Danijel Skocaj, Horst Bischof, and Ales Leonardis. A robust pca algorithm for building representations from panoramic images. In *Proceedings of the 7th European Conference on Computer Vision-Part IV*, ECCV '02, pages 761–775, London, UK, UK, 2002. Springer-Verlag. *Cité page 104*

[129] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In J. F. Lemmer and L. N. Kanal, editors, *Uncertainty in Artificial Intelligence*, pages 435–461. Elsevier, 1988. *5 citations pages 95, 97, 110, 145, et 146*

[130] G. Theocharous, K. Rohanimanesh, and S. Mahadevan. Learning hierarchical partially observable markov decision processes for robot navigation. In *Proceedings of the IEEE Conference on Robotics and Automation*, 2001. *2 citations pages 104 et 125*

[131] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1999. *9 citations pages 87, 97, 98, 105, 110, 139, 142, 154, et 158*

[132] S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93–109, 2000. *Cité page 160*

[133] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Minerva: A second generation mobile tour-guide robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-1999)*, 1999. *2 citations pages 124 et 125*

[134] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-2000)*, 2000. *2 citations pages 95 et 134*

[135] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*. Intelligent robotics and autonomous agents. The MIT Press, 2005. *2 citations pages 45 et 152*

[136] Nicola Tomatis, Illah R. Nourbakhsh, and Roland Siegwart. Hybrid simultaneous localization and map building: a natural integration of topological and metric. *Robotics and Autonomous Systems*, 44(1):3–14, 2003. *Cité page 98*

[137] D. S. Touretzky, H. S. Wan, and A. D. Redish. Neural representations of space in rats and robots. In J. M. Zurada, R. J. Marks, and C. J. Robinson, editors, *Computational Intelligence: Imitating Life*, pages 57–68. IEEE Press, 1994. *5 citations pages 92, 93, 104, 109, et 137*

[138] O. Trullier and J. A. Meyer. Biomimetic navigation models and strategies in animats. *AI Communications*, 10:79–92, 1997. *Cité page 17*

[139] O. Trullier and J. A. Meyer. Animat navigation using a cognitive graph. *Biological Cybernetics*, 83(3):271–285, 2000. *Cité page 103*

[140] O. Trullier, S. Wiener, A. Berthoz, and J. A. Meyer. Biologically-based artificial navigation systems: Review and prospects. *Progress in Neurobiology*, 51:483–544, 1997. *Cité page 17*

[141] I. Ulrich and I. Nourbakhsh. Appearance-based place recognition for topological localization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-2000)*, 2000. *3 citations pages 92, 104, et 109*

[142] G. Von Wichert. Mobile robot localization using a self-organised visual environment representation. *Robotics and Autonomous Systems*, 25:185–194, 1998. *5 citations pages 92, 93, 104, 109, et 137*

[143] Chieh-Chih Wang, Charles Thorpe, Sebastian Thrun, Martial Hebert, and Hugh Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research*, 26(9):889–916, September 2007. *Cité page 85*

[144] O. Wijk and H. I. Christensen. Localization and navigation of a mobile robot using natural point landmarks extracted from sonar data. *Robotics and Autonomous Systems*, 31(1-2):31–42, 2000. *2 citations pages 104 et 110*

[145] B. Yamauchi and R. Beer. Spatial learning for navigation in dynamic environments. *IEEE Transactions on Systems, Man, and Cybernetics-Part B,Special Issue on Learning Autonomous Robots*, 26(3):496–505, 1996. *3 citations pages 92, 109, et 137*

[146] B. Yamauchi and P. Langley. Place recognition in dynamic environments. *Journal of Robotic Systems, Special Issue on Mobile Robots*, 14(2):107–120, 1997. *2 citations pages 93 et 136*

[147] B. Yamauchi, A. Schultz, and W. Adams. Integrating exploration and localization for mobile robots. *Adaptive Behavior*, 7(2):217–230, 1999. *3 citations pages 97, 110, et 142*