République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Centre Universitaire de Souk-Ahras

Institut des Sciences et de Technologie

Ecole Doctorale en Informatique de l'Est – Pôle Annaba

Mémoire

Présenté en vue de l'obtention du diplôme de
Magistère en Informatique
Option : Intelligence Artificielle

Titre

Evitement d'Obstacles Dynamiques par un Robot Mobile

Présenté par

CHERIBET Mohamed

Devant le jury suivant

1	Dr. BOUDOUR R	MC	UBM Annaba	Président
2	Dr. KIMOUR M.T	MC	UBM Annaba	Examinateur
3	Dr. GHANEMI S	MC	UBM Annaba	Examinateur
4	Dr. AMIRAT A	MC	CU Souk Ahras	Examinateur
5	Pr. LASKRI M.T	PR	UBM Annaba	Directeur de recherche

Centre Universitaire de Souk-Ahras – 2010


U. Souk-Ahras

ملخص

لتشغيل الروبوت في بيئة معينة ، فإن المشكلة العامة للتخطيط تتمثل في تحديد حركة الروبوت التي تسمح له بالتنقل بين موضعين معينين مع احترام عدد من القيود والشروط. هذه تنجم عن عوامل عدة من مختلف الأنواع وغالبا ما تعتمد على خصائص الروبوت ، البيئة ونوع المهام التي ستنفذ. النهج التقليدي لتخطيط مسار لا يعمل في بيئة تكون ديناميكية وغير مؤكدة. لأنها تتطلب حسابات كثيرة ومعقدة جدا يتم تنفيذها في الوقت الحقيقي. الطرق التي تقوم على مبدأ رد الفعل البحثة لتجنب العقبات هي فقط القابلة للتطبيق. لكن هذه الأخيرة لا تتناسب إذا كانت مقيدة بإنجاز مهمة مثل الوصول إلى هدف أو موضع معين. المناهج الحديثة تميل إلى الجمع بينهما في طرق مختلطة تعرف بالتكرارية أوالحد من تعقيد وضع نماذج للمساحة الحرة باستعمال النهج الإحصائي. عملنا يتمثل في تخطيط حركة روبوت متحرك للوصول إلى هدف معين مع تجنب العقبات في طريقه. لقد اقترحنا الجمع بين طريقة محلية (تقوم على مبدأ رد الفعل) لتجنب العقبات و مخطط شامل يقوم على مبدأ انتشار الموجة.

الكلمات الرئيسية : الروبوت المتحرك ، التخطيط ، تجنب عقبة ، البيئة الديناميكية.

Abstract

For a robot operating in a given environment, the general problem of planning is to determine the robot motion allowing it to travel between two given configurations under a number of constraints and criteria. These arise from several factors of various kinds and often depend on the characteristics of the robot, the environment and the type of tasks to perform. Traditional trajectory planning approaches do not work in environment that is both dynamic and uncertain. Calculations are too numerous and complex to be executed in real time. Only purely reactive methods for obstacles avoidance are applicable. These are poorly suited to the consideration of tasks constraint like reaching a goal or a given position. Recent approaches tend to combine them in hybrid methods known as iterative methods or reduce the complexity of modeling the free space by statistical approaches. Our work consists of planning the movement of a mobile robot to reach a goal while avoiding obstacles in its path. We have proposed an approach to combine a local method (reactive) for obstacle avoidance with global planner type wavefront.

Keywords: mobile robot, planning, obstacle avoidance, dynamic environment.

Résumé

Pour un robot évoluant dans un environnement donné, le problème général de planification consiste à déterminer pour le robot un mouvement lui permettant de se déplacer entre deux configurations données tout en respectant un certain nombre de contraintes et de critères. Ceux-ci découlent de plusieurs facteurs de natures diverses et dépendent généralement des caractéristiques du robot, de l'environnement et du type de tâches à exécuter. Les approches classiques de planifications de trajectoires ne fonctionnent pas en environnement à la fois dynamique et incertain: les calculs sont trop nombreux et complexes pour être exécutés en temps réel. Seules les méthodes purement réactives d'évitement d'obstacles sont applicables. Ces dernières sont cependant mal adapté a la prise en compte de contrainte de tâches comme atteindre un but ou une position donnée. Des approches récentes tendent à les combiner dans des méthodes hybrides appelées méthodes itératives ou à réduire la complexité de la modélisation de l'espace libre par des approches statistiques. Notre travail consiste en la planification du mouvement d'un robot mobile devant arriver à un but tout en évitant des obstacles sur son chemin. Nous avons proposé de combiner une approche locale (réactive) pour l'évitement d'obstacle avec un planificateur global de type propagation de vague.

Mots clés: Robot mobile, planification, évitement d'obstacle, environnement dynamique.

Remerciements

Je tiens à remercier vivement le professeur ***M.T. Laskri*** pour son esprit scientifique et compréhensif qui m'a beaucoup aidé avec ses idées, ses conseils et surtout ses critiques.

Je tiens également à remercier les membres de notre groupe pour leur coopération.

Je remercie le docteur ***R.Boudour*** d'avoir accepté la présidence du jury, Le docteur ***M.T. Kimour***, le docteur ***S. Ghanemi*** et le docteur ***A. Amirat*** d'avoir pris de leurs temps pour examiner mon mémoire.

A toute ma famille.
A tous mes amis.

Table des Matières

Liste des Figures

Liste des Algorithmes

Liste des Abréviations

RMA	Robot Mobile Autonome
CIR	Centre Instantané de Rotation
AGV	Automated Guided Vehicle
VFH	Vector Field Histogram
NF1	Navigation Function
CV	Curvature Velocity
LCV	Lane Curvature Velocity
DW	Dynamic Window
ND	Nearness Diagram
EB	Elastic Band
RCP	Robot Center Point

Liste des Symboles

W	Espace de travail
C	Espace des configurations
C_{free}	Espace des configurations sans collisions
CT	Espace des configurations-temps
S	Espace des états
ST	Espace des états-temps
U	Espace des commandes immédiates
Γ	Espace des trajectoires de fuite
v	Vitesse de translation
w	Vitesse de rotation
gC	Grille de cellules rectangulaires
gC_{free}	Cellules libres
$gC_{occupied}$	Cellules occupées
gC_{border}	Cellules frontières
C_a	Région active de la grille
H^p	Histogramme polaire
$\beta_{i,j}$	Vecteur de direction
$C_{i,j}$	Cellule active
$m_{i,j}$	Amplitude du vecteur d'une cellule active
$c_{i,j}$	Valeur de certitude de la cellule active $C_{i,j}$
$d_{i,j}$	Distance de la cellule active $C_{i,j}$ au RCP
a	Résolution angulaire fixé à 5°
k	Secteur Angulaire
ρ	Angle discret $\rho = k \cdot a$.
r_r	Rayon du robot
d_s	Distance minimale entre le robot et un obstacle
r_{r+s}	Rayon d'élargissement des cellules d'obstacles
$\gamma_{i,j}$	Angle d'élargissement des cellules d'obstacles
h'	Densité d'obstacle polaire sert aussi comme un filtre passe-bas
τ	Seuil
H^b	Histogramme polaire binaire
H^m	Histogramme polaire masqué
k_n	Direction candidate
φ_n	Direction du mouvement $\varphi_n = \alpha . k_n$
C_n	Direction centre
C_r	Direction vers le coté droit
C_l	Direction vers le coté gauche
C_t	Direction du but
g	Fonction de coût

Introduction Générale

Les approches classiques de planifications de trajectoires ne fonctionnent pas en environnement à la fois dynamique et incertain: les calculs sont trop nombreux et complexes pour être exécutés en temps réel. Seules les méthodes purement réactives d'évitement d'obstacles sont applicables. Ces dernières sont cependant mal adaptées à la prise en compte de contrainte de taches comme atteindre un but ou une position donnée. Par conséquent un des grands problèmes ouverts de la robotique mobile autonome est celui de la navigation autonome pour ces environnements particuliers. La littérature oppose généralement deux grandes approches pour calculer un déplacement sûr du robot indépendamment du type d'environnement considéré :

Méthodes d'évitement d'obstacles dites locales.

Méthodes de planification de trajectoire ou globales.

Des approches récentes tendent à les combiner dans des méthodes hybrides appelées méthodes itératives ou à réduire la complexité de la modélisation de l'espace libre par des approches statistiques.

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 : 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 prévoir les 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.

Notre travail consiste en la planification du mouvement d'un robot mobile devant arriver à un but tout en évitant des obstacles sur son chemin. Nous avons proposé de combiner une approche locale (réactive) pour l'évitement d'obstacle avec un planificateur global de type propagation de vague.

La suite du document est organisée comme suit: un premier chapitre sur des notions de la robotique, le deuxième chapitre donne une revue des méthodes d'évitement d'obstacles et méthodes hybrides suivi d'un troisième chapitre sur les techniques de planification de chemin. Le quatrième chapitre est consacré à la présentation de notre approche de l'évitement d'obstacles par un robot mobile. Dans le dernier chapitre nous donnerons une description de l'environnement de simulation et les résultas obtenus. Nous terminerons par une conclusion et des perspectives.

Chapitre 1

Notions de Robotique

1.1 Introduction

Robot Mobile Autonome (RMA)

Depuis toujours l'homme a su se doter d'outils performants pour prolonger sa main et réaliser des actions qui n'auraient pas été possible sans. Cette évolution est motivée par la nécessité de satisfaire des besoins impératifs, mais également par l'envie de disposer d'un outil assez performant pour pouvoir le seconder voir le remplacer totalement dans toutes les tâches ingrates ou dangereuses.

Le terme « Robot » correspond à cette idée d'esclave mécanique. D'après le Larousse, il s'agit d'un « appareil automatique pouvant se substituer à l'homme pour exécuter diverses tâches ». Crée par Karel Capek en 1927, le mot vient de « Robuta » signifiant « travaux forcés » en tchèque.

Au début du XXéme siècle, le développement de la robotique répond essentiellement à un besoin de production dans les usines. Les robots ne se déplacent pas ou seulement dans un espace connu et parfaitement contrôlé. Ils sont dépourvus de toute intelligence, et leur rôle se limite à l'accomplissement d'actions entièrement prédéfinies. Cette simplicité des robots réduits à l'état d'automates, est rendue possible grâce à des environnements étudiés spécifiquement pour chaque robot et chaque application. En contrepartie, tout un imprévu implique un blocage du robot. En marge de ce mouvement apparaissent, dans les laboratoires de recherche, les premiers rares robots mobiles capables de réagir de manière simple et pré-conditionnée à un environnement moins structuré.

Avec l'arrivée de l'électronique, ils deviennent capables de suivre des lignes au sol ou de réagir à la lumière, à la chaleur ou à la présence d'un obstacle situé sur leur route. Il reste toutefois plus proche des automates mécaniques du moyen-âge que des robots pensants de Capek.

C'est l'informatique qui va finalement permettre d'apporter aux robots « l'intelligence » qui leur manquait, c'est à dire la capacité de « raisonner » sur leur environnement. Le premier robot mobile capable de percevoir puis de modéliser son environnement pour décider seul de son prochain déplacement, est créé en 1967. Il est baptisé *Shakey*. L'absence d'intervention humaine dans le processus de décision du robot lui confère la dénomination de « Robot Autonome ».

Les premiers robots mobiles autonomes (RMA) ont en fait une autonomie très restreinte. Les moyens de perception et de calcul de *Shakey* par exemple, sont encore limités et planifier un mouvement lui prend des heures. De plus tout changement dans l'environnement l'oblige à s'arrêter pour planifier un nouveau mouvement. Se déplacer en présence d'obstacles mobiles lui est par exemple impossible.

Les premières approches proposées pour permettre aux « RMA » de se débrouiller seuls, consistent à les aider en adaptant leur environnement comme pour les robots industriels.

Les technologies et les techniques évoluant, les robots voient plus de choses plus rapidement, pensent d'avantage et plus vite, et l'exploration d'espaces moins travaillés et moins figés devient envisageable. Une réelle autonomie de robots offrirait de nombreuses perspectives et dans cet espoir, la robotique mobile autonome devient une discipline à part entière.

Les robots de service (pour aider dans les hôpitaux, les aéroports, les bureaux, les usines ou à domicile), les robots de maintenance pour les milieux dangereux ou difficile d'accès pour l'homme (fonds sous-marins, terrains minés, espace, centrales nucléaires), les robots de divertissement (comme les robots-chiens de Sony apparus récemment en provenance du Japon) ou encore les moyens de transports intelligents (véhicules autonomes ou systèmes d'assistance) sont quelques exemples d'applications potentielles des RMA.

Les résultats obtenus dans ces domaines au cours des dernières années témoignent des progrès considérables réalisés. Pourtant, depuis *Shakey*, peu de RMA sont réellement sortis des laboratoires pour se confronter au monde réel. La raison principale en est le nombre important de problèmes technologiques et scientifiques encore non résolus pour des environnement *incertains* (informations incomplètes, bruitées, imprécises et/ou imprédictibles) *dynamiques* (présence d'obstacles mobiles, changeant, déformables et/ou déplaçables) et *ouverts* (virtuellement illimités). De telles caractéristiques définissent les milieux dans lesquels l'homme évolue et auxquels les RMA sont destinés.

De manière générale, on regroupe sous l'appellation robots mobiles l'ensemble des robots à base mobile, par opposition notamment aux robots manipulateurs.

L'usage veut néanmoins que l'on désigne le plus souvent par ce terme les robots mobiles à roues. Les autres robots mobiles sont en effet le plus souvent désignés par leur type de locomotion, qu'ils soient marcheurs, sous-marins ou aériens.

On peut estimer que les robots mobiles à roues constituent la grande partie des robots mobiles. Historiquement, leur étude est venue assez tôt, suivant celle des robots manipulateurs. Leur faible complexité en a fait de bons premiers sujets d'étude pour les roboticiens intéressés par les systèmes autonomes.

Cependant, malgré leur simplicité apparente, ces systèmes ont soulevé un grand nombre de problèmes difficiles. De ce fait, les applications industrielles utilisant des robots mobiles sont rares. Cela est dû au fait que, contrairement aux robots manipulateurs qui travaillent exclusivement dans des espaces connus et de manière répétitive, les robots mobiles sont destinés à évoluer de manière autonome dans des environnements dynamiques qui peuvent ne pas être connus. Néanmoins, l'intérêt indéniable de la robotique mobile est d'avoir permis d'augmenter considérablement les connaissances sur la localisation et la navigation des robots mobiles autonomes.

1.2 Définitions

En général, on peut définir un robot mobile [1] comme étant 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.

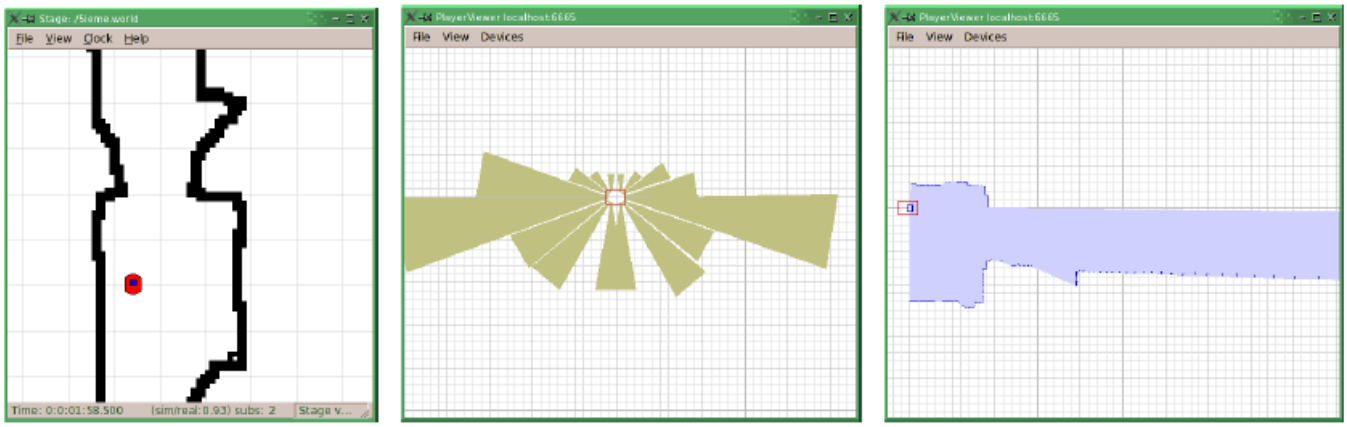
En particulier, un robot mobile autonome est un système mécanique, électronique et informatique complexe mettant en œuvre :

- Un ensemble de capteurs (extéroceptifs et/ou proprioceptifs) :

Les capteurs ont pour fonction d'acquérir des données provenant de l'environnement. Les capteurs typiquement installés sur un robot mobile (voir Figure 1.1) sont des sonars à ultrasons, un dispositif à balayage laser, des encodeurs de roues (odomètres), une ou deux caméras optiques et des microphones. Les types d'informations perçues ainsi que leurs précisions varient beaucoup d'un capteur à l'autre : comme montré à la figure 1.2, on peut voir qu'un dispositif à balayage laser permet de mieux percevoir les contours de l'environnement que les sonars [2].

Les capteurs extéroceptifs ont pour objectif d'acquérir des informations sur l'environnement proche du robot. Les capteurs proprioceptifs fournissent des données sur l'état interne du robot (telles que sa vitesse ou sa position).

Fig.1.1 – *Capteurs typiques d'un robot.*



(a) Simulateur Stage (b) Vue des sonars (c) Vue du laser

Fig.1.2 – *Perceptions des quelques capteurs.*

- Un ensemble d'effecteurs :

L'objectif du robot est d'atteindre un objectif dans son environnement en évitant les obstacles. Le problème que l'on doit résoudre est de déterminer en fonction des données capteurs quelles commandes doivent être envoyées à chaque instant au robot pour atteindre cet objectif. Ces effecteurs ont comme but de permettre au robot d'évoluer dans un monde prévu à l'origine pour l'homme. Les plus courants sont les systèmes à roues, mais il existe aussi des robots à chenilles, à pattes ou se déplaçant par reptation.

Le type de locomotion définit deux types de contraintes :

- Les contraintes cinématiques, qui portent sur la géométrie des déplacements possibles du robot.
- Les contraintes dynamiques, liées aux effets du mouvement (accélérations bornées, vitesses bornées, présence de forces d'inertie ou de friction).

Selon sa cinématique, un robot est dit :

- **holonome**, s'il peut se déplacer instantanément dans toutes les directions.
- **non holonome**, si ses déplacements autorisés sont des courbes dont la courbure est bornée.

1.3 Les robots mobiles à roues

Les roues sont le moyen de locomotion le plus répandu en matière de robotique mobile. En fait, les robots mobiles à roues sont faciles à réaliser et présentent de grandes possibilités de déplacement et de manœuvrabilité avec une vitesse et une accélération importantes. Bien évidemment, pour un ensemble donné de roues, toute disposition ne conduit pas à une solution viable. Un mauvais choix peut limiter la mobilité du robot ou occasionner d'éventuels blocages.

Par exemple, un robot équipé de deux roues fixes non parallèles ne pourrait pas aller en ligne droite ! Pour qu'une disposition de roues soit viable et n'entraîne pas de glissement des roues sur le sol, il faut qu'il existe pour toutes ces roues un unique point de vitesse nulle autour duquel tourne le robot de façon instantanée. Ce point, lorsqu'il existe, est appelé centre instantané de rotation (CIR).

Les points de vitesse nulle liés aux roues se trouvent sur leurs axes de rotation, il est donc nécessaire que le point d'intersection des axes de rotation des différentes roues soit unique. Pour cette raison, il existe en pratique trois principales catégories de robots mobiles à roues, que l'on va présenter par la suite. En général, les robots mobiles à roues ont des structures à locomotion différentielle.

En ce qui concerne la commande des moteurs, cette indépendance permet au robot mobile de changer de direction en jouant uniquement sur les vitesses des deux roues motrices. Les roues folles sont libres, leur rôle essentiel est d'assurer la stabilité de la plateforme du robot.

1.3.1 L'unicycle

On désigne par unicycle un robot actionné par deux roues indépendantes et possédant éventuellement un certain nombre de roues folles assurant sa stabilité. Le schéma d'un robot de type unicycle est donné à la figure 1.3. On y a omis les roues folles, qui n'interviennent pas dans la cinématique, dans la mesure où elles ont été judicieusement placées. Ce type de robot est très répandu en raison de sa simplicité de construction et de ses propriétés cinématiques intéressantes.

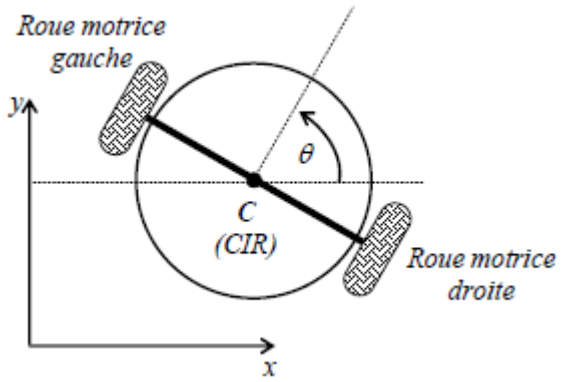


Fig. 1.3 – *Robot mobile de type unicycle.*

1.3.2 Le tricycle

L'architecture d'un robot mobile de type tricycle est donnée sur la figure 1.4, il utilise une roue motrice avant et deux roues passives arrières (ou vice versa). Le mouvement est conféré au robot par deux actions : la vitesse longitudinale et l'orientation de la roue orientable. Ce type de robots est très connu dans les applications des AGVs pour leur simplicité inhérente. Pour la commande et la localisation ce type de robot : un capteur d'orientation est associé à la roue orientable, et deux encodeurs sont associés aux roues motrices.

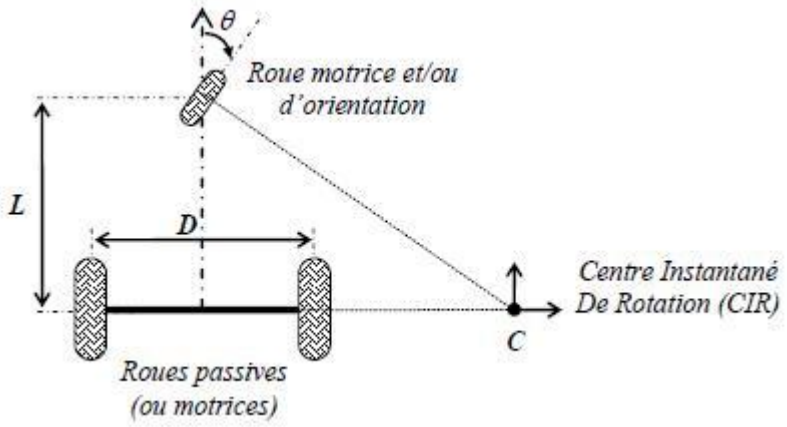


Fig.1.4 – *Robot mobile de type tricycle.*

1.3.3 Les véhicules (structure d'Ackerman)

Le cas du robot de type voiture est très similaire à celui du tricycle. La différence se situe au niveau du train avant, qui comporte deux roues au lieu d'une. Cela va de soit, on rencontre beaucoup plus souvent ce type de systèmes.

On parle de robot dès lors que la voiture considérée est autonome, donc sans chauffeur ni télé-pilotage. Il s'agit là d'un des grands défis issus de la robotique mobile. La figure 1.5 représente ce type d"architecture.

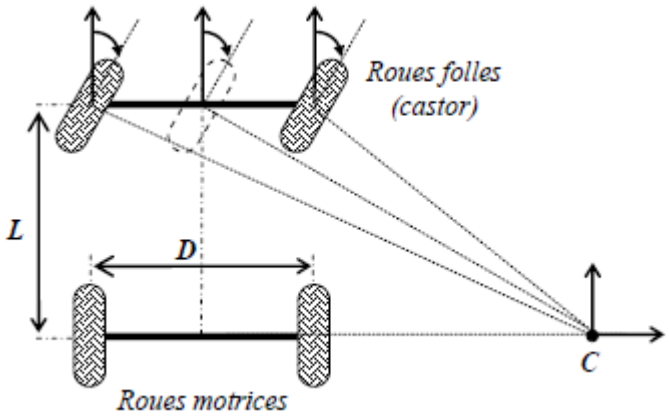


Fig.1.5 – *Robot mobile de type voiture.*

1.3.4 Robot à traction synchrone

La traction synchrone est une technique utilisée pour minimiser l'effet de glissement et augmenter la force de traction. On rencontre ce type de robot dans l'industrie automobile, et dans les robot tout terrains. La configuration du robot à traction synchrone est similaire à un robot à trois roues couplées de façon qu'elles soient actionnées en même temps. La figure 1.6 montre un robot à quatre roues couplées avec des chaînes.

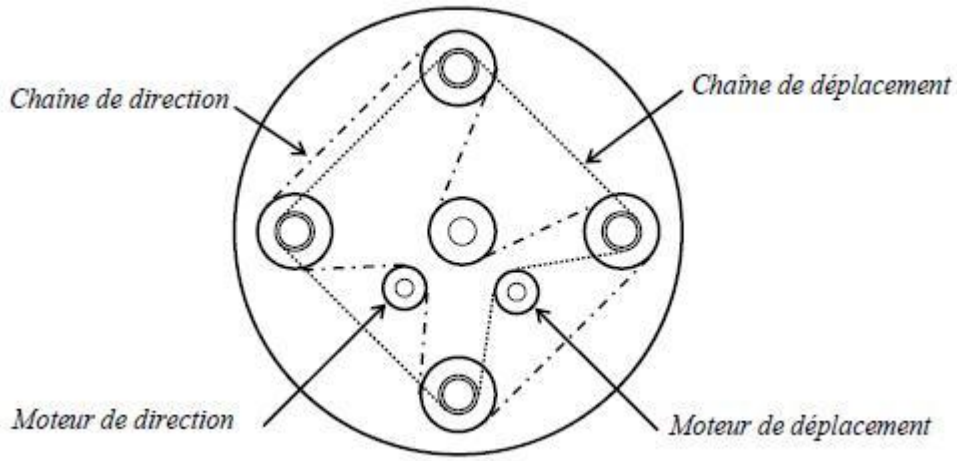


Fig.1.6 – *Robot mobile à traction synchrone.*

1.4 La perception

La notion de perception en robotique mobile est relative à la capacité du robot à recueillir, traiter et mettre en forme des informations qui lui sont utiles pour agir et réagir dans l'environnement qui l'entoure. Elle est donc la faculté de détecter et/ou appréhender l'environnement proche ou éloigné du robot. Alors que pour des tâches de manipulation on peut considérer que l'environnement du robot est relativement structuré, ce n'est plus le cas lorsqu'il s'agit de naviguer de manière autonome dans des lieux très partiellement connus. Aussi, pour extraire les informations utiles à l'accomplissement de sa tâche, il est nécessaire que le robot dispose de nombreux capteurs mesurant aussi bien son état interne que l'état de l'environnement dans lequel il évolue. Le choix des capteurs dépend bien évidemment de l'application envisagée.

La perception est nécessaire pour la sécurité du robot, la modélisation de l'environnement et l'évitement et le contournement d'obstacles.

Les moyens utilisés pour la perception de l'environnement sont nombreux et variés, parmi ceux-ci nous pouvons citer :

- Les systèmes de vision,
- Les télémètres laser et ultrasonores,
- Les capteurs optiques et infrarouges,
- Les capteurs tactiles.

1.5 La localisation

La localisation d'un robot mobile s'effectue par des capteurs proprioceptifs et/ou extéroceptifs. Cette localisation est d'autant plus nécessaire que le lieu d'évolution est encombré et complexe. Le comportement de l'être vivant illustre bien ces propos, en effet il doit toujours connaître sa situation pour se déplacer d'un point à un autre, soit en identifiant des repères artificiels, on parle de localisation absolue, soit tout simplement en mesurant les distances parcourues et les directions empruntées depuis sa position initiales. Un robot mobile doit connaître ses coordonnées de position pour être autonomes vis-à-vis de l'espace et de l'intervention humaine.

Les premières applications de robotique mobiles consistaient essentiellement en des tâches répétitives. Ces robots nécessitaient soit l'intervention humaine à distance pour beaucoup d'opérations, un environnement structuré avec des systèmes de guidage passifs ou actifs.

Le filoguidage, ou guidage inductif, est une des techniques les plus utilisées pour les robots mobiles d'atelier. Le principe consiste à encastrer, dans le sol, un fil parcouru par un courant alternatif. Une force électromotrice est induite dans deux bobines disposées sous le robot de part et d'autre du fil inductif. Le guidage consiste à minimiser la différence entre les deux forces électromotrices induites afin de maintenir par asservissement le robot mobile sur la trajectoire matérialisée par le fil. Ce fil sert également à transmettre des informations au robot. La modification de la trajectoire avec ce système de guidage nécessite des travaux importants qui peuvent entraîner l'arrêt de toute ou d'une partie de la production.

Dans le cas du guidage optique, la position relative du robot mobile est obtenue par la réflexion d'une onde lumineuse sur une piste matérialisée par des peintures ou des bandes adhésives réflectrices disposées sur le sol. Le signal détecté est maximal si le faisceau lumineux se réfléchit sur la bande, et il est minimal quand le faisceau se réfléchit en dehors de la bande. Des marquages particuliers sont disposés en certains endroits stratégiques pour informer le robot mobile sur l'évolution de la trajectoire, sur sa position ou sur le point d'arrêt.

Ce système ne fonctionne correctement que si le contraste entre le sol et le marquage au sol est suffisant, ainsi, il ne supporte pas les poussières et convient mal au milieu industriel. La lourdeur de ces systèmes de guidage : statisme de l'installation, difficultés pour le faire évoluer ... etc., ont conduit les chercheurs à étudier d'autres systèmes plus souples d'utilisation et plus performants [3]. C'est ainsi que les méthodes de localisation se regroupent en deux catégories, soit :

- La localisation à l'estime ou relative qui est obtenue par des informations issues de capteurs proprioceptifs.
- La localisation absolue qui est obtenue par des informations issues de capteurs extéroceptifs.

1.5.1 Localisation relative ou à l'estime

Cette méthode est basée sur l'intégration des déplacements élémentaires du robot mobile, on l'appelle aussi localisation relative car les coordonnées du robot sont calculées en fonction de la position précédente et du déplacement en cours. Les erreurs dues à cette méthode peuvent être importantes car cumulatives avec la distance et fonction du type de trajectoires. On distingue deux méthodes principales de localisation relative :

- La méthode odométrique (directe ou indirecte)
- La méthode inertielle.

1.5.1.1 Odométrie directe

La technique de l'odométrie est très utilisée pour localiser les robots mobiles, elle présente l'avantage d'être simple d'emploi et d'un coût faible. Son principe repose sur la mesure de la vitesse angulaire de rotation des roues motrices. Pour un robot mobile à roues non orientables, on déduit à partir de la somme et de la différence des vitesses de rotation de chaque roue, la vitesse linéaire du robot ainsi que sa vitesse angulaire. Le déplacement du robot est obtenu par l'intégration des déplacements élémentaires entre deux instants et ainsi de proche en proche on obtient une estimation de la position à partir d'une référence initiale. Ceci suppose de parfaites conditions d'évolution.

La mesure des angles est généralement faite par des codeurs optiques incrémentaux couplés à des roues qui peuvent être distinctes des roues motrices. Les erreurs de cette technique peuvent être de différentes natures. Elles peuvent être systématiques et constantes ou non selon qu'elles sont liées à des paramètres du robot mobile (longueur de l'entraxe, diamètre des roues...) ou à l'environnement. Ainsi, l'état du terrain introduit des erreurs de même que le glissement des roues. Des erreurs peuvent provenir de la résolution des capteurs numériques de mesure et de la discrétisation des calculs.

1.5.1.2 Odométrie indirecte

La localisation relative par odométrie indirecte s'effectue par des moyens inertiels qui mesurent soit la vitesse soit l'accélération puis par intégration, on en déduit les déplacements élémentaires. Ces moyens permettent une localisation précise pour certains engins parcourant de longues distances.

Il existe d'autres méthodes de localisation relative, mais toutes présentent des dérives assez importantes qui ne peuvent être compensées que par des informations supplémentaires issues d'un autre mode de localisation.

1.5.2 Localisation absolue

Les inconvénients cités ci-dessous ont conduit à développer des systèmes de localisation absolue qui fournissent la position et l'orientation du robot mobile par rapport à des points fixes du repère de travail donc en coordonnées absolues.

Pour se faire le robot doit effectuer des mesures par rapport à l'environnement. Ces mesures sont fournies par des capteurs dits extéroceptifs ou externes car ils ne s'intéressent pas aux mouvements internes du robot, comme c'est le cas pour l'odométrie [3].

1.6 La navigation

Les sections précédentes ont permis de mettre en place les outils nécessaires pour faire naviguer un robot mobile dans un environnement d'intérieur : la compréhension du mode de locomotion et de localisation de ce robot dans son environnement. Il s'agit maintenant d'utiliser au mieux la motricité du robot et sa localisation pour accomplir la tâche de navigation de manière autonome.

Un mouvement est une application définie en fonction du temps ***t***, reliant un point initial à l'instant ***t0*** à un point final à l'instant t_f. Une trajectoire est le support d'un mouvement. Il s'agit donc d'une courbe paramétrée par une variable ***s*** quelconque, par exemple l'abscisse curviligne normalisée

($s \in [0,1]$) de la courbe sur laquelle se déplace le robot. L'évolution du paramètre ***s*** en fonction du temps ***t*** est appelée mouvement sur la trajectoire.

Le problème de navigation d'un robot mobile consiste de la manière la plus générale à trouver un mouvement dans l'espace des configurations sans collisions, traditionnellement noté C_{free}. Ce mouvement amène le robot d'une configuration initiale $q_0 = q(t_0)$ à une configuration finale $q_f = q(t_f)$.

On peut néanmoins donner des définitions différentes de la tâche de navigation à accomplir, selon le but recherché par exemple on peut souhaiter seulement placer le robot dans une zone donnée et relâcher la contrainte d'orientation, etc.

La tâche de navigation ainsi définie est donc limitée à un seul mouvement. Il existe néanmoins une très grande variété de travaux et de méthodes permettant d'aborder ce problème difficile. Pour différencier les techniques de navigation, on peut de manière générale distinguer deux approches :

– la première consiste à planifier le mouvement dans l'espace des configurations et à l'exécuter par asservissement du robot sur le mouvement de consigne (schéma planification-exécution);

– la seconde consiste à offrir un ensemble de primitives plus réactives. Elles correspondent alors à des sous tâches (suivre un mur, éviter un obstacle) dont on estime que l'enchaînement est du ressort d'un planificateur de tâches ayant décomposé la tâche globale.

1.7 Boucle de contrôle

Un robot mobile est commandé par une boucle de contrôle, comme illustré à la figure 1.7. De façon itérative, celle boucle fait une lecture des données reçues par les capteurs, les interprète, calcule les commandes motrices et les envoie aux actionneurs. Typiquement, cette boucle est exécutée environ dix (10) fois par seconde ; la fréquence peut varier selon les types de capteurs et d'actionneurs utilisés. La boucle de contrôle n'est pas unique ; selon l'architecture utilisée, elle peut être décomposée en plusieurs sous boucles de contrôle agencées de manières différentes.

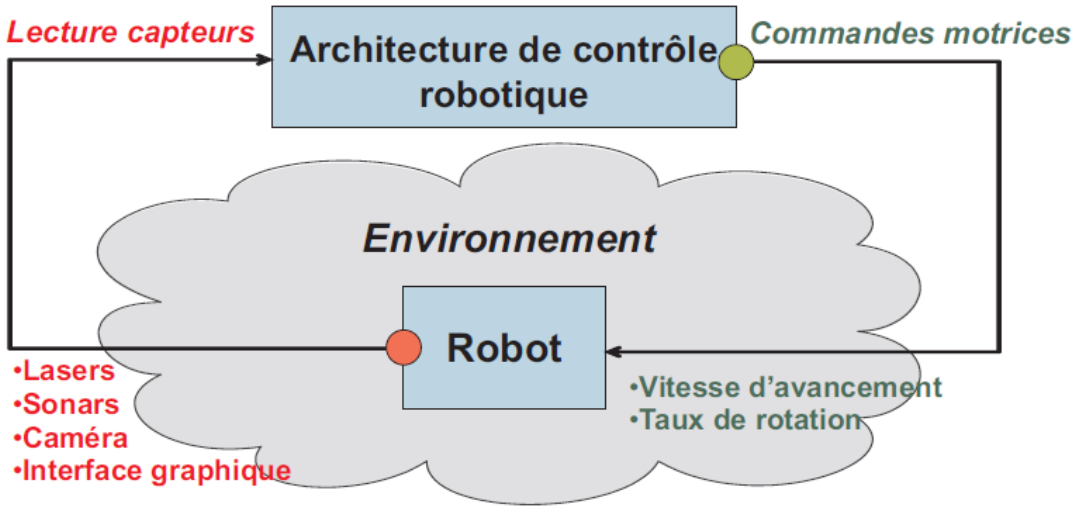


Fig.1.7 – *Boucle de contrôle.*

1.8 Architectures décisionnelles

Un robot est donc composé d'un ensemble de modules, chacun étant responsable d'une ou plusieurs capacités. Un des premiers défis à résoudre est de déterminer comment relier efficacement les différents modules. Pour ce faire, il faut élaborer une architecture décisionnelle qui dictera les responsabilités de chacun des modules et comment les informations circuleront entre ces derniers. Depuis les débuts de la robotique, beaucoup d'architectures ont été proposées. Elles peuvent être généralement classées en trois grandes catégories : délibérative, comportementale et hybride.

1.8.1 Architecture délibérative

Les architectures délibératives [4] sont les premières à avoir été proposées. Comme son nom l'indique, les architectures de ce type sont basées sur des processus complètement planifiés. Par exemple, afin d'exécuter un déplacement, un robot basé sur ce type d'architecture calcule un plan complet, lui disant d'avancer de tant de mètres, ensuite de tourner de tant de degrés, et ainsi de suite. Lorsqu'un changement dans l'environnement est perçu, l'exécution est suspendue et un nouveau plan est généré.

Limitations

Ce type d'architecture souffre de plusieurs lacunes importantes. Premièrement, puisque les capteurs sont imprécis, et que l'environnement est dynamique et partiellement observable, il est très difficile de tout prévoir à l'avance. Pour ces raisons, il n'est pas d'une très grande utilité de tout planifier à l'avance puisque les plans seront constamment à refaire. Un autre problème avec ce type d'architecture est que la génération de plans précis demande beaucoup de ressources (temps de calcul et mémoire).

1.8.2 Architecture comportementale

L'architecture comportementale, proposée par [5], est inspirée par le comportement des insectes [6]. L'idée générale est de développer plusieurs petits modules simples et indépendants les uns des autres et, une fois regroupés, un comportement plus intelligent émerge sans qu'il ait été spécifiquement programmé. Ce type d'architecture est complètement à l'opposé des architectures délibératives et ne fait aucune place à des processus raisonnés. Un exemple d'architecture comportementale est illustré à la figure 1.8. Elle est composée de plusieurs modules simples appelés comportements et d'un module d'arbitration.

Dans leurs conceptions, chaque comportement se limite à une seule fonctionnalité pour le contrôle du robot. Les comportements sont tous indépendants les uns des autres. Par exemple, on peut avoir des comportements pour l'évitement d'obstacles, le suivi de trajectoire, le suivi d'objets de couleur ou la manipulation d'objets. Les comportements sont exécutés parallèlement à une certaine fréquence habituellement dix fois par seconde).

Lors d'une itération, chaque comportement calcule une ou plusieurs commandes motrices qui sont envoyées à un module d'arbitration. Ce dernier fusionne l'ensemble des commandes reçues et calcule les commandes finales devant être envoyées à chacun des actionneurs du robot.

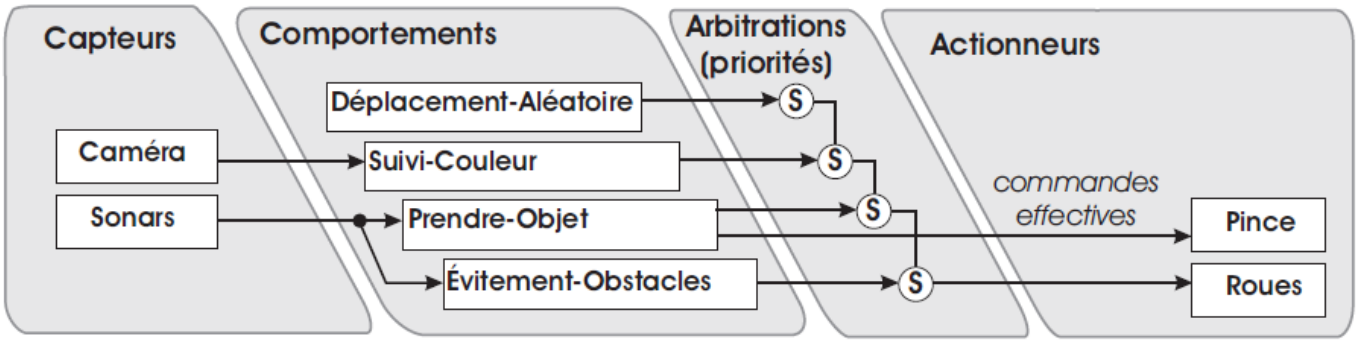
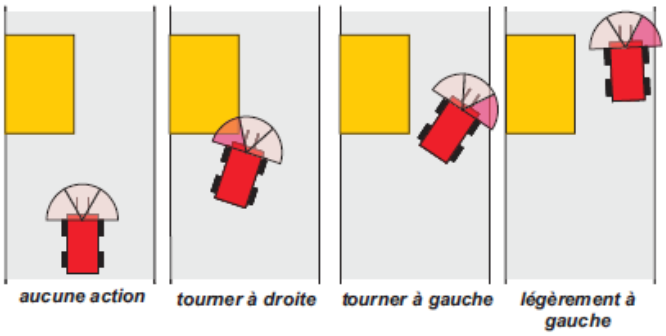


Fig.1.8 – *Exemple d'architecture comportementale.*

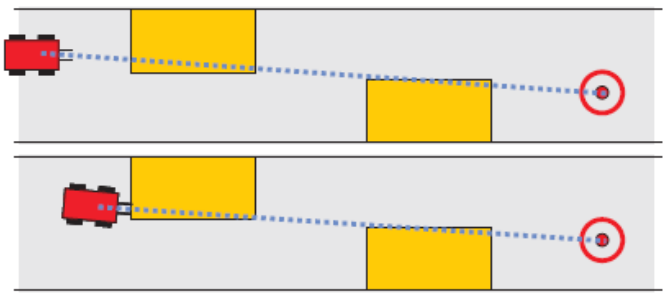
Exemples d'émergence de comportements

En guise d'exemple, examinons l'intégration d'un comportement d'évitement d'obstacles et d'un comportement de suivi de trajectoire. En présence d'un obstacle sur un côté (voir figure 1.9(a)), le comportement d'évitement d'obstacles ne fait que tourner vers le côté opposé. Indépendamment, le comportement de suivi de trajectoire (souvent appelé «Goto» en anglais) aligne et dirige en ligne droite le robot vers la cible courante.

Comme nous pouvons l'apercevoir à la figure 1.9(b), ce comportement ignore la présence d'obstacle. Afin de diriger le robot vers la cible sans collision, les capacités de ces deux comportements doivent être fusionnées par un mécanisme d'arbitration. L'approche classique consiste à utiliser une arbitration par priorité.



(a) Évitement d'obstacles



(b) Suivi de trajectoire

Fig.1.9 – *Exemples de comportements*.

Dans le cas présent, l'évitement d'obstacles a priorité sur le suivi de trajectoire. Lorsqu'aucun obstacle n'est détecté, le module d'évitement d'obstacles ne génère aucune commande et laisse le plein contrôle au module de suivi de trajectoire. Par contre, quand un obstacle est détecté, le module d'évitement substitue la commande du module de suivi de trajectoire par une commande de rotation dans le sens opposé à l'obstacle perçu. Lorsque l'obstacle est complètement évité et sorti du champ de vision des capteurs, le comportement de suivi de trajectoire reprend le plein contrôle du robot.

Limitations

Les architectures comportementales ont de la difficulté à réaliser des tâches structurées, puisqu'elles ne contiennent aucun processus délibératif. En effet, les tâches complexes requièrent la capacité du robot à prédire les conséquences futures de ses actions afin de sélectionner celles qui conviennent le mieux pour la réalisation de ses activités. En d'autres mots, ces tâches complexes ont besoin d'être planifiées. À cela, on peut souligner d'autres capacités délibératives manquantes, comme l'apprentissage machine.

1.8.3 Architecture hybride

Les limitations des deux types d'architecture précédentes justifient l'émergence récente des architectures hybrides [4], tentant de combiner les avantages des architectures délibératives et comportementales. Elles sont généralement décomposées en deux niveaux. Dans la partie supérieure, on place les modules de type délibératif. Dans la partie inférieure, on retrouve les modules de type comportemental. Comme montrée à la figure 1.10, la partie délibérative est représentée par un séquenceur.

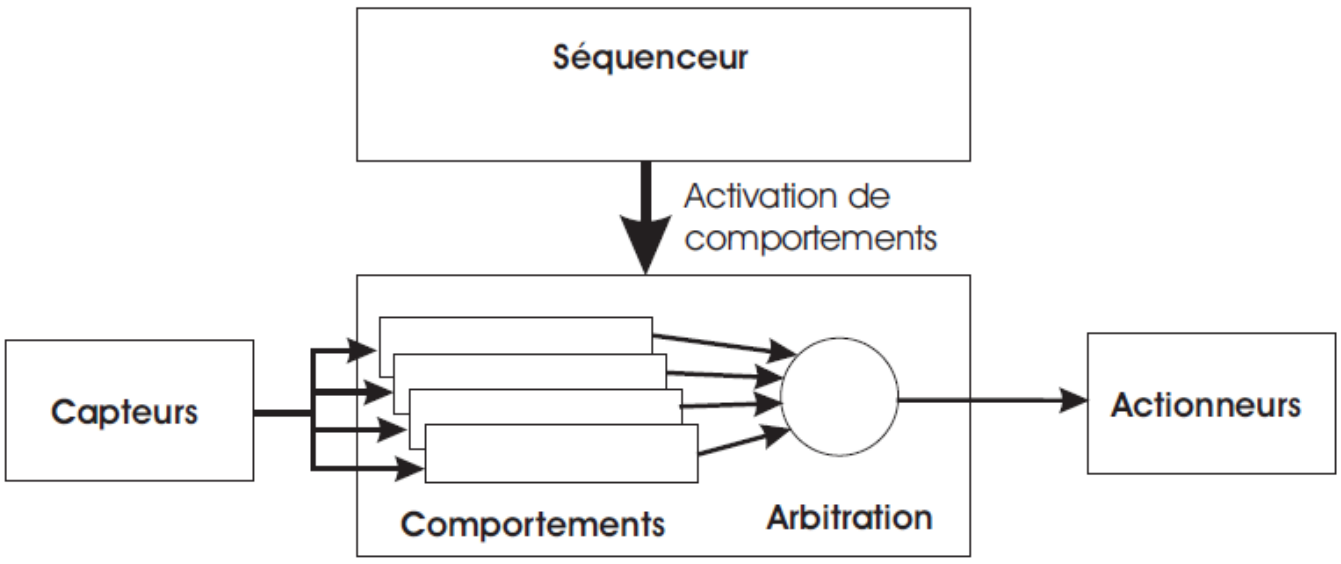


Fig.1.10 – *Architecture hybride.*

1.9 Conclusion

Les robots mobiles à roues sont les robots mobiles les plus répandus, vu leurs structures mécaniques simples et leur commande relativement plus facile que les autres robots mobiles qui diffèrent par leurs moyens de locomotion.

La commande d'un robot mobile se divise en trois étapes principales :

Perception, décision et action. La dernière étape concerne l'exécution des mouvements planifiés, c'est une étape que doit maîtriser un robot mobile pour accomplir ses missions avec succès.

Chapitre 2

Problématique et Approches Existantes

2.1 Environnement dynamique et incertain

2.1.1 Notion d'environnement dynamique

Nous dirons qu'un environnement est dynamique lorsque l'ensemble des position occupées par les obstacles est susceptible de changer au cours du temps. C'est le cas des obstacles qui se déplacent, de ceux qui changent de forme (e.g. un piéton tenant un chien en laisse est perçu par le robot comme un seul objet à géométrie variable) ou de ceux qui apparaître/disparaître (e.g. une porte coulissante semble "disparaître" dans le mur quand elle s'ouvre).

2.1.2 Notion d'incertitude

La notion d'incertitude est souvent sujette à discussions. Dans la suite de ce mémoire, nous dirons d'une manière générale qu'une information est *incertaine*, si elle est bruitée (mauvaises conditions de mesures), incomplète (obstruction d'un capteur ou portée limitée, absence d'informations sur l'évolution d'un objet ou d'un phénomène) ou imprécise (les glissements des roues par rapport au sol sont observable mais rarement mesurables avec précision).

2.1.3 Implications sur les traitements

L'absence de connaissance a priori sur l'environnement contraint le robot à faire des hypothèses qui peuvent s'avérer fausses par la suite. Une réactualisation permanente de sa connaissance ainsi qu'une capacité à réagir rapidement aux changements sont nécessaires dans ce contexte. En environnement statique, les hypothèses portent uniquement sur la position de robot et la présence de nouveaux obstacles à proximité du robot. Celles-ci sont levées au fur et à mesure de la progression du robot. En environnement dynamique, les hypothèses portent également sur les trajectoires des obstacles mobiles. Celles-ci doivent être prises en compte par le robot lors du choix d'un déplacement pour anticiper les collisions et assurer sa sécurité. En raison du nombre important de possibilités dans ce cas et de l'absence de connaissance a priori complète les erreurs sont nombreuses et la validité d'une solution est de courte durée.

En résumé, un environnement à la fois dynamique et incertain implique davantage de calculs (liés à l'anticipation des collisions) et demande une plus forte réactivité du robot et donc des temps de réponse plus courts (en raison du caractère éphémère des solutions).

2.2 Méthodes de génération de mouvement existantes

Le problème de la génération de mouvement au sens large consiste à calculer un déplacement sûr du robot puis à l'exécuter. Tandis que l'Homme apprend à lever rapidement les incertitudes grâce à ses expériences passées, cela représente un véritable challenge pour un RMA, dont le raisonnement purement logique et mathématique demande du temps. En environnement dynamique et incertain, les méthodes actuelles de planification d'un déplacement sûr ne sont pas assez rapides pour assurer la sécurité du robot. Seules les méthodes réactives le permettent, mais elles ne prennent pas en compte (ou mal) les contraintes de tâche comme atteindre un but.

Par conséquent, un des grands problèmes ouverts de la robotique mobile autonome est celui de la navigation autonome pour ces environnements particuliers [7]. Dans ce rapport, nous avons choisi d'étudier deux thèmes fondamentaux:
- le calcul d'un déplacement sûr du robot, compte tenu de sa connaissance actuelle sur lui-même et sur son environnement,
- le calcul de la commande du robot permettant de réaliser ce déplacement compte tenu de sa connaissance actuelle sur lui-même et sur ses interactions avec son environnement.

2.2.1 Calcul d'un déplacement sûr

La littérature oppose généralement deux grandes approches pour calculer un déplacement sûr, indépendamment du type d'environnement considéré.

- les méthodes d'**évitement d'obstacles**,
- les méthodes de **planification de trajectoires**.

Des approches récentes tendent à les combiner dans des méthodes hybrides appelés **méthodes itératives** ou à réduire la complexité de la modélisation de l'espace 1ibre par des **approches statistiques**.

2.2.1.1 Notion d'espace de recherche

Les traitements réalisés en navigation reposent essentiellement sur des outils de géométrie. Typiquement, la recherche d'un déplacement libre fait appel à des techniques classiques de test de collision (calculs d'intersections, calculs de distances). Le choix des espaces mathématiques dans lesquels sont réalisés les calculs est fonction de l'environnement et des informations à représenter (i.e. à prendre en compte). Leur dimension peut par conséquent être très élevée ce qui a une influence directe sur la quantité et la complexité des traitements. Sous contrainte de temps réel, des compromis

doivent être faits pour réduire cette complexité. Cela se traduit souvent par la recherche avant tout d'un espace de dimension plus réduite pour pouvoir maîtriser la complexité des calculs, tout en conservant les informations essentielles.

L'espace de modélisation le plus intuitif pour un robot est son espace de travail. Noté *W*. il représente l'ensemble des positions que ce dernier peut atteindre. Par conséquent sa dimension est faible et reste constante quelque soit la complexité du robot (2 pour les robots mobiles considérés dans ce rapport et 3 pour des robots qui se déplacent dans l'espace, comme un drone ou un bras de robot).

A chaque type d'environnement et à chaque traitement correspond un espace. *W* est généralement utilisé pour formuler le problème. En revanche, il ne permet pas de le traiter car la représentation du robot et des obstacles dans *W* n'est pas approprie pour les calculs. Nous donnons ici les principaux espaces utilisés en robotique à titre indicatif, avec leur utilisation respective la plus courante:

- **Espace des configurations** (***C***): La configuration d'un robot et l'ensemble des **n** paramètres qui permettent de définir de manière unique et sans équivoque, la posture (position+orientation) de chaque partie d'un robot dans son environnement. ***C*** permet de représenter par un point donné une configuration donnée. Sa dimension est donc **n**. Permettant de ramener un problème de planification complexe à un problème purement géométrique, ***C*** est très employé en robotique, en particulier pour la planification de chemin géométrique en environnement statique avec prise en compte de la cinématique du robot.
- **Espace des états** (***S***): L'état du robot peut être vu comme l'union des paramètres de configuration du robot, et de leurs dérivées (premières et éventuellement secondes) par rapport au temps. I1 permet de prendre en compte des contraintes dynamiques et par conséquent est utilisé pour la planification de trajectoire en environnement statique avec prise en compte de la cinématique et de la dynamique du robot.
- **Espace des configurations-temps** (***CT***): il s'agit de ***C*** augmenté d'une dimension de temps pour pouvoir traiter les environnements dynamiques, d'où son utilisation pour la planification de trajectoire en environnement dynamique avec prise en compte de la cinématique du robot.
- **Espace des états-temps** (***ST***): I1 s'agit de l'espace des états augmenté d'une dimension de temps pour les mêmes raisons que pour ***CT***. I1 est utilisé en planification de trajectoire en environnement dynamique avec prise en compte de la cinématique et de la dynamique du robot.
- **Espace des commandes immédiates** (***U***): Cet espace ne permet pas de représenter une trajectoire, mais uniquement le prochain déplacement élémentaire du robot. Il est de faible dimension et permet une représentation simple des contraintes sur le robot. Son utilisation typique est l'évitement d'obstacles en environnement statique ou dynamique avec prise en compte de la cinématique et de la dynamique du robot pour le prochain déplacement.

- **Espace des trajectoires de fuite** (***Γ***): Il s'agit d'une extension de ***U***. ***Γ*** contient un sous-ensemble des trajectoires immédiatement réalisables par le robot sur un intervalle de temps borné et généralement court. Le test d'une trajectoire au lieu de se limiter au prochain mouvement permet de mieux traiter les environnements dynamiques tout en conservant une dimension raisonnable. Cela permet surtout une meilleure prise en compte des contraintes cinématiques et de la dynamique du robot. ***Γ*** est utilisé par exemple pour l'évitement réactif d'obstacles.

2.2.1.2 Méthodes d'évitement d'obstacles

Les méthodes d'évitement d'obstacles sont des méthodes locales dont l'objectif est d'assurer la sécurité, du robot sur un temps très court. Elles peuvent être classées en deux catégories:

- Celles qui calculent directement une commande à partir des informations disponibles sur l'environnement. Il s'agit des approches de l'intelligence artificielle inspirées du vivant (réseaux de neurones artificiels [8], approche bayésienne, logique floue) ou des approches basées sur des phénomènes physiques naturels (comme les champs de potentiels [9]). Ces méthodes confèrent un comportement de base très réactif au robot, bien adapté aux environnements dynamiques. Cependant la prise en compte de contraintes de tâche (atteindre un but) est difficile voire impossible avec ces méthodes et nous ne nous y intéresserons pas davantage.
- Celles qui calculent un ensemble de solutions potentielles compte tenu des informations sur l'environnement, puis sélectionnent une solution particulière afin de satisfaire des contraintes de tâche. Les solutions peuvent prendre la forme d'une direction privilégiée du robot ou d'une consigne en vitesse. Nous allons nous intéresser ici à ces méthodes.

Leur principal avantage est leur grande réactivité. Elle est due au fait qu'en ne considérant que des déplacements locaux, le nombre de calculs est réduit. Le temps ainsi « économisé » permet de réduire les temps de réponse du robot ou de prendre en compte des paramètres tels que la cinématique et la dynamique du robot, pour améliorer la qualité (ie. l'exécutabilité) des mouvements proposés.

En contrepartie, leur caractère local leur confère un inconvénient majeur: les " minima locaux " de la fonction de sélection. En effet, un coût minimal peut être attribué à un déplacement répondant localement aux critères de choix désirés, mais qui, dans le futur, conduira le robot à une situation de blocage ou à une collision (figure 2.1).

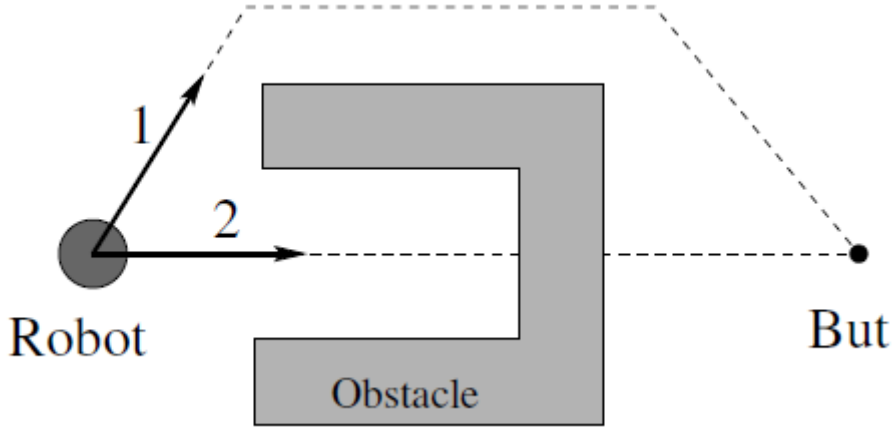


Fig.2.1 – *Minimum local Les déplacements 1 et 2 sont tous deux libres pour le robot. Localement, le déplacement 2 étant dans la direction du but, il est meilleur, pourtant, il conduit à un blocage.*

Une solution couramment rencontrée au cours des dernières années, consiste à étendre ces méthodes avec des techniques de graphe ou à les combiner avec un planificateur global utilisant une fonction de navigation classique NF1 de type « propagation de vague ». Nous ne citons ici que les approches les plus utilisées. Nous avons conservé leur dénomination anglosaxone lorsque la traduction française ne nous semblait pas naturelle:

- Vector Field Histogram (VFH) et extensions

Dans leur version d'origine, les **VFH** [10] consistent à représenter dans un premier temps l'environnement par une grille d'occupation. Chaque cellule contient une grandeur réelle correspondant en quelque sorte à la probabilité de trouver un obstacle à cet emplacement. La grille est ensuite traduite en un histogramme dont chaque « barre » représente une direction du robot et dont la hauteur est proportionnelle à la probabilité de percuter un obstacle en suivant cette direction. Par seuillage, il est ainsi possible d'identifier les directions libres du robot, appelées passages. Une fonction de coût prenant en compte la direction du but, la direction précédemment suivie et l'orientation courante des roues permet de sélectionner un des passages et la direction correspondante. La vitesse est ensuite calculée en fonction de la distance du robot aux obstacles. Une version plus récente (**VFH+**) [11] prend en compte la largeur du robot et ses contraintes cinématiques (les trajectoires ne sont plus assimilées à des droites mais à des arcs de cercles).

Un des inconvénients majeur de l'approche est la difficulté à régler le seuil déterminant les « passages » et la difficulté à naviguer en milieu contraint (couloirs étroits, passage de portes). De plus, la dynamique du robot n'est pas réellement prise en compte et le traitement purement local empêche de garantir la prise en compte des contraintes de tâches. Ce dernier point a toutefois été traité dans la

dernière version de cette approche sous la dénomination **VFH*** [12], où plusieurs déplacements sont testés à l'avance. Ceci a pour effet de limiter les situations de blocage et d'améliorer la convergence vers le but.

- Curvature Velocity (CV) et Lane Curvature Velocity (LCV)

L'idée consiste à déterminer non plus une orientation mais une commande. Elle permet de prendre en compte aussi bien des contraintes cinématiques et dynamiques sur le robot: L méthode des « Curvature Velocities » [13] représente les obstacles dans l'espace des vitesses de translation et de rotation ***(v, w)*** du robot. Une fonction de coût permet de prendre en compte des critères d'alignement avec le but comme pour les VFH, mais permet en plus de considérer la longueur des trajectoires libre avant que le robot n'entre en collision avec un obstacle.

La méthode de base a été étendue sous la dénomination de « Lane Curvature Velocity » [14]. Les trajectoires du robot considérées sont quelconques et les solutions permettent de passer plus loin des obstacles grâce à 1a notion de voie «Lane». Bien que la méthode soit purement locale et présente les inconvénients classiques de ce type d'approche, elle permet de mieux passer les passages étroits et prend mieux en compte les contraintes cinématiques et dynamiques du robot, que les précédentes.

- Fenêtres dynamiques (DW)

La méthode des fenêtres dynamiques [15] reprend en tout point le principe des CV y compris dans le choix de la fonction de coût mais dans un espace des vitesses ***(v,w)*** du robot discrétisé. Les contraintes dynamiques du robot sont prises en compte en limitant les vitesses du robot sélectionnables. Elles définissent une fenêtre dite dynamique dans l'espace des vitesses, autour de la vitesse courante du robot. Une vitesse est considérée potentiellement solution, si elle est admissible compte tenu de ces contraintes, et si elle permet au robot de s'arrêter avant de percuter un obstacle. Les fenêtres dynamiques existent en deux versions, pour robots holonomes ou non holonomes (dans ce cas les trajectoires du robot sont assimilées à des arcs de cercle).

L'inconvénient de cette approche réside dans le choix de la discrétisation de l'espace des vitesses. Ainsi une implémentation temps-réel nécessite de réduire la résolution ou la taille de la fenêtre dynamique. Limitée à un traitement local et donc sujette aux minima locaux dans sa version de base, cette approche a été étendue vers une version globale à l'aide d'une fonction NF1. Dans les expérimentations, celle-ci est calculée uniquement à partir de l'environnement perçu par le robot. Cela pose des problèmes de mise-à-jour de la carte en environnement dynamique, comme cela est noté dans [17]. Dans ce cas, l'approche est applicable en environnement statique ou peu dynamique, plutôt qu'en environnement réellement dynamique.

- Diagramme de proximité (ND)

Les ND [16] sont basés sur l'extraction d'informations de haut niveau sur l'environnement, à savoir l'identification d'une situation particulière parmi 5 prédéfinies. Cela permet de générer la commande la plus appropriée. La méthode comprend 4 étapes:

1. La construction de deux diagrammes polaires représentant la distribution des obstacles autour du robot et leur distance, respectivement au robot et à son point de référence.
2. La recherche de régions de passage à partir du second diagramme, et la sélection de l'une d'elles.
3. L'analyse du premier diagramme pour définir le niveau de sécurité du robot.
4. Enfin, la combinaison de ces informations pour identifier une situation prédéfinie parmi 5, et générer la commande appropriée à cette situation.

La première chose à remarquer est la phase d'analyse préalable, rencontrée nulle part ailleurs dans les autres approches. Cela permet au robot d'éviter les pièges classiques sur lesquelles les autres méthodes locales butent, tels que le traitement des obstacles en forme de U (figure 2.1), ou la navigation en milieu très contraint sans osciller. Nous pensons que cela fait de cette approche la plus efficace pour affronter des environnements totalement inconnus. Cependant, la survie du rabot, en milieu dynamique n'est due qu'à la réactivité de la méthode, comme pour les approches précédentes, et n'est pas due à la prise en compte explicite de cet dynamique. De plus la complexité des traitements implique des temps de réponse plus longs, qui peuvent à lui faire perdre son avantage en présence d'obstacles mobiles, par rapport à une approche moins « pensante » mais plus rapide telle que les ***DW***. Toute comme celles-ci, en revanche, elle a été récemment étendue à une version globale [17] dont elle a pris le qualificatif.

Le principe est le même à savoir l'utilisation combinée de l'approche réactive avec une fonction de navigation de type NF1. L'actualisation de la carte locale tire cependant davantage parti des ressources capteurs et permet un comportement plus naturel en milieu dynamique.

2.2.1.3 Méthodes de planification de trajectoire

Les méthodes de planification de trajectoire, par opposition aux précédentes, calculent un déplacement complet (global) jusqu'au but. Elles supposent une connaissance suffisante de l'environnement. Leur principe consiste à obtenir une représentation de l'espace libre et de sa connectivité pour y rechercher une trajectoire reliant le robot à son but. Différentes contraintes (cinématique, dynamique, incertitudes) peuvent être prises en compte lors de la recherche de la trajectoire dans l'espace libre.

Ces méthodes permettent d'éviter les problèmes de minima locaux et de proposer des solutions optimales, (L'optimalité d'une trajectoire peut concerner son temps de parcours par le robot ou sa longueur. Il s'agit dans les deux cas d'une trajectoire qui minimise ce critère.) Mais leur temps de réponse est généralement supérieur aux contraintes imposées pour la navigation:

La modélisation de l'espace libre dans le cas général nécessite en effet des calculs complexes. De plus, l'étendue de cet espace implique un nombre plus important de solutions potentielles à envisager. Les études menées sur le sujet sont moins nombreuses que pour les environnements statiques connus (introduites en annexe A), et d'une manière générale, ne sont pas utilisables en temps-réel. Nous citons cependant les plus courantes pour information.

Planification dans *CT* et *ST*

L'approche générale consiste à ajouter une dimension de temps à l'espace ***C*** des configurations, généralement utilisé pour la planification en environnement statique. Lorsque la dynamique du robot doit être prise en compte, l'espace étendu est ***S***. Les méthodes applicables dans **C** ont été étendues pour les nouveaux espaces, notamment la méthode du **graphe de visibilité** ou celle des **décompositions cellulaires**. Ces approches s'avèrent très coûteuses en calculs et non-envisageables pour une utilisation temps-réel autre que pour des cas très simplifiés.

Décomposition Chemin-vitesse

L'idée consiste à calculer, à l'aide des méthodes citées en annexe A, le chemin géométrique permettant d'éviter tous les obstacles statiques connus. Un profil de vitesse est ensuite calculé le long de ce chemin de manière à éviter les obstacles en mouvement. Il est réalisé dans un espace de dimension 2, dont l'abscisse est l'abscisse curviligne du chemin et l'ordonnée est le temps. Il s'agit d'une courbe monotone selon l'axe du temps. Les contraintes dynamiques imposent des bornes de sa pente. Cette approche a permis le calcul de trajectoires optimales en temps pour des robots dont les vitesses et les accélérations sont bornées, ainsi que pour des robots soumis à des contraintes dynamiques plus complètes. Dans ce cas cependant, les contraintes d'optimalité peuvent être relâchées légèrement. L'idée consiste à calculer une version approchée seulement de la trajectoire optimale, par le biais de grilles définies dans l'espace de travail ***W***, dans l'espace des configurations ***C*** ou dans l'espace des états ***S***.

Le temps de calcul des méthodes à deux étapes est considérablement réduit par rapport aux autres approches et une utilisation en ligne est possible. Néanmoins, ces approches ne sont pas complètes et écartent un grand nombre de solutions potentielles lors de la sélection d'un chemin géométrique particulier.

2.2.1.4 Méthodes hybrides ou itératives

Les méthodes itératives utilisent des techniques de planification classiques basées sur l'exploration d'un arbre de recherche. La différence avec les approches classiques porte sur le fait que la modélisation de l'espace libre et son exploration ne sont pas deux tâches séparées et consécutives mais une seule répétée de manière itérative. Cette procédure permet de réduire considérablement la taille de l'espace exploré, et donc les temps de calcul, en ne modélisant que ce qui nécessite de l'être au fur et à

mesure des besoins. De plus, elle peut être interrompue à tout instant pour obtenir la trajectoire de coût minimal parmi celles déjà explorées.

En théorie, ces méthodes sont moins sensibles aux minima locaux car elles anticipent les trajectoires du robot sur des temps plus longs Jusqu'au but dans le cas idéal où elles ne sont pas interrompues). Elles permettent donc une meilleure prise en compte des contraintes de tâche, tout en assurant la réactivité du robot en environnement changeant.

En pratique, les méthodes itératives actuelles utilisent des méthodes locales d'évitement d'obstacles pour étendre l'arbre, or celles-ci sont mal adaptées aux environnements dynamiques:

Pour chaque déplacement du robot, les unes demandent des calculs coûteux. Pour compenser, la plupart utilisent une approche probabiliste pour développer l'arbre et couvrir l'espace de recherche plus rapidement et de manière plus homogène [18] par exemple).

Les autres considèrent les obstacles statiques, y compris s'ils ne le sont pas réellement. Les déplacements calculés à chaque itération doivent par conséquent être suffisamment courts pour que cette hypothèse ne mette pas le robot en danger. Or, cela a une influence directe sur la durée de validité d'une solution. De plus l'absence d'anticipation de ces méthodes conduit à des trajectoires moins naturelles.

Une approche se distingue de toutes les autres: **La bande élastique (EB)**. Son principe consiste à maintenir en permanence un passage (appelé bande) sans collision entre le robot et son but. Ce passage est représenté par un chapelet de disques chevauchant appelés des bulles. Chaque bulle représente l'espace libre local le long du passage. Son diamètre peut par conséquent s'accroître ou se réduire pour couvrir au mieux cet espace libre, en fonction des changements dans l'environnement. Leur nombre peut également varier pour éviter toute interruption de la bande. Celle-ci est soumise à deux familles de forces: des forces attractives entre bulles consécutives afin de maintenir la bande «tendue ». Et des forces répulsives avec les obstacles pour maintenir la bande éloignée de ces derniers. L'équilibre de ces forces tend vers un chemin sûr jusqu'au but. La mise à jour des forces assure l'adaptation de ce chemin en temps-réel en fonction des obstacles.

L'approche a initialement été prévue pour planifier des chemins géométriques en temps réel. Elle a été étendue ensuite à des robots non holonomes. Le principe reste le même et seule la métrique considérée change (métrique non holonome de Reed et Shepp). Pour éviter que la bande ne soit totalement déformée voire coincée (figure 2.2) en présence d'obstacle mobiles, une extension a été proposée sous le terme de bandelettes élastiques [19].

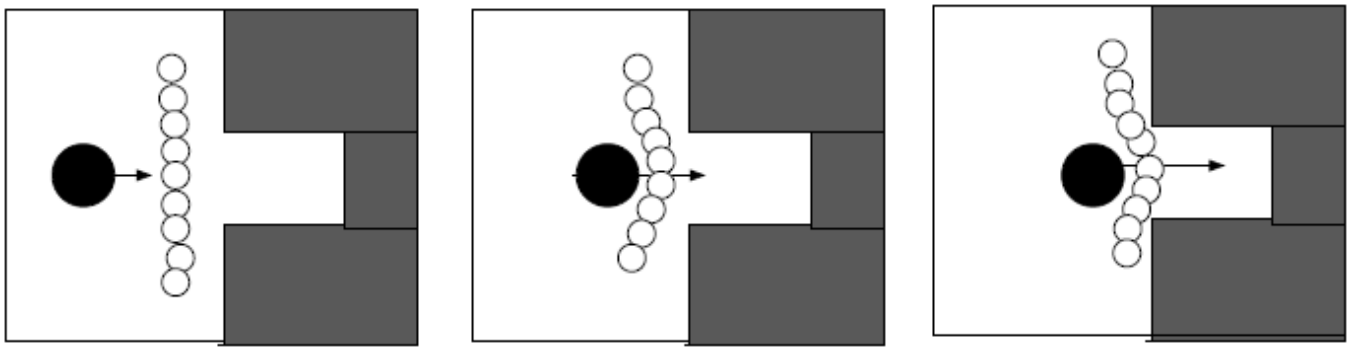


Fig.2.2 – *Bandes élastiques. Un obstacle mobile peut venir coincer la bande.*

L'idée consiste à relâcher temporairement la contrainte entre deux bulles adjacentes pour laisser passer un obstacle mobile (figure 2.3). De plus, pour limiter les cas où une replanifications complète de la bande est néanmoins nécessaire, un ensemble de bandes de secours est conservé dans [19] comme solutions alternatives rapidement disponibles.

Ceci rend la méthode des bandelettes élastiques utilisable en environnement dynamique. Néanmoins, aucune anticipation des collisions n'est faite et il nous semble qui cela fait défaut à la méthode pour garantir totalement la sécurité du robot en présence d'obstacles mobiles proches.

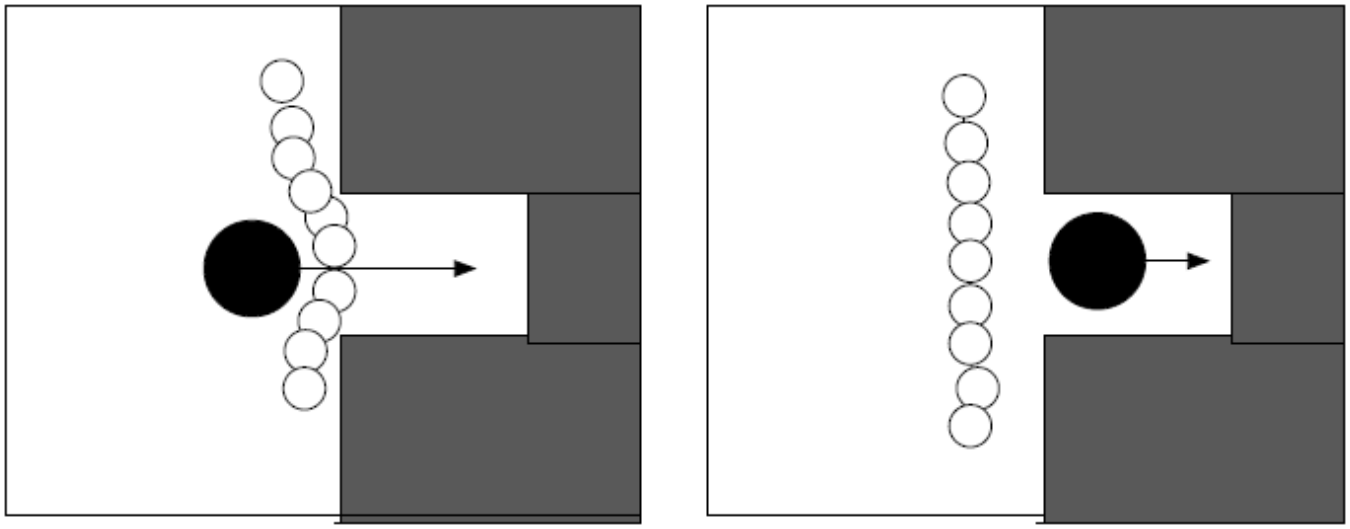


Fig.2.3 – *Bandelettes élastiques. Les bandelettes laissent passer les obstacles qui exercent Une pression trop grande sur elles.*

2.2.1.5 Conclusion sur les méthodes de calcul d'un déplacement

Les approches hybrides, et en particulier les approches itératives basées sur l'extension de méthodes locales d'évitement d'obstacles, offrent actuellement le meilleur compromis entre réactivité et prise en compte des contraintes de tâche, y compris pour des environnements dynamiques et incertains. Des méthodes locales plus adaptées à ces environnements restent néanmoins à être développées, afin de mieux anticiper les collisions. Nous apportons notre contribution à ce problème par le biais de notre approche qui combine une approche locale et une autre globale (Voir chapitre 04)

2.2.2 Calcul de la commande en présence d'incertitudes

Lorsqu'un déplacement désiré a été calculé, il reste à trouver la commande qui permettra de le réaliser: ceci est un problème de contrôle qui implique que nous possédions une modélisation du système à contrôler. Dans le cas d'un robot mobile, cela suppose un modèle du robot lui-même (contraintes mécaniques et cinématiques), ainsi que de ses interactions avec l'environnement (géométrie du robot et des obstacles pour tester les collisions, et contraintes dynamiques pour calculer les glissements par exemple). Deux types de modèles peuvent être définis (figure 2.4), selon l'utilisation souhaitée:

- Le modèle direct (encore appelé fonction de transfert), qui permet d'obtenir le déplacement du robot correspondant à l'exécution d'une commande,
- Le modèle inverse, qui permet de calculer quelle commande doit être appliquée pour obtenir un déplacement souhaité.

La principale difficulté pour obtenir ces données est due au nombre important de paramètres qui entrent en jeu et à l'impossibilité de les mesurer précisément, voire même de les identifier.

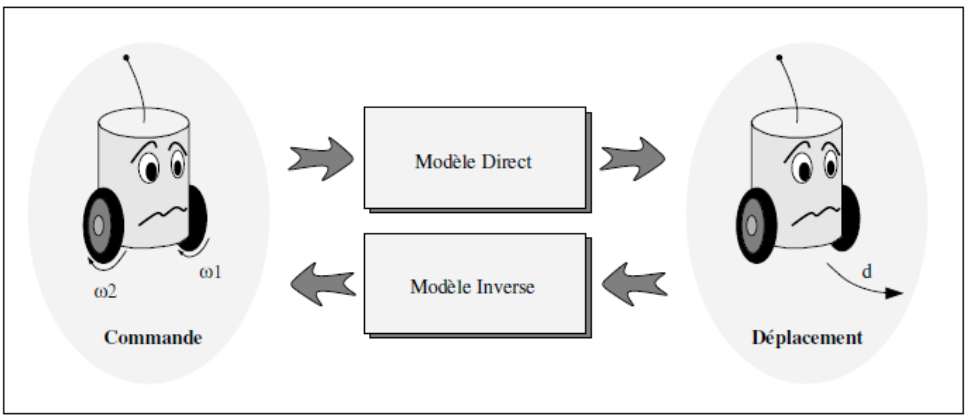


Fig.2.4 – *Modélisation d'un robot. Les modèles direct et inverse représentent les relations entre les déplacements d'un robot et les commandes envoyées à sa partie motrice "Effecteurs"*

Nous citerons l'exemple des glissements des roues d'un robot tel que le Cycab: Ils sont dépendants, entre autre, de l'architecture matérielle du robot (le Cycab possède par exemple, de par sa conception mécanique, deux centres de giration instantané théoriques distincts, impliquant des glissements inévitables), de l'états des roues (niveau d'usure, angle de braquage, vitesse angulaire, pression de gonflage), du chargement du robot (répartition du poids du robot sur chaque roue), du sol (nature état, topologie) et des conditions climatiques (présence ou non de vent, de pluie, de verglas). De plus, les capteurs permettant de mesurer ou d'estimer ces données peuvent eux aussi être défectueux, mal calibrés, bruites, ou simplement ne pas exister. Dans ces conditions, nous admettrons ici qu'un modèle n'est jamais conforme à. réalité et ne fait qu'en proposer une approximation grossière mais suffisante pour assure un contrôle satisfaisant du robot. Lorsque les paramètres de ce dernier évoluent au cours du temps, le modèle doit lui aussi évoluer: Il est dit dynamique. Le contrôle devient alors adaptatif, c'est-à-dire que ses paramètres s'adaptent pour prendre en compte 1es variations du modèle.

Le contrôle adaptatif doit ses débuts en automatique à l'armée, avec l'étude de vols supersoniques et de missiles balistiques dans les années 50. Plus tard, des approches de l'intelligence artificielle, basées sur l'observation du vivant, ont donné lieu aux contrôleurs flous (à base de la logique floue), aux contrôleurs à base de réseaux de neurones artificiels et plus récemment aux contrôleurs probabilistes.

2.2.2.1 Approche de l'automatique

Du point de vue de l'automaticien, un modèle est un système d'équations mathématiques, dont la forme et les paramètres ont été définis manuellement grâce à une étude poussée du système contrôlé. L'automaticien dispose alors d'une panoplie d'outils permettant d'apprécier la qualité d'une méthode au travers de critères tels que la stabilité et la convergence du contrôle.

Dans le cas d'un RMA, le modèle est imparfait pour les nombreuses raisons citées précédemment. Par conséquent, son utilisation pour générer une commande entraîne des écarts inévitables entre le déplacement initialement prévu et celui effectivement réalise par le robot. La mesure de ces écarts permet de corriger le contrôle. On parle alors de contrôle avec retour, ou de contrôle en boucle fermée (le contrôle en boucle ouverte étant l'application des commandes calculées sans vérification de l'effet obtenu). Deux approches sont possibles pour réduire les écarts de suivi:

- l'ajustement des paramètres du modèle afin de mieux approcher la réalité,
- l'utilisation d'une fonction correctrice appelée loi de commande.

L'identification des paramètres d'un modèle et la façon de les ajuster étant difficiles, la seconde méthode, plus extérieure au modèle, est la plus couramment employée Elle permet d'obtenir un système fiable, dont la stabilité et les performances peuvent être quantifiées avec précision. Néanmoins, elle présente les inconvénients de toute méthode d'automatique, à savoir:

- La nécessité d'effectuer une étude spécifique et complète de chaque système contrôlé.
- L'impossibilité de porter facilement une application d'un système vers un autre.

2.2.2.2 Approche de l'intelligence artificielle (IA)

L'approche de l'intelligence artificielle consiste à apprendre des relations de cause à effet d'un système en se basant uniquement sur l'observation extérieure de son comportement. Les informations sont représentées sous forme de règles de vérité « Si observation Alors action » selon l'approche de la logique floue, de probabilités « Probabilité de action Sachant observation » selon l'approche bayésienne ou encore de connexions entre neurones artificiels selon l'approche cogniticienne. Toutes sont inspirées du fonctionnement du cerveau humain ou animal.

La démarche adoptée par l'IA est typiquement celle d'un conducteur humain qui associe une orientation du volant, avec une trajectoire souhaitée. L'association se fait progressivement par apprentissage ou à l'aide de règles résultant d'une observation préalable du système, mais sans connaissance particulière des éléments mis en œuvre et de la façon dont ils interagissent.

Cela donne un attrait certain à l'approche IA, dont nous pouvons citer quelques avantages:

- Elle ne nécessite pas une étude mathématique complexe du système: Une identification des «causes » et des « effets » du système, ainsi que des relations qui les unissent est suffisante. Un apprentissage automatique de ces dernières est éventuellement possible.
- Elle facilite le portage d'une application d'un robot vers un autre de modèle semblable, ou encore permet l'adaptation des paramètres de contrôle sur un même robot, en raison du niveau élevé de généricité des informations manipulées elle permet de traiter des cas non appris en les rapprochant de cas déjà appris.

En contrepartie, I'IA ne possède pas les outils de validation de l'automatique et estimer la qualité d'un contrôleur défini selon ces principes est difficile autrement que par expérimentation. Deux problèmes fréquents sont:

- Maintenir la cohérence de la base de règles pour la logique floue ou l'approche probabiliste,
- S'assurer d'un apprentissage suffisant pour les réseaux de neurones en particulier.

2.3 Notre approche de la navigation autonome

Notre approche, qui sera détaillée dans le chapitre 4, entre dans la catégorie des approches hybrides qui consistent à combiner un planificateur global et une méthode locale pour l'exécution de mouvement. Dans le chapitre suivant nous nous intéressons aux techniques de planification de mouvement.

Chapitre 3

Planification de Mouvement

3.1 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 ce type de navigation dans son contexte général, nous reprenons ici une classification établie par Trullier et al. [43, 44], laquelle 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* [23] qui utilise 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, et a été utilisée sur divers robots [29]. Cette stratégie utilise également des actions réflexes et réalise une navigation locale qui requiert que les amers caractérisant le but soient visibles.

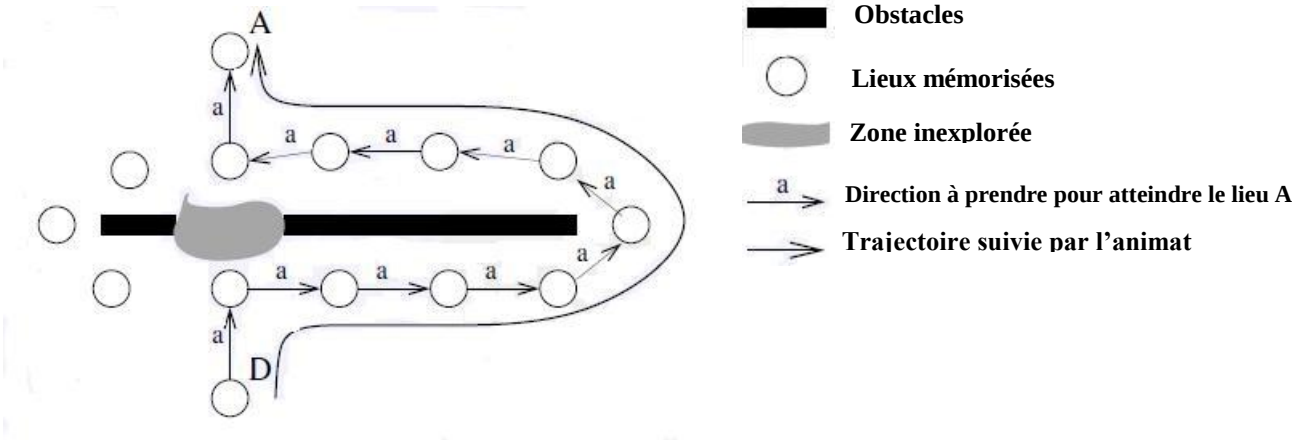


Fig.3.1 – ***Action associée à un lieu.*** *En chaque lieu, représenté par un cercle, l'action à accomplir pour rejoindre le but A est représentée par une flèche indiquant la direction à suivre à partir de ce lieu.*
Cette stratégie permet de rejoindre un but distant dans l'environnement mais repose sur 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.

– **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 ce but ou les amers qui caractérisent son emplacement sont invisibles. 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 (figure 3.1). 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.

– **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 (figure 3.2).

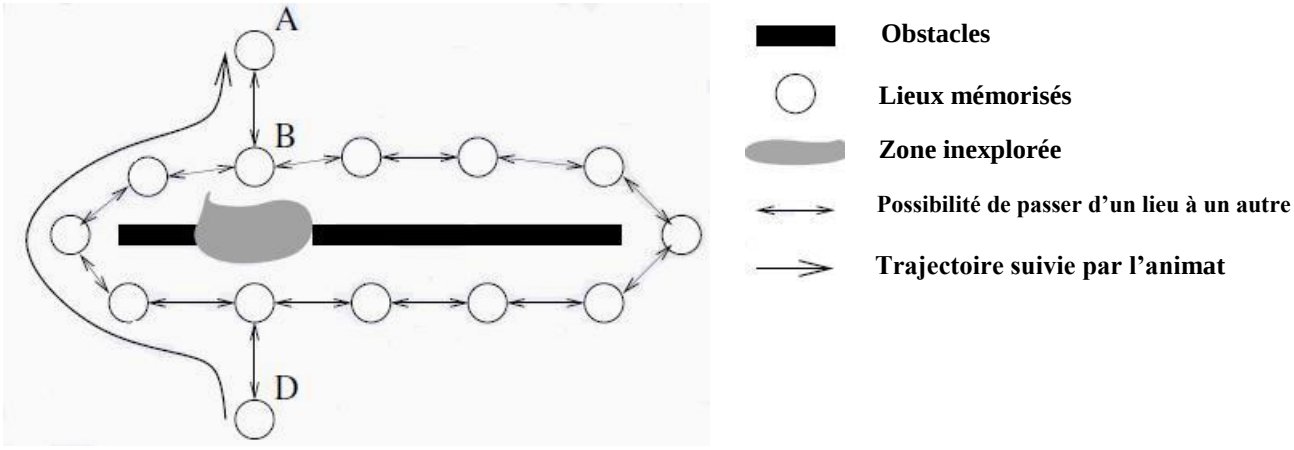


Fig.3.2 – ***Navigation topologique.*** *Cette stratégie permet de mémoriser un ensemble de lieux et les possibilités de passer de l'un à l'autre, indépendamment de tout but. Pour rejoindre un but, il faut alors une étape de planification qui permet de rechercher, parmi tous les chemins possibles, le 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 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 (figure 3.3).

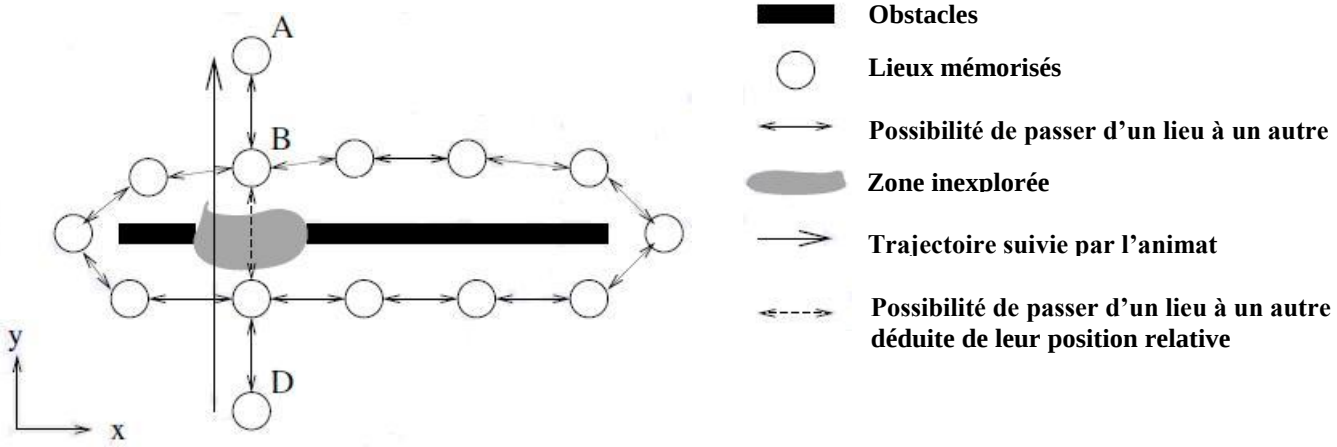


Fig.3.3 – ***Navigation métrique.*** *Cette stratégie permet de calculer le chemin le plus court entre deux lieux mémorisés, permettant même de planifier des raccourcis au sein de zones inexplorées de l'environnement. Pour cela, la carte mémorise la position métrique relative de chacun des lieux visités 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.*

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*. 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, ils 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 des sections suivantes. 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 difficiles 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, par exemple celui qui consiste à nommer les rues, pour faciliter les processus de navigation. 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 sont a priori pas destinées.

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 neurologique dans certaines parties de leur cerveau, notamment dans l'hippocampe. Cela montre que des cartes sont utilisées 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.

La partie suivante présente la navigation par carte qui permet 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 décomposent en trois processus : *cartographie, localisation et planification*,

3.2 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 [37] : 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 prévoir les 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'oeuf 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. Ainsi, pour un robot complètement autonome, il est impossible de ne pas traiter ces deux problèmes simultanément.

Dans le cadre où l'on autorise un opérateur humain à intervenir dans le processus, il est évidemment possible de découpler ces deux phases. Dans les applications réelles, 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 est 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.

3.3 Quelques hypothèses de travail

3.3.1 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 relativement 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.

3.3.2 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 environnements 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 vivent, 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 (l'architecture), 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 très fortement bruités.

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 capables 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.

3.4 Représentations de l'environnement

Les deux utilisations possibles des perceptions (avec et sans modèle métrique) trouvent en parallèle deux types de représentations de l'environnement.

Lorsque aucun modèle métrique n'est utilisé pour les capteurs, les données sont en général mémorisées dans une carte *topologique* [40] (figure 3.4). 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, pour leur part, 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* (figure 3.4) 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.

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.

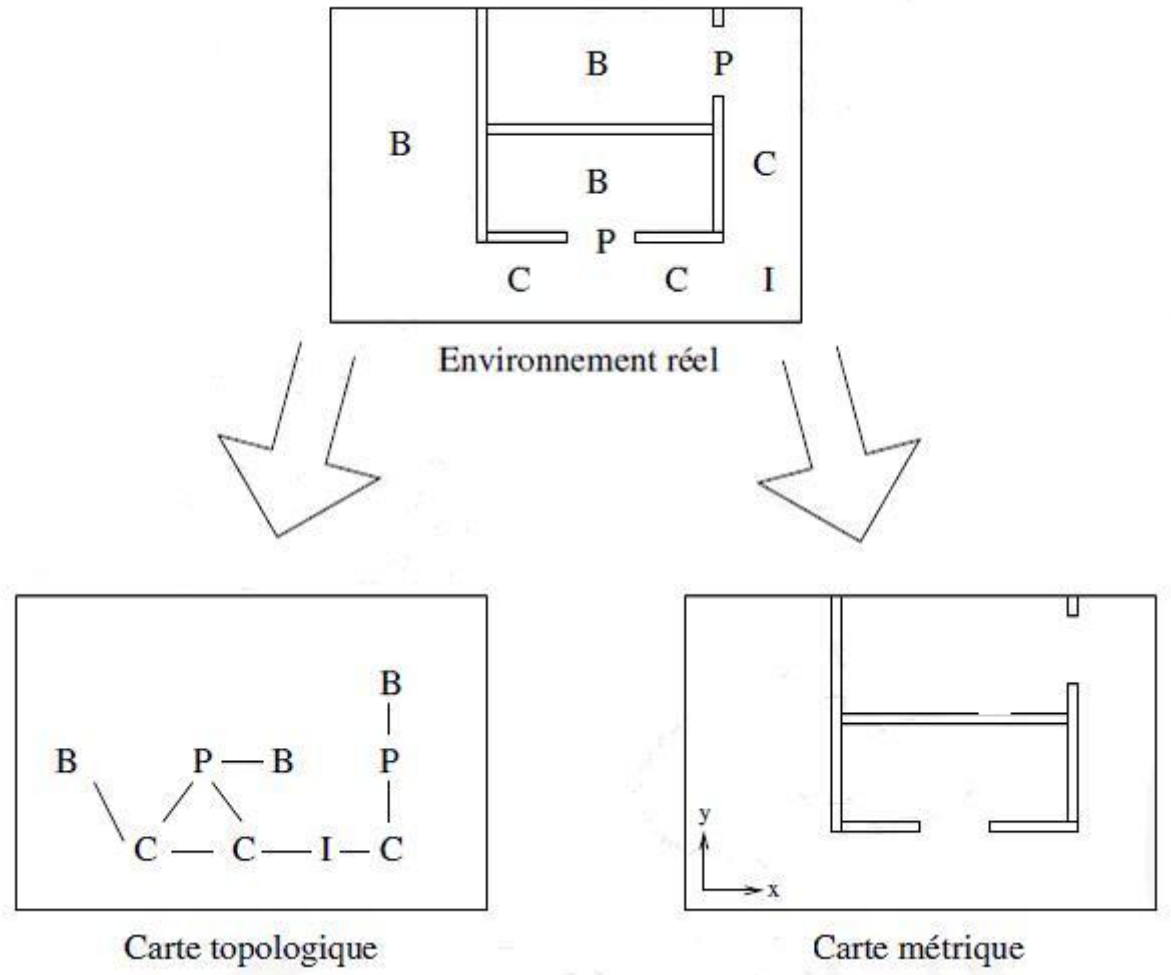


Fig.3.4 – *Les cartes utilisées en robotique peuvent être de deux types : les cartes topologiques, d'une part, qui mémorisent un ensemble de lieux, ainsi que les manières de se déplacer de l'un à l'autre (dans cet exemple, B=Bureau, C=Couloir, P=Porte et I=Intersection) et les cartes métriques, d'autre part, qui mémorisent un ensemble d'objets perçus (des murs dans cet exemple) avec une position dans un cadre de référence global.*

3.5 Cartes topologiques

3.5.1 Description

Les cartes topologiques permettent de représenter l'environnement du robot sous forme de graphe. Les noeuds du graphe correspondent à des lieux, c'est-à-dire des positions que le robot peut atteindre. Les arêtes liant les noeuds 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 ont été mises en oeuvre 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 noeuds, ou des informations sur les actions à effectuer pour parcourir cette arête.

3.5.2 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. 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. 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.

3.5.3 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 noeuds 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.

3.5.4 Implantation

Définition des noeuds

Le choix de ce que vont représenter les noeuds 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. De plus, 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 des choix des opérateurs humains ou être complètement autonome.

Noeuds 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 [26, 33].

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.

Noeuds 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. 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 noeud dans la carte, mais ce noeud 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, ou le numéro qui est inscrit dessus.

Cette méthode de définition des noeuds a été proposée par Kuipers et Byun [32] sous le nom de *distinctive places*, puis utilisée sous une forme différente par Engelson et McDermott [27] et par Kortenkamp et Weymouth [31] sous le nom de *gateways*.

Noeuds définis de manière non supervisée:

La troisième méthode pour définir les noeuds d'une carte topologique consiste à les définir comme des zones où les perceptions sont approximativement constantes. Cela est obtenu en général par la

catégorisation non supervisée des perceptions [37, 45]. 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 noeuds de la carte. Le noeud 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 noeud. A ce titre, elle est utilisée dans tous les systèmes de navigation qui s'inspirent des comportements de navigation des animaux [21].

Pour mettre en oeuvre 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 noeud reconnu.

Lorsque la différence est suffisamment importante, on considère qu'un nouveau lieu a été atteint. Cette méthode est utilisée par certains modèles [39], 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 noeud a été atteint lorsque la distance parcourue depuis la dernière reconnaissance est assez grande [21]. Ces deux approches conduisent ainsi à des discrétisations différentes de l'environnement.

Définition des arêtes

Les arêtes reliant les noeuds permettent de mémoriser des données sur les relations de voisinage entre lieux représentés par les noeuds. 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 noeuds 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 [45], 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 [27, 39, 46]. 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 noeud 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* [27].

Association de position aux noeuds

Dans le but d'intégrer les données proprioceptives à une carte topologique, il est également possible d'associer une position à chacun des noeuds. 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 noeud 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 [21, 22]. Toutefois, certains modèles utilisent également des liens pour mémoriser l'information d'adjacence. Comme l'information de position de chaque noeud 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.

3.6 Cartes métriques

3.6.1 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.

3.6.2 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 noeuds présents dans la carte et sont donc relativement imprécises. Une telle représentation, dans laquelle chaque noeud 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 de 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.

3.6.3 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'une 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, 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 nombreux modèles recourent d'ailleurs à l'extraction d'une carte topologique depuis la carte métrique pour réaliser cette opération de planification [34].

3.6.4 Implantation

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. 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. 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 [30, 42]. 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, par exemple l'angle d'un obstacle, perçu grâce à un télémètre laser.

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 des polygones sont très souvent utilisés. Ces objets sont extraits d'ensemble de points perçus par des capteurs à ultrasons ou des télémètres laser. Des cylindres et des plans, détectés par des capteurs à ultrasons sont aussi utilisés, ainsi que des structures de plus haut niveau, comme des plans en trois dimensions, détectés par stéréo-vision.

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 [25]. Toutefois, d'autres méthodes peuvent être utilisées, par exemple des intervalles [27] ou des ensembles flous [28].

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 [36]. 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*. Dans ce modèle, l'environnement est entièrement discrétisé suivant une grille régulière avec une résolution spatiale très fine. 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 soumise à une forte incertitude.

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 [20] ou à des discrétisations hiérarchiques. De telles discrétisations permettent de s'adapter à la complexité de l'environnement, en représentant de manière grossière les grands espaces libres et plus finement les contours des obstacles.

3.7 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 cette section quelques méthodes très 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 [35].

3.7.1 Espace des configurations

Dans le cadre de notre travail, nous considérons des robots capables de se déplacer dans un espace à 2 dimensions dont les commandes influent directement 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 déplacement du robot. 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.

Dans le cas 2D pour un robot holonome, l'espace des configurations que nous utiliserons sera simplement l'espace de travail auquel nous enlèverons toutes les positions conduisant à percuter un obstacle, c'est à dire les obstacles eux même, plus une marge de sécurité autour des obstacles correspondant au rayon du robot. En pratique nous confondrons souvent l'espace des déplacements et l'espace des configurations.

3.7.2 Discrétisation de l'espace de recherche

Les cartes topologiques fournissent directement une discrétisation de l'environnement réel utilisable par les techniques de planification. 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 [20]. D'autres modèles font appel à des décompositions de l'espace libre spécifiques à la planification. Cependant, différentes techniques, tels les champs de potentiel analogues à ceux décrits dans le chapitre II, permettent de calculer des chemins directement dans le domaine continu, sans phase préalable de discrétisation.

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 reproduire la topologie de l'espace libre (figure 3.5). Les cellules obtenues sont alors utilisées de manière similaire aux noeuds des cartes topologiques dans le processus de planification, les cellules adjacentes étant considérées comme reliées par une arête.

Les méthodes de la seconde catégorie font appel au pré-calcul de chemins entre des points répartis dans l'environnement (figure 3.6). Les points seront utilisés comme les nœuds d'une carte topologique, tandis que les chemins pré-calculés reliant les noeuds seront utilisés comme les arêtes de cette carte.

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 3.7).

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.

3.7.3 Recherche de chemin

A partir d'une carte métrique discrétisée ou d'une carte topologique, 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.

3.7.3.1 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**, ou l'une de ses nombreuses variantes, peuvent être utilisés pour calculer ce type de plan. 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 noeud qui ne fait pas partie du chemin planifié.

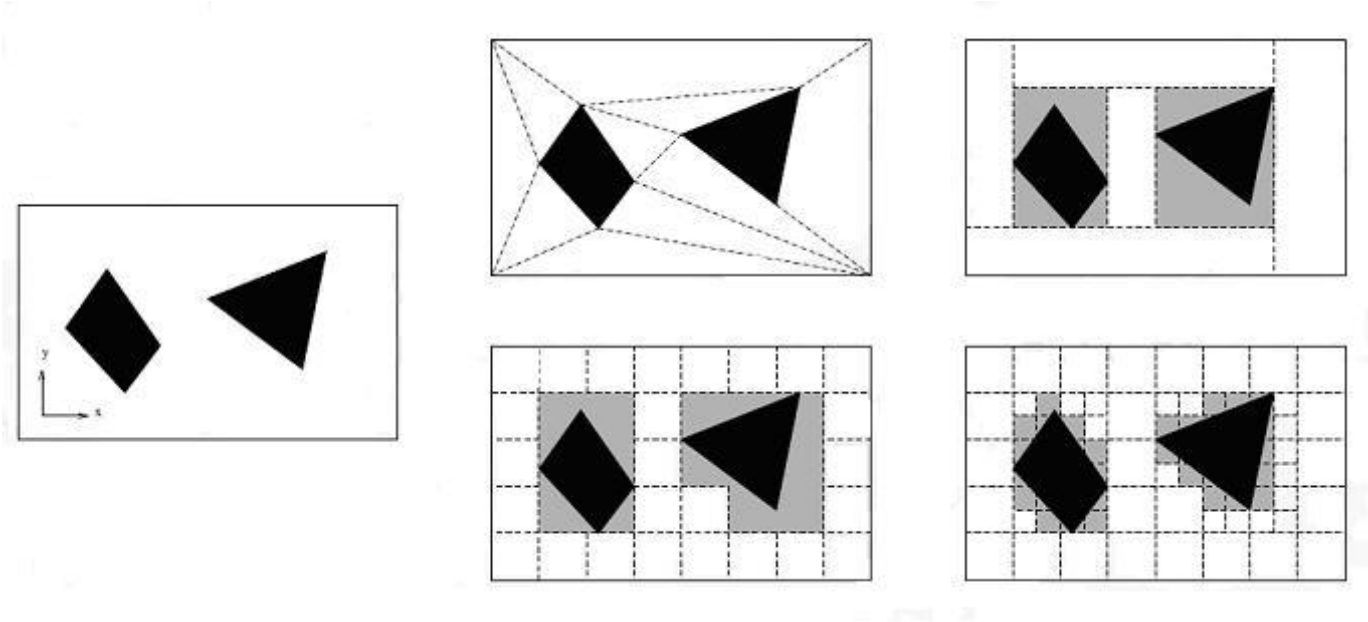


Fig.3.5 – *Exemples de décompositions en cellules de l'espace libre dans les cartes métriques. La décomposition exacte permet de représenter l'ensemble de l'espace libre, à l'aide de cellules de formes irrégulières. La décomposition en cellules rectangulaires ne représente qu'un sous-ensemble de l'espace libre suffisant pour la planification. Ces cellules peuvent être de taille régulière ou non. 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.*

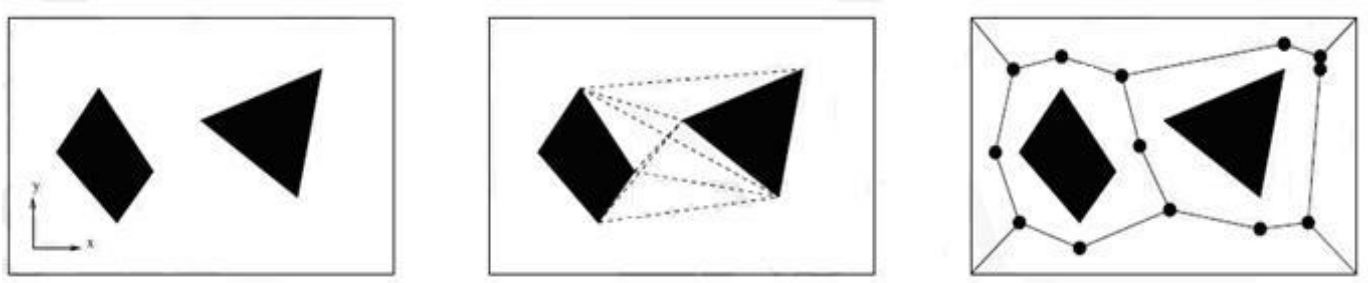


Fig.3.6 – *Exemples de décompositions en chemins pré-calculés dans les cartes métriques. Ces chemins relient des points particuliers utiles pour la navigation et répartis dans l'environnement. Différents choix de points sont possibles. Le graphe de visibilité utilise les angles d'obstacles qui sont les 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.*

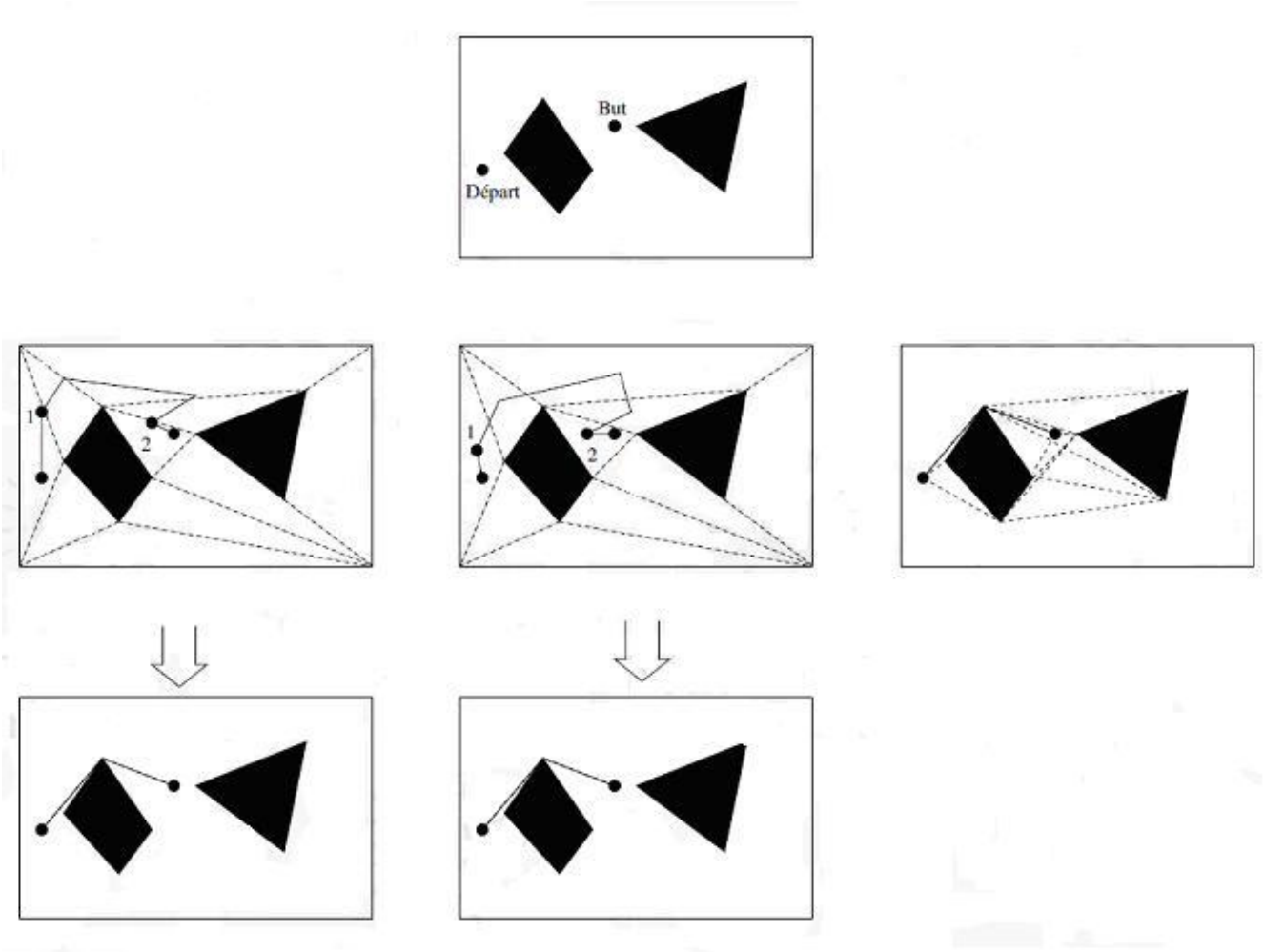


Fig.3.7 – *Exemple de planification de chemin dans une carte métrique. Deux portions de trajectoire sont calculées pour relier le point de départ et le but à des points de l'espace discrétisé (points 1 et 2 dans cet exemple). Un chemin est ensuite calculé dans l'espace discrétisé entre ces deux points. La trajectoire résultante peut ensuite être optimisée pour supprimer les effets de la discrétisation.*

La 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, qui associe à chacune des positions possibles du robot au sein de la carte l'action qu'il doit effectuer pour atteindre son but, peut être utilisé. Ce type de plan est appelé *politique* ou *plan universel*. Le résultat est alors une stratégie de déplacement similaire à la stratégie *d'action associée à un lieu* mentionnée dans la section 3.1. 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.

3.7.3.2 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* ou *wavefront propagation* [38]. 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 noeuds et à chaque lien entre les noeuds. Les coûts associés aux liens traduisent en général la distance entre noeuds, tandis que les coûts associés aux noeuds permettent de marquer des zones à éviter ou à favoriser pour les trajectoires du robot.

Algorithme 3.1: Algorithme de Dijkstra

1: S = ensemble vide ; R = ensemble de tous les noeuds ;
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, qui, partant du but fait à chaque étape la mise à jour du noeuds suivant de potentiel le plus faible. Cet algorithme suppose que tous les coûts utilisés sont positifs ou nuls.

Une seconde méthode pour calculer ces potentiels lorsque certains coûts sont négatifs est l'utilisation de l'algorithme de programmation dynamique de Bellman-Ford, aussi connu sous le nom de *value itération*, notamment en apprentissage par renforcement [40]. Une itération de cet algorithme utilise quasiment la même méthode de mise à jour que l'algorithme de Dijkstra mais sans utiliser le même ordre dans les mises à jour des noeuds. En présence de coûts négatifs, cette itération doit être répétée autant de fois qu'il y a de noeuds 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.

3.7.3.3 Calcul d'un chemin

Si l'on ne souhaite que calculer un chemin, il est par exemple possible d'employer l'algorithme *A**. Cet algorithme utilise exactement le même mécanisme que l'algorithme de Dijkstra, mais utilise une heuristique pour choisir le noeud suivant à explorer au lieu d'explorer systématiquement les noeuds voisins des noeuds déjà planifiés. Cette heuristique doit fournir une évaluation la plus rapide possible de la distance entre le noeud 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).

3.7.4 Exemples de politiques

La figure 3.8 donne un exemple de champ de potentiel obtenu en utilisant un poids nul pour les noeuds et la distance entre noeuds 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 3.9 montre le poids associé aux noeuds et les trajectoires obtenues.

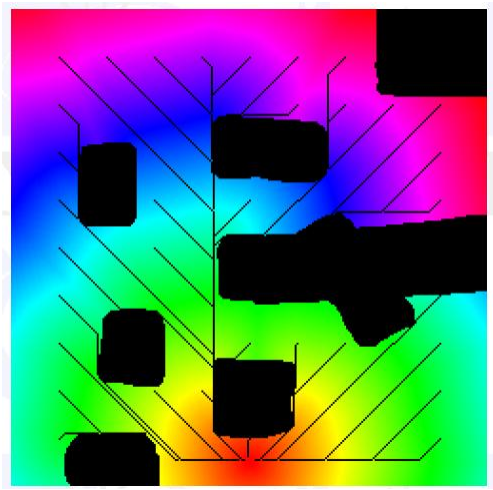


Fig.3.8 – *Potentiel et exemple de trajectoires avec des poids de liens égaux à la distance entre noeuds.*

3.7.5 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.

Différentes méthodes permettent de prendre ces probabilités en compte. Une première méthode consiste à utiliser une méthode de vote [24]. Pour cela, une action est simplement associée à chacun des noeuds 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 noeuds 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 (figure 3.10).

Algorithme 3.2: Algorithme de Bellman-Ford

1: {%} Retourne FAUX si il y a un cycle de coût total négatif

2: **for all** Noeuds *i* **do**

3: d(i) = + ∞;

4: **end for**

5: d(but) = 0 ;

6: **for** i=1 jusqu'à Nombre de sommets **do**

7: **for all** liens (u,v) du graphe **do**

8: **if** d(v) > d(u) + w(u,v) + w(v) **then**

9: d(v) = d(u) + w(u,v) + w(v)

10: **end if**

11: **end for**

12: **end for**

13: **for all** liens (u,v) du graphe **do**

14: **if** d(v) > d(u) + w(u,v) + w(v) **then**

15: return FAUX

16: **end if**

17: **end for**

18: return VRAI

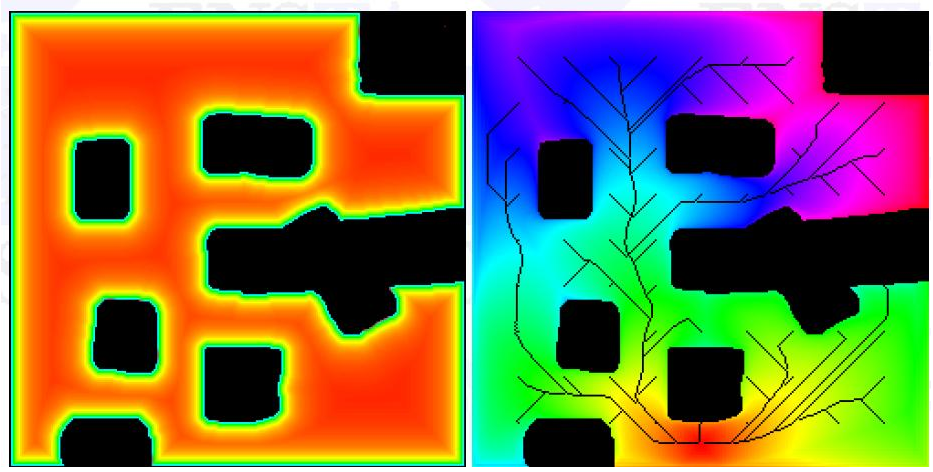


Fig.3.9 – *Poids associé aux noeuds et exemples de trajectoires obtenues en utilisant ces poids.*

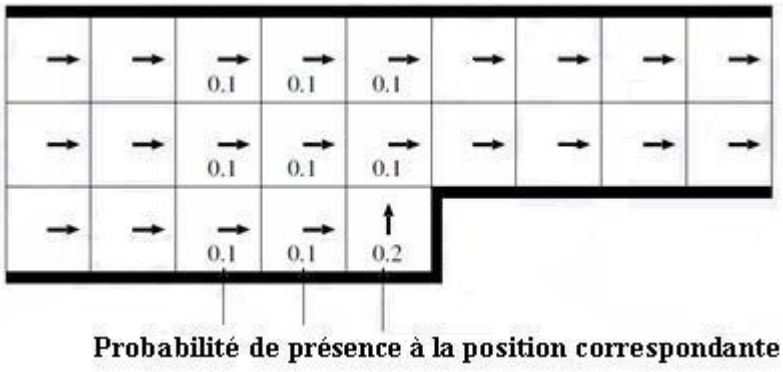


Fig.3.10 – *Exemple de l'intérêt d'une procédure de vote dans le cas où la situation du robot est incertaine. Si l'on choisit l'action associée à la position la plus probable, le robot ira en haut, ce qui correspondra à l'action correcte avec une probabilité* 0,2*. En utilisant une méthode de vote, l'action choisie sera d'aller à droite, ce qui sera correct avec une probabilité* 0,8*.*

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 [24, 41].

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).

3.8 Conclusion

Les algorithmes de planification de mouvements considèrent dans la plupart des cas des environnements statiques parfaitement connus. Dans de nombreuses situations, des modifications de la scène nécessitent pourtant d'être prises en compte dans le calcul du mouvement. On parle de façon générale de problèmes à environnement dynamique. Les travaux concernant les environnements dynamiques s'articulent plus particulièrement autour de trois grandes problématiques. La première concerne les situations pour lesquelles l'évolution dans le temps de la position des obstacles mobiles est supposée connue. Dans ce cas, la prise en compte de ces obstacles peut être faite dés la phase de planification. La deuxième problématique concerne le développement de méthodes de planification plus performantes pour rendre compte des différents changements susceptibles d'avoir lieu dans l'environnement. La dernière problématique concerne directement le sujet de ce mémoire. Elle porte sur la navigation des robots, qui nécessite de conjuguer des méthodes de planification avec des techniques d'exécution de trajectoire.

Chapitre 4

Approche Hybride pour l'Evitement d'Obstacles Dynamiques

4.1 Introduction

De plus en plus la recherche dans le domaine robotique à intérêts industriels se concentre sur l'amélioration du degré d'autonomie du système robotique. Les robots sont conçus pour opérer dans différentes conditions, en prenant en compte des missions à périodes de temps plus long sans intervention humaine. Il s'agit notamment de tâches spécifiques dans un environnement dangereux ou hostile. Les systèmes de navigation autonome sont utilisés dans des applications comme les robots de service, de surveillance ou d'exploration où le robot se déplace et effectue la tâche principale à la fois.

Un des aspects clés de ces robots est la mobilité, car elle constitue la base sur laquelle on peut intégrer plusieurs sous-systèmes avec des fonctionnalités différentes. Cependant, la performance du système de mouvement affecte fortement les performances des tâches. Des problèmes spécifiques se posent dans les applications qui peuvent avoir des conséquences fatales (par exemple, des robots qui transportent un matériel dangereux).

La mobilité est étroitement liée à la nature de l'environnement. Dans de nombreuses applications, l'environnement ne peut être spécifié à priori avec une carte et peut être dynamique. Dans ces circonstances, des capteurs peuvent recueillir des informations sur l'environnement et adaptent les mouvements du robot à toute nouvelle situation imprévue ou un événement.

Les systèmes de mouvement à base de capteurs semblent être le choix naturel, mais la plupart d'entre eux ne permettent pas une navigation robuste et fiable dans des environnements très compliqués. Cela concerne généralement des endroits avec peu d'espace pour manœuvrer, des environnements hautement dynamiques ou qui conduisent à des situations de blocage. Un exemple est un bureau, où le robot se déplace parmi les chaises, les tables et étagères (avec des positions inconnues) et les humains (ce qui rend l'environnement hautement dynamique). La manoeuvre est également extrêmement difficile dans certaines circonstances quand il y a peu d'espace pour manoeuvrer. Notre travail s'intéresse au contrôle du mouvement d'un robot en vertu de ces conditions.

Nous avons conçu un système à base d'un capteur laser avec trois modules qui fonctionnent ensemble pour mener à bien la tâche de mouvement.

Notre conception est basée sur certaines exigences que nous avons identifiées pour conduire le robot dans des environnements dynamiques et gênants. Le système diffère des travaux antérieurs dans le choix et l'implémentation des modules et dans l'architecture d'intégration. Notre contribution concerne les aspects fonctionnels et informatiques de la conception des modules et de leur intégration, et la validation expérimentale (par simulation) dans des environnements dynamiques.

Notre système à base d'un capteur laser a été développé pour déplacer le robot aux positions souhaitées sans collisions. Cette fonctionnalité est uniquement un sous-ensemble du problème de navigation. D'autres aspects impliquent la perception, la modélisation et le contrôle. Ils ne seront pas abordés dans cette thèse, mais ils sont essentiels pour construire un système complet de mouvement autonome.

Le chapitre est organisé comme suit: d'abord nous discuterons des travaux liés (section 2), et les exigences d'un système de contrôle de mouvement à base de capteurs (Section 3). Ensuite, nous donnerons un aperçu du système (section 4). Les modules et leur intégration (section. 5).

4.2 Travaux dans le domaine

Le problème de planification de mouvement dans des environnements dynamiques englobe des questions telles que la représentation de l'environnement (construction de modèle), la délibération globale et la réactivité.

La planification sans tenir compte de l'exécution est une restriction à un petit domaine du problème. C'est parce qu'il devient difficile de considérer tous les imprévus et ce n'est pas pratique de formuler des plans qui ne reflètent pas un environnement changeant. D'autre part, les systèmes réactifs limitent leur application à la perception-action, la flexibilité et la robustesse de mouvement. Le problème global ne peut être résolu individuellement par ces systèmes puisqu'ils ont besoin de modèles plus vastes de connaissances et d'une certaine manière à intégrer la mémoire. L'intérêt est porté sur la synthèse d'un mode de contrôle qui intègre les deux méthodes à la fois et non pas sur l'extension de deux environnements séparés, [47].

Les systèmes hybrides tentent de combiner les deux paradigmes en utilisant l'intelligence artificielle pour représenter et utiliser les connaissances, avec la meilleure réactivité, robustesse, adaptation et flexibilité. Fondamentalement, ces systèmes doivent combiner un planificateur (délibération) et un reactor (exécution).

Les principales différences entre eux sont les suivantes:

(I) L'interaction entre le planificateur et le reactor (i.e. comment la méthode réactive utilise les informations disponibles du planificateur).

(II) Les techniques utilisées pour mettre en œuvre chaque module.

Une manière de préciser l'interaction entre la délibération et la réaction est de considérer la planification comme un composant qui fait la composition des différents comportements lors de l'exécution, [48]. Par exemple, ces comportements peuvent être mis en œuvre avec des champs de potentiel, [9], de sorte que la modification de leur poids change le comportement global du système.

Une autre possibilité est d'utiliser la planification pour orienter le contrôle réactif, [49], ou comme un système qui adapte les paramètres de la composante réactive à base de l'évolution de l'environnement, [50]. Dans tous les cas, la planification joue un rôle tactique, alors que le reactor dispose d'un degré de liberté d'exécution. L'avantage de cette configuration planner-reactor est qu'elle combine la composante délibérative [le plan est toujours disponible dans l'exécution et s'améliore dans le temps] et la composante réactive (exécuteur du mouvement). Une perspective sur les architectures hybrides est donnée dans [51].

Dans le contexte de mouvement, une stratégie commune est de calculer une trajectoire et de l'utiliser pour orienter le module réactif, [52].

Ces techniques nécessitent de grandes ressources de calcul, à cause de l'utilisation de planificateurs globaux (ils trouvent toujours un chemin s'il existe). D'autres techniques calculent un chemin qui se déforme en exécution en se basant sur l'évolution de l'environnement [dans l'espace de travail, [19], ou dans l'espace de configuration, [53]].

Néanmoins, ces méthodes doivent faire une replanification lorsque le chemin est invalidé ou quand le robot se déplace très loin de la configuration initiale en raison des obstacles inattendus. Alternativement, Ulrich et Borenstein (1998) [11] présente une stratégie de création d'arbres de chemins obtenus par l'exécution de l'algorithme réactive à quelques longueurs d'avance sur l'exécution. Ce système obtient de bons résultats en plateformes ayant peu de ressources de calcul, mais ne garantit pas une convergence vers la cible. Une autre possibilité consiste à calculer un canal d'espace libre qui contient des ensembles de chemins, en laissant le choix à l'exécution, [54]. Ces questions sont étroitement liées au choix et à l'implémentation des techniques pour chaque module.

Toutes les stratégies citées antérieurement, utilisent le planificateur global pour obtenir un chemin afin d'orienter le contrôle réactif. Le planificateur est habituellement une technique numérique efficace exécutée en temps réel.

Un autre module clé est la méthode réactive elle-même. Certaines techniques sont basées sur des méthodes potentielles, mais elles ont des limites. D'autres techniques calculent un ensemble de commandes de mouvement intermédiaires et de choisir une qui sera la prochaine à exécuter.

Les commandes sont des directions du mouvement, ensembles de vitesses, ou ensembles de trajectoires. Toutefois, ces méthodes réactives sont d'une utilité limitée quand le scénario rend difficile la manœuvre du robot (généralement avec une densité d'obstacles élevée). Le travail de Minguez et Montano (2004) [55] a identifié certaines conséquences comme les situations locales de blocage, les

mouvements irrégulières et oscillants, ou l'impossibilité de conduire le robot dans des zones à forte densité d'obstacles ou lorsque il est loin de la direction du but. Ces comportements acquièrent une plus grande pertinence dans le développement d'applications robustes pour naviguer indépendamment de la difficulté de l'environnement.

Les systèmes de navigation qui reposent sur des méthodes réactives héritent aussi de ces inconvénients, limitant leur utilisation dans des applications pratiques. Enfin, dans ce type d'architecture, les modèles sont généralement construits pour servir de base pour le planificateur et fournir une mémoire de courte durée pour le comportement réactif. Dans les environnements d'intérieur, les grilles d'occupation sont généralement utilisées avec des ultrasons, et le laser.

Dans ce chapitre, nous décrivons un système hybride avec une configuration Planner – Reactor synchrone et hétérogène, où les deux modules utilisent le modèle construit dans l'exécution et de mener à bien la tâche de mouvement. Notre principale contribution est le choix et la conception des modules et leur intégration. En conséquence, le nouveau système peut déplacer le robot dans des environnements très difficiles, là où d'autres systèmes peuvent trouver des difficultés.

4.3 Exigences pour un système de control à base de capteur

La version de base de ces systèmes déplace le robot entre positions sans collision. L'opération est régie par un processus *perception - action* répété à une fréquence élevée. Les capteurs recueillent des informations sur l'environnement (obstacles) et le robot, qui servent pour calculer le mouvement. Le robot exécute le mouvement et le processus redémarre. Le résultat est une séquence de mouvements en ligne qui conduit le robot jusqu'au but sans collisions. Dans cette section, nous précisons certaines exigences générales relatives à ces systèmes [56]:

(1) L'intégration de l'information: les mesures successives sensorielles doivent être stockées ou intégrées pour construire une représentation de l'environnement. Cela est nécessaire pour éviter les obstacles qui ne sont pas perçus au moment (contraintes de visibilité du capteur), et d'accroître la portée de l'information utilisée (augmentation de la domination spatiale). En outre, les changements dans les scénarios dynamiques doivent être rapidement intégrés dans le modèle. Sinon, le robot va éviter les zones perçues comme un espace libre ou il ne va pas éviter des obstacles déjà perçus, puisque dans les deux cas, l'information n'a pas pu être stockée dans le modèle.

(2) L'évitement des situations de piège et des comportements cycliques: le système doit être muni d'une stratégie visant à éviter ces situations. Plusieurs configurations différentes d'obstacles peuvent piéger le robot (l'exemple le plus typique est un obstacle de la forme U-shape ou les end-zones) ou créer un mouvement cyclique (par exemple, Les distributions symétriques d'obstacles). Le robot ne pourra jamais atteindre l'emplacement final dans ces circonstances.

(3) Génération d'un mouvement robuste: le mouvement final doit être calculé par une méthode réactive robuste. Cet algorithme doit être capable d'éviter les collisions indépendamment de la difficulté de l'environnement. Comme règle générale, les scénarios qui posent plus de problèmes ont une large densité d'obstacles où la manœuvre est difficile et critique.

(4) L'intégration de fonctionnalités: Toutes les fonctionnalités doivent être intégrées au sein d'une architecture pour la spécification, la coordination et la non détection et la récupération. L'intégration doit être munie d'un cycle *perception - action* à une fréquence élevée, ce cycle fixe la réactivité du système pour les changements imprévisibles (détectés par les capteurs). En outre, cela favorise la portabilité entre les différentes plates-formes et capteurs (les interactions entre les modules ne sont pas conçus à partir de zéro lorsque les modules sont remplacés).

Dans notre travail, nous nous efforçons de rendre notre système capable de remplir ces exigences.

4.4 Système hybride

Nous fournissons une vue générale de notre système hybride qui comporte trois modules et une architecture pour la supervision. Les fonctionnalités des modules sont la construction du modèle, la planification et la navigation réactive:

• **Module de modélisation:** il construit un modèle de l'environnement (intégration de l'information sensorielle). Nous avons utilisé une grille d'occupation binaire dont les cellules sont mises à jour chaque fois qu'une mesure sensorielle est disponible. La grille a une taille limitée (représentant une partie fixe de l'espace), et dont l'emplacement est recalculée en permanence pour inclure la position du robot.

• **Module de planification:** il extrait la connectivité d'espace libre (utilisée pour éviter les situations de pièges et les mouvements cycliques). Nous avons utilisé la fonction de navigation (*Navigation Function 1* ou NF1 en bref) basée sur le principe de propagation de vagues en espace libre *(wavefront).* Le planificateur est libre de minima potentiels, il peut travailler sur une grille (représentation existante), et le plan peut être efficacement exécuté en temps réel (au moins une fois dans chaque cycle de la perception).

• **Module de navigation réactive:** il calcule un mouvement libre en évitant la collision avec les obstacles. La méthode utilisée est la *Vector Fields Histogramm* (VFH en bref, [11]), qui est basée sur la sélection d'une situation de navigation à chaque moment et l'application de la loi de mouvement associée. Cette méthode est très efficace et robuste dans les environnements avec peu d'espace pour manœuvrer.

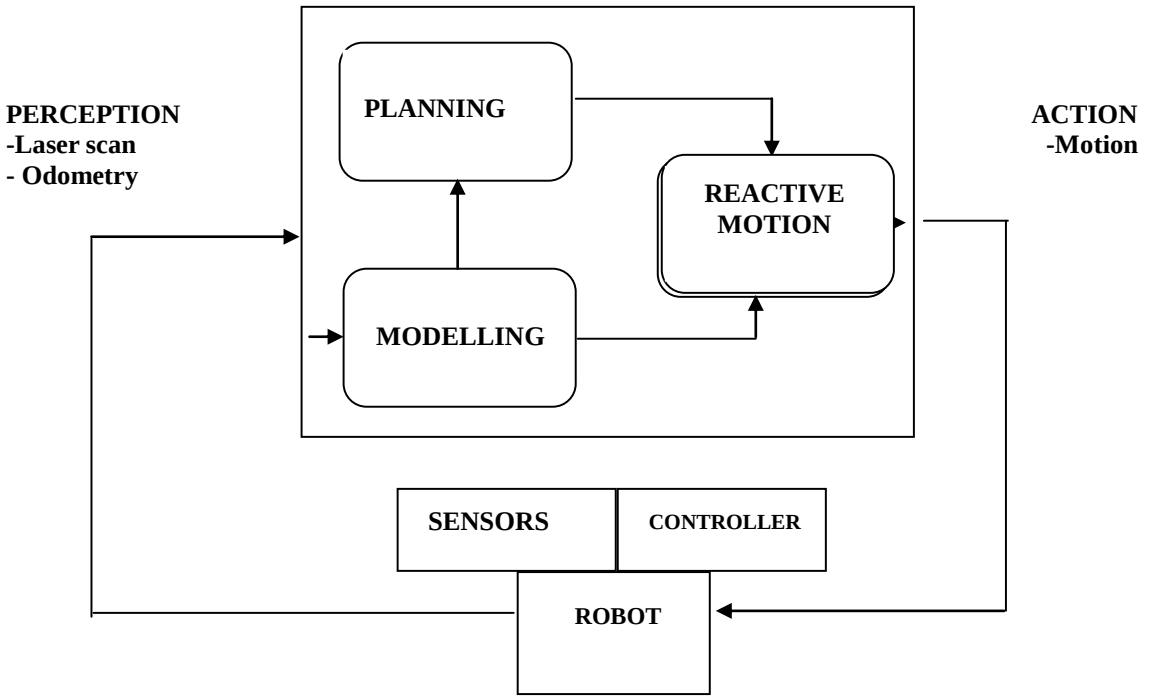


Fig.4.1 – *Cette figure illustre le cycle perception - action du système de contrôle de mouvement à base de capteurs. Les perceptions sont des mesures du capteur Laser et l'odométrie du robot. L'action est le calcul de commandes pour un mouvement sans collision. Le système comporte trois modules qui coopèrent pour mener à bien la tâche de navigation: la construction du modèle, le planificateur global et la méthode de navigation réactive.*

• **Architecture d'intégration:** elle intègre les modules suivant une configuration synchrone Planner - Reactor, [50], où les deux parties utilisent le modèle construit pendant l'exécution. La synchronisation entre modules permet d'éviter les problèmes d'incohérences dans le temps.

Le système fonctionne comme suit (Figure 4.1): étant donné un Laser scan et de l'odométrie du robot, le constructeur du modèle intègre cette information dans le modèle existant. Ensuite, l'information sur les obstacles et l'espace libre dans la grille d'occupation est utilisée par le module de planification pour calculer le chemin jusqu'au but. Enfin, le module réactif utilise l'information sur les obstacles dans la grille et l'information du planificateur global pour générer un mouvement (conduire le robot vers sa destination sans collisions). Le mouvement est exécuté par le contrôleur de robot et le processus redémarre. Il est important de souligner que les trois modules travaillent de manière synchrone dans le cycle perception - action. Cela renforce l'importance du choix et les aspects informatiques des techniques utilisées dans chaque module. Dans la section suivante, nous décrivons la conception des trois modules et l'architecture d'intégration.

4.5 Conception et intégration des fonctionnalités

Nous présentons ici la conception des modules dans le système. Nous discutons du module de modélisation (sous-section 5.1), le module de planification (sous-section 5.2), la méthode de navigation (sous-section 5.3) et l'architecture d'intégration (sous-section 5.4).

4.5.1 Module de construction du modèle

Ce module intègre les mesures successives sensorielles pour créer un modèle local de l'environnement. Nous avons choisi d'utiliser une grille d'occupation binaire dont les cellules sont *occupées*, *libres* ou *inconnues*.

Nous n'avons pas utilisé les facteurs de traversabilité ou de l'incertitude car le laser a une précision élevée dans les environnements d'intérieur. La grille a une taille fixe qui représente une partie limitée de l'espace de travail (une taille suffisante pour représenter la portion de l'espace nécessaire pour résoudre le problème de la navigation) et dont la position est recalculée pour maintenir le robot dans sa zone centrale (les obstacles qui entourent le robot sont toujours disponibles, même s'ils ne sont pas visibles de l'emplacement actuel).

La conception de ce module comprend:

(i) l'intégration de l'information capteur (scans du Laser) dans le modèle.

(ii) la mise à jour de la position de la grille d'occupation pour maintenir le robot centré.

Pour intégrer le scan laser dans le modèle, nous considérons qu'un scan est un nuage de points où:

(a) à chaque point, il existe un obstacle (cellule occupée).

(b) la ligne qui relie chaque point d'obstacle au capteur est l'espace libre (cellules libres), voir la figure 4.2a.

Cette procédure est mise en œuvre par l'utilisation de l'algorithme *Bresenham*, [57] qui est optimal dans le nombre de cellules visitées pour projeter une ligne dans une grille. Cet algorithme permet de réduire considérablement le temps d'intégration d'une mesure sensorielle.

Deuxièmement, pour maintenir le robot dans la zone centrale de la grille on définit une zone appelée zone de contrôle (*control zone*). Quand le robot quitte cette zone, la nouvelle position de la grille est recalculée pour centrer le robot dans cette zone. Le robot est toujours dans la zone centrale de la grille, dont la position ne change pas jusqu'à ce que le robot change de position. Le recalcul de la position de la grille est toujours fait en multiple de la dimension de la cellule et la rotation n'est pas autorisée (cette stratégie réduit la diffusion de fausses informations des obstacles dans les cellules, ce qui constitue une source importante d'erreur). En outre, cette stratégie peut être efficacement mis en œuvre avec des *memory shifts* pour réduire le temps de calcul.

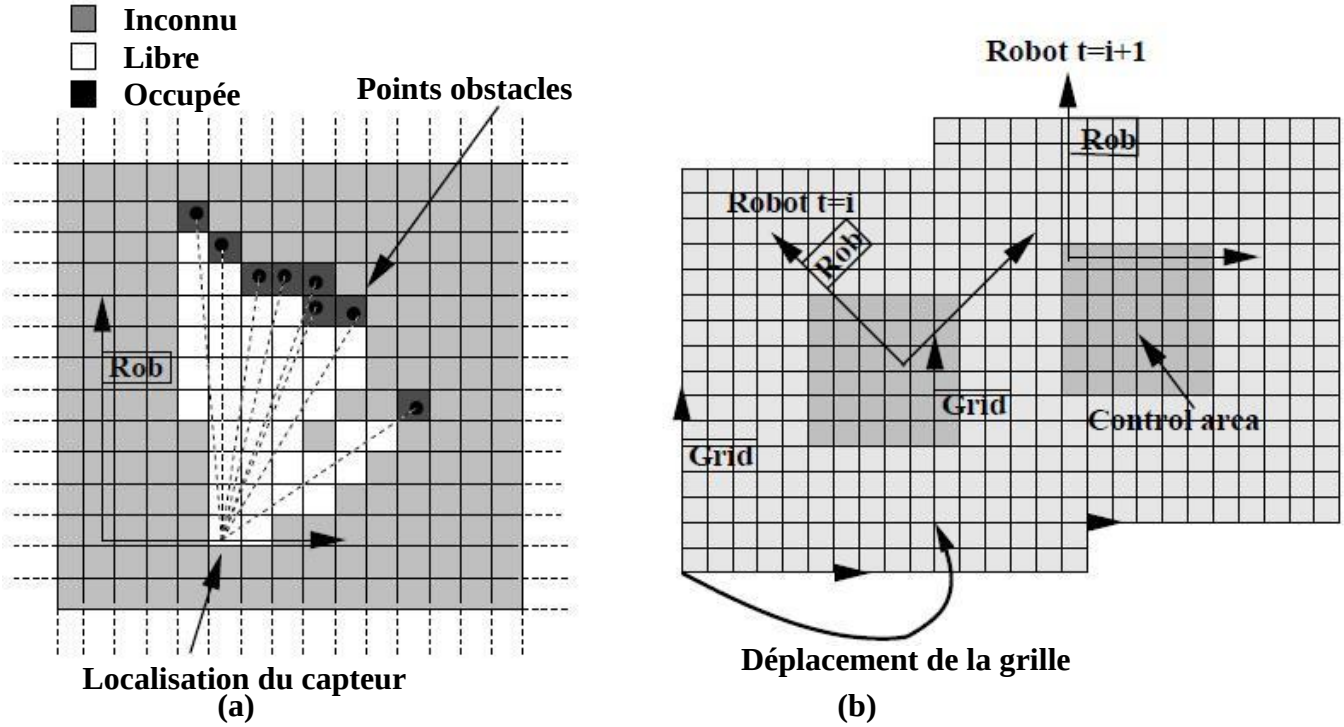


Fig.4.2 – *Ces figures montrent comment les mesures laser sont intégrés dans la grille et comment la position de grille est recalculée pour le laisser robot dans la zone centrale.*
(a) Dans le scan laser, les cellules sont occupées pour les points obstacles, et les cellules dans les lignes qui joignent les points obstacles et le capteur sont libres.
(b) À l'instant t = i le robot est dans la zone de contrôle de la grille. Ensuite, à l'instant t = i + 1, le robot a quitté la zone de contrôle et l'emplacement de la grille est recalculé (en multiples de taille de cellule et sans rotation) afin que la nouvelle position du robot soit dans cette zone.

En résumé, ce module intègre les données du capteur de telle sorte que le domaine spatial de l'information est augmenté alors que le robot évolue. En outre, le modèle représente rapidement les changements dans l'environnement (évolution des scénarios) en mettant à jour tout l'espace couvert par la dernière perception. Pour ces raisons, ce module est conforme à l'exigence de ***l'intégration de l'information*** de la Section 3. Dans ce qui suit, nous abordons le module de planification.

4.5.2 Module de planification

4.5.2.1 Description du fonctionnement du module

Ce module utilise un planificateur de mouvement pour obtenir des informations afin d'éviter les situations de piège et les mouvements cycliques (pas pour contrôler le robot). Le planificateur utilise une fonction de navigation (NF1) sur la grille du module précédent, et calcule alors un chemin vers la destination en utilisant une stratégie de descente de gradient. Nous avons choisi ce planificateur parce que la fonction de navigation n'a pas de minima locaux (elle trouve un chemin s'il existe), et c'est une fonction numérique qui fonctionne d'une manière efficace sur une grille (i.e., la représentation existante).

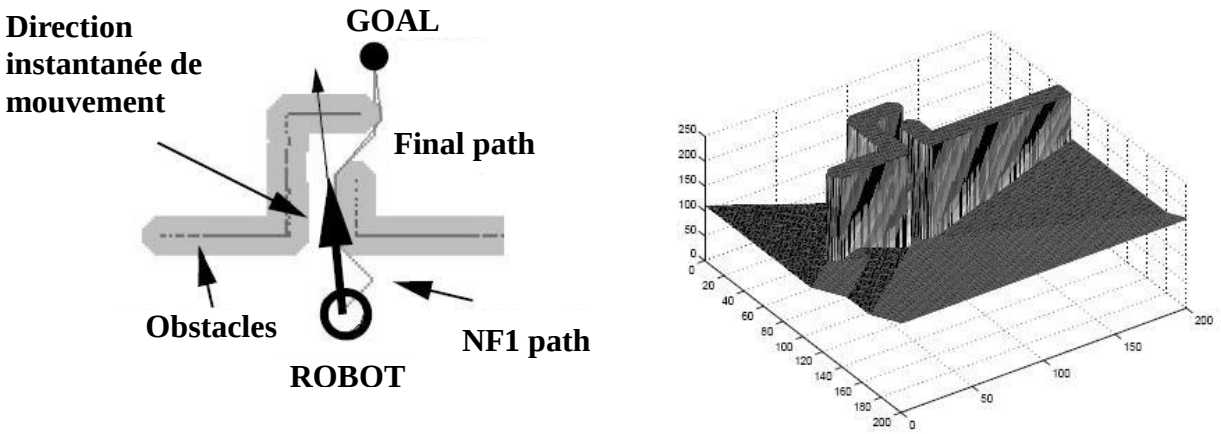


Fig.4.3 – *Cette figure montre le fonctionnement du planificateur. Premièrement, les obstacles sont agrandis au rayon du robot (a). Ensuite, la NF1 est calculée en propageant une vague à partir de la position du but, où chaque cellule est étiquetée avec le coût cumulé de la vague (b). Sur cette fonction un chemin est calculé grâce à une stratégie de recherche basée sur la descente du gradient, le chemin NF1 dans la figure (a). Enfin, le chemin est étiré (pour être optimisé) afin d'obtenir la direction instantanée du mouvement à partir de la première partie du chemin.*

Le planificateur utilise un processus à deux étapes pour calculer un chemin de la position du robot à la destination (figure 4.3).

Premièrement, la fonction de navigation est calculée. Chaque obstacle est élargi au rayon du robot, puis la NF1 est construite par propagation d'une vague à partir de la destination sur les cellules libres (chaque cellule est marquée par la distance mesurée en nombre de cellules jusqu'au but).

Deuxièmement, un chemin est calculé en utilisant une stratégie de recherche basée sur la descente du gradient.

Toutefois, ce chemin est optimal dans la métrique définie sur la grille. Ainsi, dans un processus itératif, le chemin est tendu pour le rendre optimal dans l'espace de configuration. Cela évite les effets de frontière de la NF1 et les chemins qui frôlent les frontières des obstacles (figure 4.3a).

Deux types d'informations sont obtenus à partir du chemin. Premièrement, la possibilité d'atteindre le but à partir de la position courante (le planificateur trouve le chemin s'il existe). Deuxièmement, la *direction du mouvement instantanée* (la première partie du chemin, voir les figures 4.3a et 4.4). Cette direction sera utilisée pour informer à propos du mouvement, mais pas comme un chemin à exécuter (car la génération de mouvements sera traitée par le module réactif).

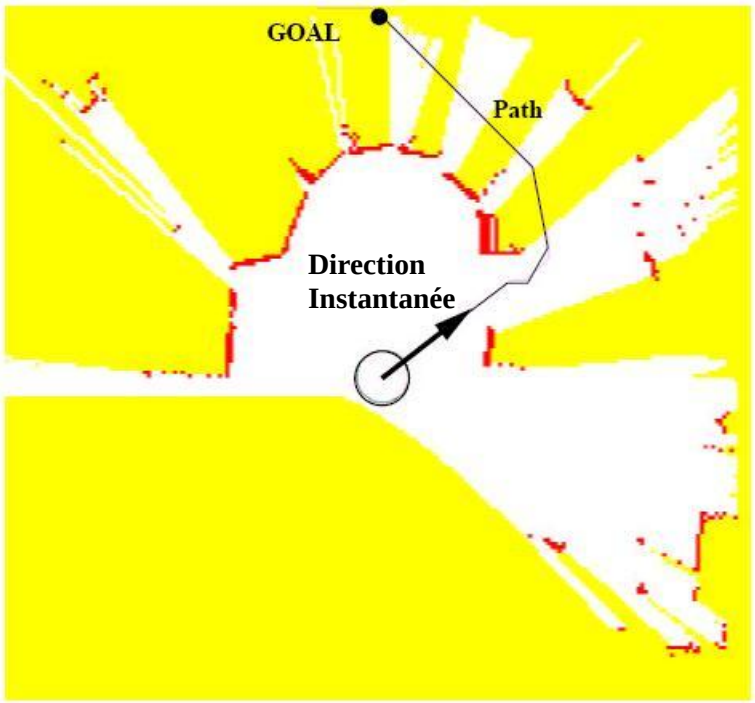


Fig.4.4 – *Cette figure montre une expérience où le module de planification calcule un chemin vers la destination sur la grille. La direction instantanée du mouvement est extraite de la direction principale du chemin. Cette direction vise vers la sortie de l'obstacle de forme U, cette sortie est utilisée pour éviter la situation de piège.*

Autrement dit, la direction instantanée est l'information tactique calculée par le planificateur pour éviter les situations de piège. Ceci est illustré graphiquement dans la figure 4.4, où le robot était en face d'un obstacle de la forme U où il pourrait être pris au piège. Néanmoins, après avoir calculer le chemin et extraire la direction instantanée, cette dernière vise vers la sortie.

En utilisant cette direction, le robot évite la situation de piège. L'aspect calcul de ce module est limité à la taille de la grille.

Avec une taille de 200 × 200 cellules, le module prend environ 0.08sec (dans le pire des cas) et peut donc être effectué dans le cycle sensoriel après la construction du modèle. Concernant l'intégration, le temps de calcul de la fonction de navigation est proportionnel au carré du nombre de cellules. Ainsi, l'augmentation de la taille de la grille devrait augmenter le temps de calcul (ce qui pénalise l'inclusion dans le cycle sensoriel). D'autre part, la réduction de la taille de la grille ou de sa résolution diminue la précision ou le domaine spatial. Autrement dit, la taille du modèle (portion de l'espace et de précision) engage le temps de calcul du planificateur, et tous les deux doivent être équilibrés dans une implémentation réelle.

Il convient de noter que la direction du mouvement instantané contient une information tactique pour **éviter les mouvements cycliques et les situations de piège** [56] (deuxième exigence de la section 3). Dans les sections suivantes, nous analysons comment le système utilise cette information pour résoudre ces situations.

4.5.2.2 Fonctions de navigation

La planification globale de trajectoire implique la recherche d'un chemin sûr à partir de la position initiale du robot jusqu'au but dans un environnement connu. Il existe de nombreuses approches qui tentent de résoudre le problème. L'une d'elles est basée sur la fonction de navigation lorsque le problème de planification de trajectoire est transformé en un problème de descente de gradient de la position initiale jusqu'au but.

L'algorithme basé sur une fonction de navigation discrétise l'espace de travail $\boldsymbol{W}$ du robot en une grille de cellules rectangulaires $\boldsymbol{gC}$. En plus, un cadre fixe $\boldsymbol{FW}$ est incorporé dans l'espace de travail du robot. La position d'un $\boldsymbol{gC}$ peut donc être spécifiée comme des coordonnées cartésiennes par rapport à $\boldsymbol{FW}$. Chaque $\boldsymbol{gC}$ est un espace libre ou occupé. Les sous-ensembles de $\boldsymbol{gC}$ dans l'espace libre et occupé sont désignés respectivement *par* $\boldsymbol{gC_{free}}$ *et* $\boldsymbol{gC_{occupied}}$. L'algorithme de propagation de vagues "wavefront" est utilisé pour calculer la fonction de navigation.

L'algorithme est pratique, facile à mettre en œuvre et robuste. Le chemin généré par la fonction de navigation est calculé en quatre étapes:

Premièrement, les coordonnées des cellules libres aux frontières de chaque obstacle dans l'espace de travail, dénoté *par* $\boldsymbol{gC_{border}}$, sont extraites et stockées dans une liste FIFO $\boldsymbol{L_B}$. $\boldsymbol{gC_{border}}$ est défini comme tout $\boldsymbol{gC_{free}}$ dans la grille $\boldsymbol{gC}$ qui a au moins un voisin $\boldsymbol{gC_{occupied}}$.

Deuxièmement, les régions à risque sont calculées à partir de $\boldsymbol{gC_{border}}$. La région à risque comprend toutes les cellules qui sont au moins de $\boldsymbol{\alpha}$ distance de la frontière de l'obstacle. Les régions à risque sont alors marquées comme cellules occupées. Cela empêchera le chemin calculé d'être plus proche des obstacles.

Troisièmement, les valeurs *N* de la fonction de navigation sont calculées et appliquées au reste *de* la gC_{free} en utilisant l'algorithme de propagation de vagues "wavefront".

Quatrièmement, un chemin de longueur minimale peut donc être généré en suivant la descente du gradient de *N*. Les coordonnées de toutes les cellules de la grille qui constituent le chemin sont stockés dans une liste L_P.

Le pseudo-code pour le calcul du chemin généré par la fonction de navigation est comme suit:

Algorithme 4.1: calcul du chemin généré par la fonction de navigation

Etape 1-Extraction des cellules libres aux frontières des cellules obstacles

1. begin
2. for every gC_{free} in the map do
3. if there exist a $gC_{occupied}$ neighbor then
4. begin
5. insert coordinates of gC_{free} to end of L_B;
6. end;
7. end;

Etape 2-Calcul des régions à risque

1. begin
2. for every element i in L_B, where i = 1,2,.. do
3. for every neighbor of the cells in L_{Bi} do
4. if the neighbor is gC_{free} and $< \alpha$ distance then
5. begin
6. neighbor ← occupied;
7 insert coordinate of neighbor at the end of L_B;
8. end;
9. end;

Etape 3-Calcul des valeurs *N* de la fonction de navigation

1. begin
2. for every gC_{free} in the map do
3. $N \leftarrow M$ (large number)
4. N for goal cell $\leftarrow 0$;
5. insert coordinates of the goal cell to the end of L_0;
6. for every element i in L_0, where $i = 1,2,\ldots$ do
7. for every 1-neighbor cells of L_{0i} do
8. if $N = M$ then

9. begin
10. $N \leftarrow N$ of $L_{0i} + 1$;
11. insert coordinate of 1-neighbor at end of L_0;
12. end;
13. end;

Etape 4 – Generaion du chemin optimal L_P

1. begin
2. $temp \leftarrow$ cell coordinates of robot currently occupying;
3. insert ***temp*** to the end of L_P;
4. while N for $temp \neq 0$ do
5. $temp \leftarrow$ coordinate of ***min*** (neighboring cells of ***temp***);
6. insert ***temp*** to the end of L_P;
7. end;

La figure 4.5 montre la définition de 1-voisin et la priorité des cellules voisines. La fonction **min** retourne les coordonnées (x, y) de la cellule voisine qui a la plus petite valeur ***N***. Toutefois, dans le cas où il y a plus d'une cellule voisine ayant la même plus petite valeur ***N***, la cellule avec la priorité la plus élevée sera retenue.

La figure 4.6 montre un chemin généré par la fonction de navigation. Les cases noires sont les obstacles et le gris représente les cellules à risque.

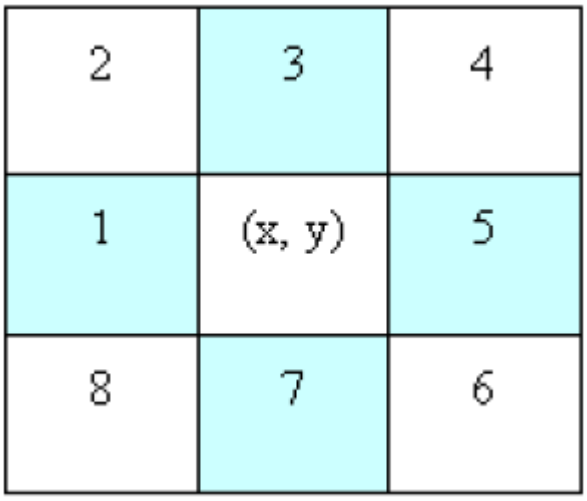


Fig.4.5 – *Les cellules colorées sont les 1-Voisin de la cellule (x, y) et le nombre indique la priorité des cellules voisines.*

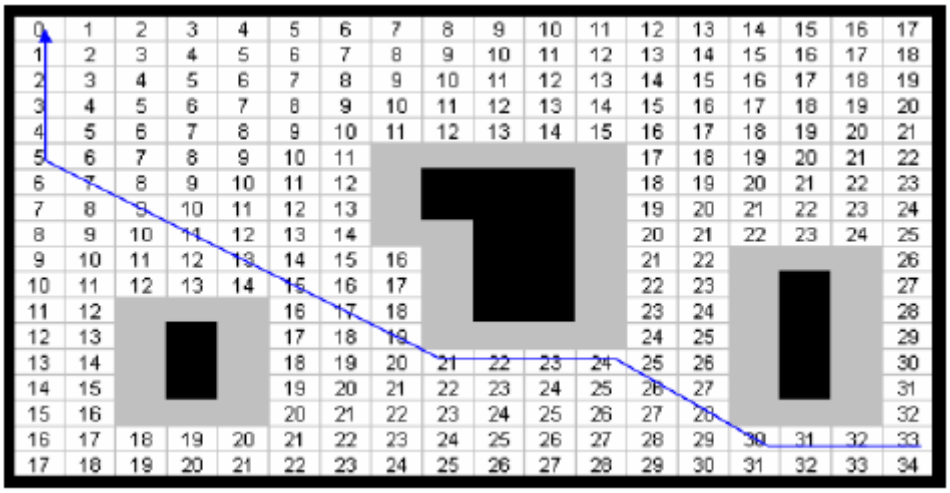


Fig.4.6 – *Chemin généré par la fonction de navigation.*

La fonction de navigation nécessite un environnement connu et statique. Un chemin libre et continu peut être trouvé à l'avance par l'analyse de la connectivité de l'espace libre. La méthode trouve toujours un chemin libre et continu s'il existe. Toutefois, tout changement dans l'environnement pourrait invalider cette hypothèse puisque les valeurs ***N*** doivent être recalculées lorsqu'il y a des changements dans l'environnement. En outre, une information complète sur l'environnement est nécessaire pour calculer la fonction de navigation. Par conséquent, la fonction de navigation n'est généralement pas adaptée pour la navigation dans un environnement initialement inconnu avec des obstacles dynamiques.

Dans la section suivante, nous décrivons le module réactif qui calcule le mouvement.

4.5.3 Module de la navigation réactive

La méthode VFH + est une version améliorée du *Vector Field Histogramme* (VFH), méthode locale développée à l'origine par Borenstein et Koren [1991][10] pour l'évitement d'obstacles en temps réel par les robots mobiles. VFH + a été développé pour un type particulier de robot mobile appelé le *GuideCane.* Le GuideCane est un dispositif de guidage pour les aveugles. En opération, un utilisateur aveugle pousse le GuideCane devant lui. Lorsque le GuideCane rencontre un obstacle il tourne et l'utilisateur suit la nouvelle orientation du périphérique intuitivement et sans aucun effort conscient.

En raison de la similitude des fonctions entre le GuideCane et les robots mobiles conventionnels , la méthode d'évitement d'obstacles VFH + est tout aussi bien applicable à d'autres robots mobiles.
L'algorithme VFH + a été testé à grande échelle en simulation et avec un GuideCane réel en environnement non structuré et inconnu.

4.5.3.1 L'algorithme VFH+

Le concept de l'algorithme d'évitement obstacle VFH+ est similaire à l'algorithme original VFH. L'entrée de cet algorithme est une grille de la carte de l'environnement local, appelée *histogram grid* [10], basée sur les méthodes antérieures de *certainty grid* [Moravec, 1988] [58] et grille d'occupation *(occupancy grid*) [Elfes, 1989] [59].

La méthode VFH + emploie processus de quatre étape de réduction de données afin de calculer la nouvelle direction du mouvement. Au cours des trois premières étapes, la grille de la carte de deux dimensions est réduite à un *histogramme polaire* à une dimension construit dans l'espace autour du robot .

Dans la quatrième étape, l'algorithme sélectionne la direction la plus appropriée basée sur l'histogramme polaire masqué et une fonction de coût.
Les sections suivantes résument brièvement chaque étape.

Algorithme 4.2: VFH+

4.5.3.1.1 Première étape – l'histogramme polaire primaire

Une description plus détaillée de la première partie de cette étape est donnée dans [Borenstein et Koren, 1991][10]. La première étape de réduction de données utilise la région active C_a de la grille de la carte C pour construire *l'histogramme polaire* H^p . La région active C_a est une fenêtre circulaire de diamètre w_s qui se déplace avec le robot. Le contenu de chaque cellule active dans la grille de la carte est traité comme vecteur d'obstacle. Basé sur le système de référence de la figure 4.9, le vecteur de direction $\beta_{i,j}$ est déterminé par la direction de la cellule active vers le point du centre du robot *(RCP):*

$$\beta_{i,j} = \tan^{-1}\left(\frac{y_0 - y_j}{x_i - x_0}\right) \qquad \textbf{(1)}$$

Où:

x_0, y_0 : Coordonnées actuelles du *RCP.*

x_i, y_j : Coordonnées de la cellule active $C_{i,j}$.

L'amplitude du vecteur d'une cellule active $\boldsymbol{C_{i,j}}$ est donné par:

$$m_{i,j} = c_{i,j}^2\left(a - bd_{i,j}^2\right) \qquad \textbf{(2)}$$

Où:

$\boldsymbol{c_{i,j}}$: Valeur de certitude de la cellule active $\boldsymbol{C_{i,j}}$.

$\boldsymbol{d_{i,j}}$: Distance de la cellule active $\boldsymbol{C_{i,j}}$ au *RCP.*

Les paramètres ***a*** et ***b*** sont choisis en fonction de:

$$a - b\left(\frac{w_i - 1}{2}\right)^2 = 1 \qquad \textbf{(3)}$$

Notez que $\boldsymbol{c_{i,j}}$ est au carré. Ceci exprime notre confiance dans la lecture des valeurs (high $\boldsymbol{c_{i,j}}$) qui constituent des obstacles réels, par opposition à une simple occurrence de lecture (low $\boldsymbol{c_{i,j}}$) qui peut être causée par le bruit.

L'amplitude du vecteur est aussi fonction du carrée de la distance $\boldsymbol{d_{i,j}}$ **.**

Les cellules occupées produisent de plus grandes amplitudes de vecteurs quand elles sont proches du robot.

Une propriété de cette fonction d'amplitude est la symétrie de rotation par rapport au *RCP.*

En conséquence, le comportement du robot est indépendant de la direction dans laquelle se trouvent les obstacles rencontrés.

L'histogramme polaire primaire $\boldsymbol{H^p}$ est construit sur la base des vecteurs d'obstacles.

$\boldsymbol{H^p}$ a une résolution angulaire arbitraire ***a*** tel que $\boldsymbol{n = 360° / a}$ est un entier. Dans notre implémentation, ***a*** est fixé à **5°**, ce qui donne $\boldsymbol{n = 72}$ secteurs angulaires. Chaque secteur angulaire ***k*** correspond à un angle discret $\boldsymbol{\rho = k \cdot a}$**.**

La méthode VFH originale ne prend pas en compte la largeur du robot. Au lieu de cela, elle utilise un filtre passe-bas déterminé empiriquement pour compenser la largeur du robot et de lisser l'histogramme polaire. Le réglage de ce filtre est la principale difficulté dans l'application de l'algorithme VFH original. Toutefois, même avec un filtre bien réglé, le robot a tendance à couper coins.

La méthode VFH +, en revanche, utilise un filtre passe-bas déterminée théoriquement pour compenser la largeur du robot. Les cellules d'obstacles dans la carte sont agrandies par le rayon du robot r_r, Qui est défini comme la distance entre le Centre du robot au point le plus éloigné de son périmètre.

Pour plus de sécurité, les cellules d'obstacles sont en fait élargies par un rayon $r_{r+s} = r_r + d_s$ où d_s est la distance minimale entre le robot et un obstacle.

Avec les obstacles agrandis par r_{r+s} , Le robot peut être traité comme un véhicule point. Cette méthode fonctionne bien pour les robots mobiles dont la forme peut être approchée par un disque. Si la forme du robot est très asymétrique, les cellules obstacles devront être agrandies en fonction de dimensions et l'orientation momentanée du robot.

Cette méthode de compensation de la largeur est mise en œuvre très efficacement en élargissant les obstacles tout en construisant l'histogramme polaire primaire. Au lieu de mettre à jour uniquement un secteur histogramme pour chaque cellule comme cela se fait dans la méthode originale VFH, tous les secteurs de l'histogramme qui correspondent à la cellule agrandie sont mis à jour. Un exemple d'une cellule est montré dans la figure 4.7.

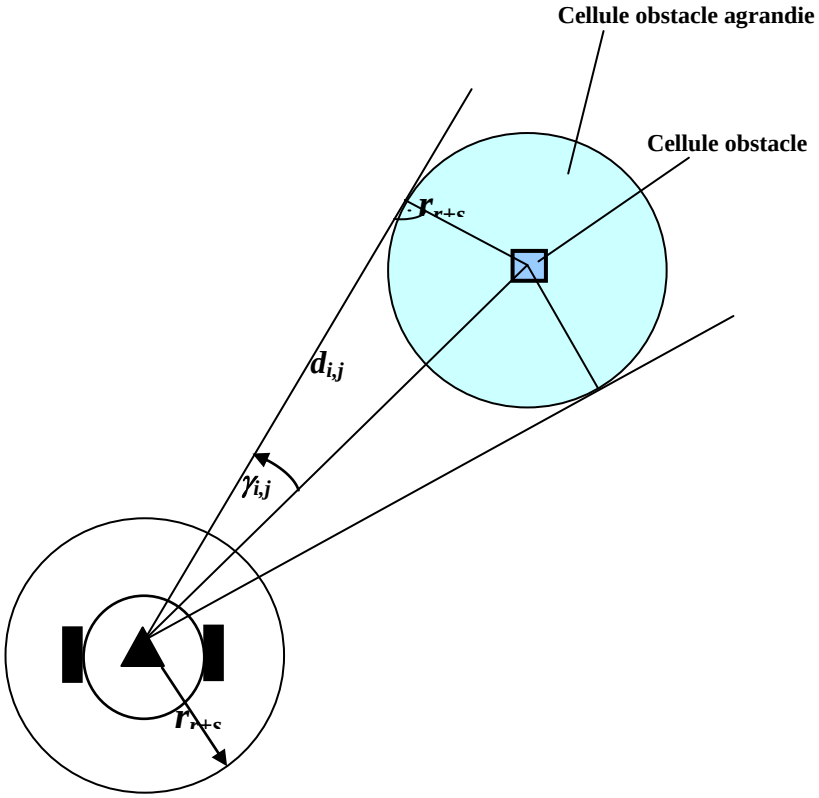


Fig.4.7 – *angle agrandi*

Pour chaque cellule, l'angle de l'élargissement $\gamma_{i,j}$ est défini par:

$$\gamma_{i,j} = \arcsin\frac{r_{r+s}}{d_{i,j}} \qquad \textbf{(4)}$$

Pour chaque secteur ***k***, la densité d'obstacle polaire est alors calculée par:

$$H_k^p = \sum_{i,j \in c_a} m_{i,j} . h'_{i,j} \qquad \textbf{(5)}$$

avec:

$h'_{i,j=1}$ **si** $k \cdot \alpha \in \left\lfloor \beta_{i,j} - \gamma_{i,j}, \beta_{i,j} + \gamma_{i,j} \right\rfloor$ **(6)**

$h'_{i,j=0}$ sinon

Le résultat de ce processus est un histogramme polaire qui prend en compte la largeur du robot.
La fonction h' sert aussi comme un filtre passe-bas et adoucit l'histogramme polaire.
Une autre amélioration importante est que ce processus élimine les difficultés de réglage du filtre passe-bas de la méthode VFH.
Comme l'histogramme est construit autour de la position courante du robot, indépendamment de son orientation, cette première étape de l'algorithme VFH peut être implémentée très efficacement par l'utilisation de tables de la taille $w_s * w_s$.
Les β_{ij} , γ_{ij} et $a - bd^2_{i,j}$ valeurs de chaque cellule active dans la région active C_a peuvent être stockées dans des tables pour une exécution plus rapide.

4.5.3.1.2 Deuxième étape - The Binary Polar histogram (*l'histogramme polaire binaire*)

Pour la plupart des applications, une trajectoire lisse est souhaitée et des oscillations dans la commande de direction devraient être évitées. La méthode originale VFH affiche généralement une trajectoire très lisse. Toutefois, le seuil fixé τ utilisé dans la méthode originale VFH peut causer un problème dans des environnements avec plusieurs ouvertures étroites, comme l'ouverture correspondante dans l'histogramme peut alterner plusieurs fois entre un état ouvert et un état bloqué pendant quelques temps d'échantillonnage. Dans une telle situation, la position du robot peut alterner plusieurs fois entre cette étroite ouverture et une autre ouverture. Le résultat est un comportement indécis, au cours duquel le robot mobile peut être très proche d'un obstacle. Ce problème peut

facilement être réduit par l'utilisation de deux seuils, à savoir τ_{bas} et τ_{haut} . Sur la base de l'histogramme polaire primaire H^p et les deux seuils, un *histogramme polaire binaire* H^b est construit. Au lieu d'avoir les valeurs de densité polaire, les secteurs du H^b sont soit *libres* **(0)** ou *bloqués* **(1).** Cet histogramme polaire indique quelles sont les directions libres pour un robot qui peut instantanément changer la direction du mouvement. L'histogramme polaire binaire est mis à jour par les règles suivantes:

$$H^b_{k,j} = 1 \quad \textbf{si} \quad H^P_{k,j} > \tau_{high}$$

$$H^b_{k,j} = 0 \quad \textbf{si} \quad H^P_{k,j} < \tau_{low} \qquad \textbf{(7)}$$

$$H^b_{k,j} = H^b_{k,j-1} \quad \textbf{sinon}$$

4.5.3.1.3 Troisième étape - The Masked Polar histogram(Histogramme polaire masqué)

La méthode originale VFH néglige la dynamique et la cinématique du robot. Elle suppose implicitement que le robot est capable de changer de sens de direction instantanément comme le montre la Figure 4.8a. À moins que le robot s'arrête à chaque fois, cette hypothèse est manifestement violée.

La VFH + utilise une simple, mais une plus proche approximation de la trajectoire de la plupart des robots mobiles. Elle suppose que la trajectoire du robot est basée sur des arcs circulaires (courbes à courbures constantes) et des lignes droites, comme illustré à la figure 4.8b. La courbure d'une courbe est définie par ***k=1/r***.

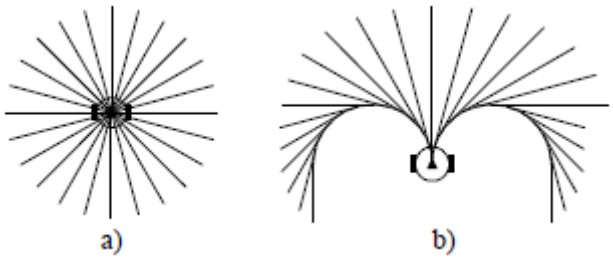


Fig.4.8 – *approximation des trajectoires:*
a) sans dynamique, b) avec dynamique

La courbure maximale de la trajectoire d'un robot mobile est souvent fonction de la vitesse du robot. Plus le robot se déplace rapidement, plus la courbure maximale est plus petite. Le rayon de braquage minimum peut être nul pour un robot mobile différentiel si sa vitesse transversale est nulle.

Pour les robots mobiles à base de véhicule d' *Ackerman* ou de mécanisme de *tricycle*, le rayon de braquage minimal n'égale jamais zéro. Dans ces cas, le rayon de braquage minimal est approximativement constant si la vitesse maximale n'est pas trop élevée. Dans des cas particuliers, par exemple le GuideCane, les valeurs de courbure maximale peuvent être différentes pour aller tout droit et tourner à gauche.

Les valeurs pour le rayon de braquage minimum en tant que fonction de la vitesse du robot peuvent être facilement mesurées. Nous définissons ces rayons pour les deux cotés comme $r_r = 1/k_r$ *et* $r_l = 1/k_l$. Avec ces paramètres et la grille de la carte, on peut déterminer quels secteurs sont bloqués par des obstacles. Un exemple avec deux obstacles est illustré à la figure 4.9. De nouveau, pour tenir compte de la largeur du robot, les obstacles sont agrandis par r_{r+s}. Si un cercle trajectoire et une cellule obstacle agrandie se chevauchent, toutes les directions de l'obstacle en sens inverse du mouvement sont bloqués. Dans notre exemple, l'obstacle **A** bloque toutes les directions à sa gauche à cause de la dynamique du robot. D'autre part, L'obstacle **B** ne bloque pas les directions à sa droite.

Avec la méthode VFH d'origine, les directions à gauche de l'obstacle **A** sont considérées comme des directions appropriées du mouvement. Si la direction souhaitée est à gauche, l'algorithme VFH original pourrait guider incorrectement le robot vers la gauche de l'obstacle **A**.

Avec méthode VFH +, le robot pourrait aller correctement entre les obstacles **A** et **B** et de faire un virage à gauche après l'obstacle **A**.

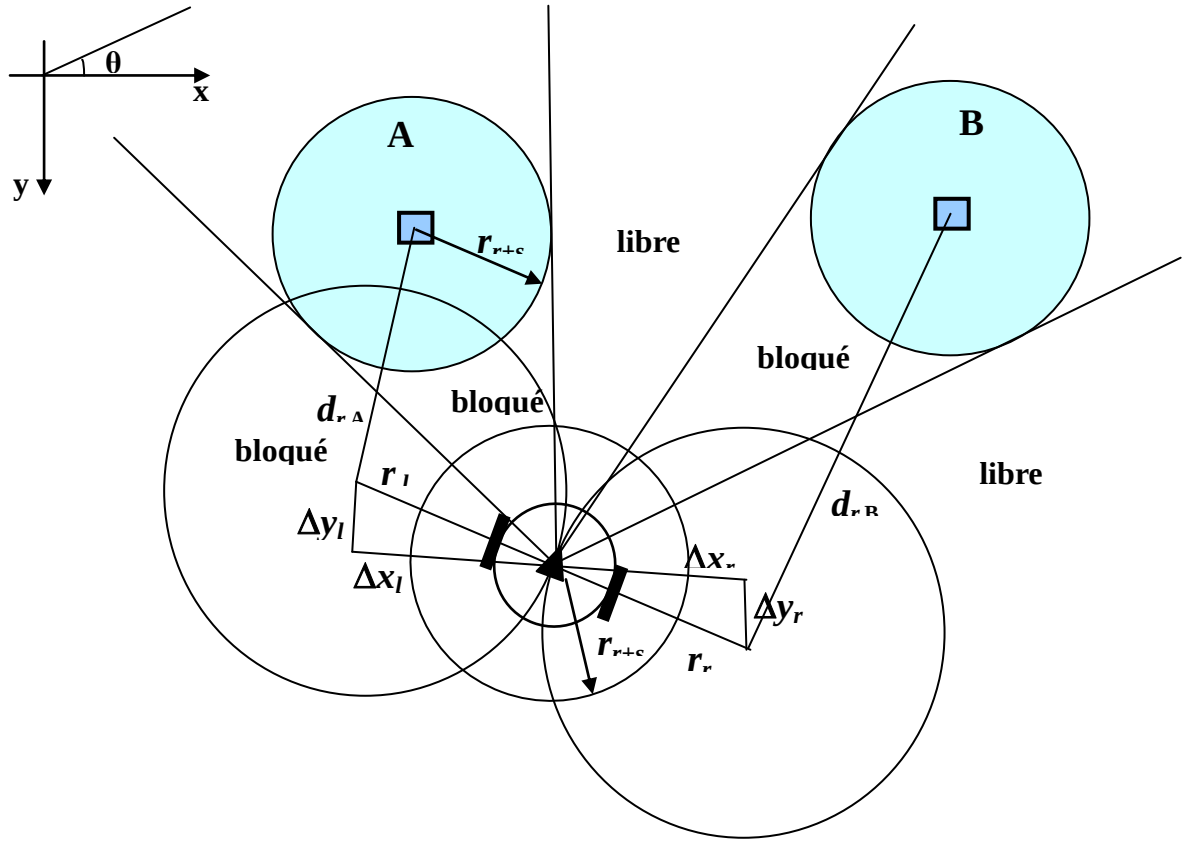


Fig.4.9 – *Exemple de directions bloquées*

Les positions des centres de trajectoire droit et gauche relatives à la position actuelle du robot sont définies par:

$$\Delta x_r = r_r \cdot \sin\theta \qquad \Delta y_r = r_r \cdot \cos\theta$$
$$\Delta x_l = -r_l \cdot \sin\theta \qquad \Delta y_l = -r_l \cdot \cos\theta \qquad \textbf{(8)}$$

Les distances à partir d'une cellule active $\boldsymbol{C_{i,j}}$ aux deux centres de trajectoire sont donnés par:

$$d_r^2 = (\Delta x_r - \Delta x(j))^2 + (\Delta y_r - \Delta y(i))^2$$
$$d_l^2 = (\Delta x_l - \Delta x(j))^2 + (\Delta y_l - \Delta y(i))^2 \qquad \textbf{(9)}$$

Un obstacle bloque les directions à sa droite si:

$$d_r^2 < (r_r + r_{r+s}) \quad \textbf{[condition 1]} \qquad \textbf{(10a)}$$

Et un obstacle bloque les directions à sa gauche si:

$$d_l^2 < (r_l + r_{r+s}) \quad \textbf{[condition 2]} \qquad \textbf{(10b)}$$

En vérifiant chaque cellule active avec ces deux conditions, on obtient deux angles limites, $\boldsymbol{\varphi_r}$ pour les angles de droite et $\boldsymbol{\varphi_l}$ pour des angles de gauche. Nous définissons également $\boldsymbol{\varphi_b = \theta + \Pi}$ comme la direction vers l'arrière pour l'orientation actuelle du mouvement.

Cette méthode peut être implémentée très efficacement par un algorithme qui ne considère que les cellules qui ont une influence sur $\boldsymbol{\varphi_r}$ ou $\boldsymbol{\varphi_l}$:

1) Déterminer $\boldsymbol{\varphi_b}$. Mettre $\boldsymbol{\varphi_r}$ et $\boldsymbol{\varphi_l}$ égal à $\boldsymbol{\varphi_b}$.
2) Pour chaque cellule $\boldsymbol{C_{i,j}}$ dans la fenêtre active $\boldsymbol{C_a}$ avec $\boldsymbol{c_{i,j} > \tau}$:
   a) Si $\boldsymbol{\beta_{i,j}}$ est à la droite de θ et à la gauche de $\boldsymbol{\varphi_r}$, Vérifiez Condition 1. Si la condition est satisfaite, mettre $\boldsymbol{\varphi_r}$ égal à $\boldsymbol{\beta_{i,j}}$.
   b) Si $\boldsymbol{\beta_{i,j}}$ est à la gauche de θ et à droite de $\boldsymbol{\varphi_l}$, Vérifiez Condition 2. Si la condition est satisfaite, mettre en $\boldsymbol{\varphi_l}$ égal à $\boldsymbol{\beta_{i,j}}$.

Si les capteurs du robot ne sont pas très fiables, φ_r et φ_l pourraient également être déterminés de façon aléatoire. Au lieu de comparer les valeurs de certitude de la cellule à un seuil, on pourrait construire un histogramme polaire dont les valeurs du secteur indiquent la certitude qu'un secteur est bloqué à cause de la dynamique du robot. Les valeurs de φ_r et φ_l pourraient alors être déterminées par application d'un seuil à cet histogramme. Comme la première méthode est plus efficace, la deuxième méthode ne devrait être appliquée que si vraiment c'est nécessaire.

Avec φ_r , φ_l, et l'histogramme polaire binaire, on peut construire *l'histogramme polaire masqué(the masked polar histogram):*

$$H_k^m = 0 \quad \textbf{si} \quad H_k^b = 0 \ \textbf{and} \ (k \cdot \alpha) \in \{[\varphi_r, \theta], [\theta, \varphi_l]\}$$
$$H_k^m = 1 \quad \textbf{sinon} \qquad \textbf{(11)}$$

L'histogramme polaire masqué montre quelles directions de mouvement sont possibles à la vitesse actuelle. Si tous les secteurs sont bloqués, le robot ne peut pas aller à la vitesse actuelle. Le robot devra déterminer une série de nouvelles valeurs *(* φ_r , φ_l) Basées sur une vitesse plus lente.
Si l'histogramme polaire masqué reste toujours bloqué dans toutes les directions, le robot devrait s'arrêter immédiatement.

L'histogramme polaire masqué peut donc également être utilisé pour détecter que le robot est piégé dans une impasse.

Dans la figure 4.10, l'histogramme polaire primaire, l'histogramme polaire binaire, et l'histogramme polaire masqué sont montrés pour la situation de la figure 4.9 , l'histogramme polaire binaire indique incorrectement que les directions vers la gauche de l'obstacle **A** sont libres. L'histogramme polaire masqué bloque correctement ces directions.

Notez que les amplitudes du vecteur pour l'obstacle **A** sont plus grandes que pour l'obstacle **B**. La raison en est que l'obstacle **A** est plus proche du robot. Notez également que l'obstacle **A** occupe plus de secteurs que l'obstacle **B**. Comme l'obstacle **A** est plus proche du robot, son angle agrandi est plus large.

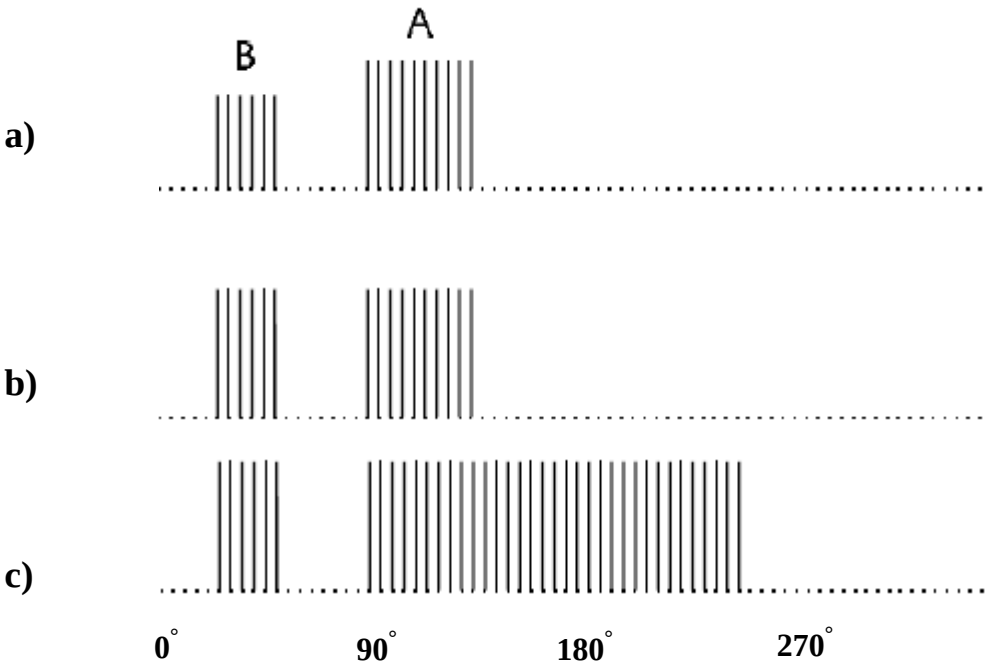


Fig.4.10 – *a) histogramme polaire primaire, b) histogramme polaire binaire, c) histogramme polaire masqué*

4.5.3.1.4 Étape quatrième - Sélection de la Direction de pilotage

L'histogramme polar masqué montre quelles directions sont libres d'obstacles et celles qui sont bloquées. Toutefois, certaines directions libres sont de meilleures candidates que d'autres pour la nouvelle direction du mouvement.

La méthode VFH d'origine est très orientée but en sélectionnant la vallée qui se rapproche le plus de la direction du but k_t . Elle sélectionne alors la nouvelle orientation du mouvement dépendant de la taille de la vallée.

La méthode VFH + trouve d'abord toutes les ouvertures dans l'histogramme polaire masqué et détermine ensuite un ensemble de directions candidates possibles. Une fonction de coût qui prend en compte non seulement la différence entre la direction candidate et la direction du but, est ensuite appliquée à ces directions candidates. La direction candidate k_n avec le coût le plus bas est alors choisie pour être la nouvelle direction du mouvement $\varphi_{n = \alpha}. k_n$.

Dans la première étape, les cotés droit et gauche k_r et k_l de tous les passages dans l'histogramme polaire masqué sont déterminés. Similaire à la méthode VFH d'origine, deux types de passages sont alors distingués, à savoir, *large* et *étroit*. Un passage est considéré comme large si la différence entre ses deux cotés est plus grand que s_{max} secteurs (dans notre système $s_{max} = 16$). Dans le cas contraire, le passage est considéré comme étroit.

Pour un passage étroit, il n'y a qu'une seule direction candidate de sorte que le robot passe par le centre de l'écart entre les obstacles correspondants:

$$c_n = \frac{k_r + k_l}{2} \qquad \textbf{direction centrée} \qquad \textbf{(12)}$$

Pour un passage large, il y a deux directions candidate, l'une sur le coté droit et l'autre sur le côté gauche du passage. La direction du but est aussi une direction candidate, si elle se situe entre les deux autres directions candidates:

$$c_r = k_r + \frac{s_{\max}}{2} \qquad \textbf{vers le coté droit}$$

$$c_l = k_l + \frac{s_{\max}}{2} \qquad \textbf{vers le coté gauche} \qquad \textbf{(13)}$$

$$c_t = k_t \qquad \textbf{si} \quad k_t \in [c_r, c_l]$$

Les directions candidates $\boldsymbol{C_r}$ *e*t $\boldsymbol{C_l}$ laisse le robot suivre un contour d'obstacle avec une distance de sécurité, tandis que $\boldsymbol{C_t}$ conduit le robot vers la direction du but.

Pour les robots qui ne sont pas orientés but, d'autres directions candidates pourraient être ajoutées. Pour un robot qui devrait explorer au hasard son environnement, on pourrait ajouter les directions candidates qui sont égales à l'actuelle direction du mouvement $\boldsymbol{\theta_i}$ ou égales à la direction du mouvement sélectionnée précédemment $\boldsymbol{k_{n,i-1}}$**:**

$$c_\theta = \frac{\theta_i}{\alpha} \qquad \textbf{si} \qquad \frac{\theta_i}{\alpha} \in [c_r, c_l]$$

$$c_\varphi = k_{n,i-1} \qquad \textbf{si} \qquad k_{n,i-1} \in [c_r, c_l] \qquad \textbf{(14)}$$

Dans le cas d'un Robot orienté but, on obtient entre un et trois directions candidates pour chaque passage dans l'histogramme polaire masqué. Ensuite, nous aurons besoin de définir une fonction de coût appropriée, de sorte que le robot choisira la direction candidate la plus appropriée en tant que sa nouvelle direction du mouvement $\boldsymbol{\varphi_n}$.

Nous proposons la fonction de coût ***g*** suivante comme une fonction de la direction candidate ***c:***

$$g(c) = \mu_1 \cdot \Delta(c, k_t) + \mu_2 \cdot \Delta\left(c, \frac{\theta_i}{\alpha}\right) + \mu_3 \cdot \Delta\left(c, k_{n,i-1}\right) \qquad \textbf{(15)}$$

Où $\Delta(c_1, c_2)$ est une fonction qui calcule l'angle de différence absolue entre les deux secteurs c_1 , c_2 de telle sorte que le résultat est $\leq n/2$. Une des implémentations possibles est:

$$\Delta(c_1, c_2) = \min\left\{|c_1 - c_2|, |c_1 - c_2 - n|, |c_1 - c_2 + n|\right\} \qquad \textbf{(16)}$$

Le premier terme de notre fonction de coût représente le coût associé à la différence entre la direction candidate et la direction du but. Plus cette différence est importante, plus la direction candidate guide le robot loin de sa direction cible, et donc augmente le coût.

Le second terme représente le coût associé à la différence entre la direction candidate et l'orientation de la roue du robot .

Plus cette différence est grande, plus la modification requise de la direction du mouvement est grande.

Le troisième terme représente le coût associé à la différence entre la direction candidate et la direction du mouvement précédemment sélectionné. Plus cette différence est importante, plus la variation de la commande de la nouvelle direction est plus grande.

En bref, le premier terme est responsable du comportement orienté but, tandis que le deuxième et le troisième terme oriente le robot mobile dans une direction. Ces deux termes approvisionnent le robot avec une forme de mémoire à court terme. Le second terme est similaire à une mémoire mécanique. A l'aide du troisième terme, le robot s'engage à une direction avant même que son orientation ait changée.

Plus μ_1 est plus élevé, plus le comportement orienté but du robot est plus important. Plus μ_2 est plus élevé, plus le robot tente d'exécuter une trajectoire efficace avec un minimum de modifications dans la direction du mouvement. Plus μ_3 est plus élevé, plus le robot tente de se diriger vers la direction précédemment sélectionnée et plus la trajectoire est lisse.

Seule la relation entre les trois paramètres est importante, pas leurs grandeurs. Afin de garantir un comportement orienté but, la condition suivante doit être satisfaite:

$$\mu_1 > \mu_2 + \mu_3 \qquad \textbf{[condition 3]} \qquad \textbf{(17)}$$

Si une trajectoire est plus importante que les variations dans les commandes de direction, alors μ_2 devrait être supérieure à μ_3. Si la simplicité des commandes de pilotage est plus importante que l'efficacité de la trajectoire du robot, alors μ_3 devrait être supérieure à μ_2.

Des expériences ont montré qu'une bonne série de paramètres pour un robot mobile orienté but est:

$\mu_1 = 5,\ \mu_2 = 2,\ \mu_3 = 2.$

Il est également possible d'ajouter d'autres termes pour la fonction coût. Par exemple, nous pouvons permettre au robot mobile d'éviter les passages étroits en ajoutant un terme qui prend en compte la largeur du passage:

$\mu_4 \cdot \Delta(k_r, k_l)$.

D'autre part, la fonction de coût pourrait également être temporairement modifiée pour permettre au robot mobile de chercher et aller à travers des passages étroits comme des portes en ajoutant le terme:

$\mu_4 \cdot 1/\Delta(k_r, k_l)$.

La fonction de coût permet à l'utilisateur d'implémenter un comportement plus subtil que l'approche grossière de la méthode originale VFH. L'utilité de la fonction de coût est très importante, surtout lorsque le robot s'approche d'un seul objet qui est dans son droit chemin. Sans laquelle, le robot peut effectivement tomber sur cet obstacle, en hésitant entre l'éviter sur le coté droit ou sur celui de gauche. Un autre avantage est que le comportement du robot mobile peut être facilement changé en modifiant soit les paramètres de la fonction de coût ou la fonction de coût elle-même.

4.5.3.2 Conclusion sur la méthode locale (VFH +)

La méthode VFH + est le résultat de plusieurs améliorations par rapport à la méthode VFH originale. Tout d'abord, en utilisant un seuil, la trajectoire du robot devient plus fiable. Deuxièmement, la méthode VFH + prend explicitement en compte la largeur du robot, et par conséquent, cette méthode peut être facilement implémentée sur des robots de tailles différentes. Cette amélioration élimine également le temps de réglage du filtre passe-bas utilisé précédemment. Troisièmement, la méthode VFH + prend en compte la trajectoire du robot mobile en masquant les secteurs qui sont bloqués par les obstacles. En conséquence, le robot ne peut pas être dirigé vers un obstacle, comme il a été possible avec la méthode VFH originale. Enfin, en appliquant une fonction de coût pour la sélection de la direction, la performance de l'algorithme d'évitement l'obstacle devient plus efficace et plus fiable. La fonction de coût donne aussi la possibilité de changer entre les comportements en modifiant simplement la fonction de coût ou ses paramètres.

Le problème ; qui reste; de la méthode VFH + est sa nature locale, ce qui amène parfois le robot mobile en impasses qui pourraient être évitées. Pour surmonter ce problème, nous allons combiner la méthode d'évitement d'obstacle VFH+ avec un planificateur global de type propagation de vagues (*wavefront*) qui calcule un chemin jusqu'au but.

4.5.4 L'architecture d'intégration

L'architecture intègre les modules en tenant compte des limitations et restrictions imposées par la partie physique (capteurs et actionneurs) et la partie logique (ordinateurs) du robot. L'architecture a une configuration synchrone *planificateur - réacteur*, où les deux parties utilisent le modèle construit dans le temps d'exécution (Figure 4.11). Les fonctionnalités des modules sont la construction du modèle, le calcul du plan de mouvement (pour guider la méthode réactive), et la génération de commande de mouvement.

Dans certaines situations, les modules produisent des échecs qui sont gérés par l'architecture:

- **Exception dans le module de planification:** Parfois, le planificateur ne trouve pas une solution, soit parce qu'elle n'existe pas (par exemple lorsque l'objectif tombe sur un obstacle) ou parce que le module prend un temps trop long. Un chemin peut ne pas exister dans un environnement inconnu lorsque des buts à atteindre sont placés pour l'exploration et l'un d'eux tombe sur un obstacle, dans un environnement dynamique où un obstacle dynamique se déplace, ou s'arrête sur le but, dans des environnements statiques où le but est déplacé en raison de la dérive du robot, ou lorsque le but ou le robot sont entourés par des obstacles. Bien que ces situations pourraient être évitées en déplaçant la position du but, mais ce n'est pas le rôle du système de navigation. Les conséquences de cette décision pourraient affecter le développement normal de la tâche robotique en général.

- **Exception dans le module réactif:** le robot est complètement entouré par des obstacles lorsqu'il n'y a pas de zones de mouvement, et il ne peut pas progresser.

Dans les deux cas, on peut utiliser des stratégies qui ferment le contrôle en boucle (pour éviter les états morts). Dans le premier cas, le module réactif utilise directement la position du but à la place de l'information du planificateur. Dans le second cas, le robot s'arrête et tourne sur lui-même (ce comportement permet la mise à jour du modèle dans toutes les directions pour la recherche d'un nouveau chemin).

Les modules sont exécutés suivant la séquence Modèle - Planificateur – Réacteur (Figure 4.12). Le flot de données entre modules est unidirectionnel, à partir du module de modélisation vers les modules planificateur et réactif et du module de planification vers le module réactif. Les modules assurent que leurs contraintes de temps sont en synchronie avec le temps capteur 0.20sec. Ceci est important pour éviter les incohérences dans le temps qui peuvent apparaître dans le cas des stratégies asynchrones (le modèle est utilisé pour la planification locale et l'évitement d'obstacles et doit être cohérent dans le temps avec les deux modules). Les temps peuvent être de 0.08, 0.08 et 0.04sec pour chaque module respectivement afin de garder le contrôle du mouvement en boucle fermée (Le temps d'exécution maximum est de 0.20sec).

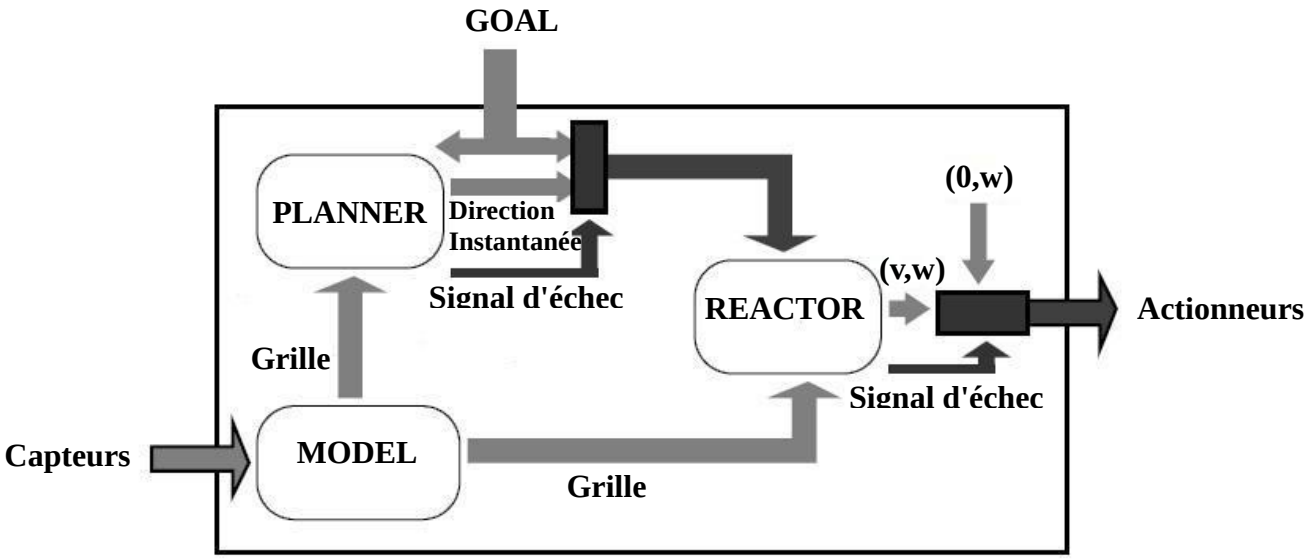


Fig.4.11 – *Cette figure montre l'architecture d'intégration. Les modules fonctionnent ensemble de façon synchrone et partagent certaines données. En outre, dans certaines circonstances les modules échouent, et les exceptions sont lancées afin de gérer la situation et de fermer la boucle de contrôle de mouvement.*

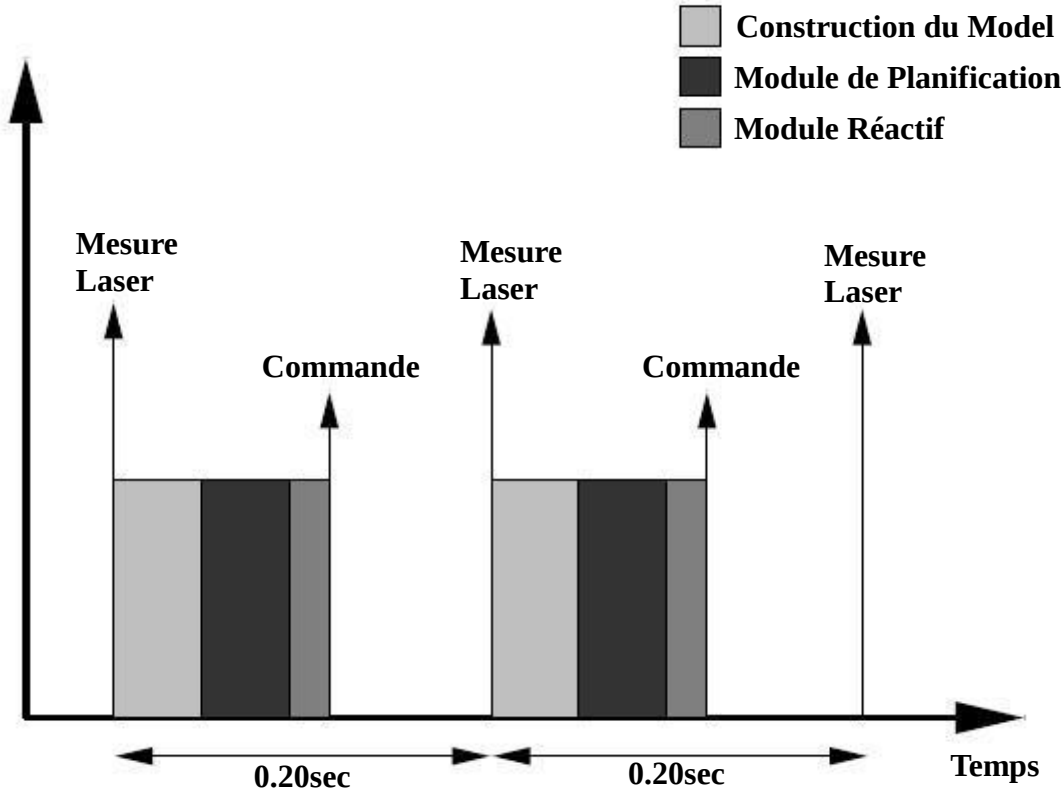


Fig.4.12 – *La figure représente le Diagramme du temps d'exécution des modules, qui fonctionnent de manière synchrone dans le cycle du capteur. Une fois la nouvelle mesure laser est disponible, les trois modules sont exécutés séquentiellement avant l'arrivée de la prochaine mesure du capteur.*

4.6 Conclusion

Pour conclure, la structure modulaire du système permet de remplacer les différents modules facilement car les aspects informatiques et les interfaces entre les fonctionnalités sont clairement spécifies (Exigence de *l'intégration modulaire* spécifiée dans la section 3). Nous avons examiné la façon dont chaque module est conforme aux exigences proposées. Dans le chapitre suivant nous montrons comment aussi ils sont conformes lors de l'intégration.

Chapitre 5

Expérimentation et Résultats de la Simulation

5.1 Introduction

Les premiers travaux de notre recherche ont été d'identifier une expérience à réaliser et d'identifier le simulateur à utiliser pour cette expérience. Nous avons étudié plusieurs plateformes [60] de simulation et nous nous sommes limités à deux d'entre elles (Player/Stage/Gazebo et CARMEN) utilisées dans de nombreux articles. Ces simulateurs sont aujourd'hui des projets GNU, initialement développés par une collaboration internationale des chercheurs en robotique. Ils permettent d'écrire des programmes et d'exécuter ces derniers en simulation ou sur des plateformes réelles. Après évaluation des capacités de ces deux plateformes, nous avons choisi Player/Stage/Gazebo dans sa version 2D (Player/Stage) pour sa souplesse d'utilisation dans le contexte multi-robot. Ce simulateur nous permet de reproduire des expériences concrètes d'exploration d'un environnement dynamique. Le simulateur nous permet aussi de planifier des trajectoires pour un robot mobile dans un environnement référencé capteurs.

Nous considérons comme point de départ un environnement partiellement connu, dans lequel il s'agit de réaliser une mission robotique. La question de la planification robotique peut se résumer à l'affectation d'une cible au robot mobile afin de minimiser le temps global de la mission.
L'état de l'art de ce problème propose la prise en compte de la structure de l'environnement (partiellement connu) pour définir des objectifs potentiels.
Ce chapitre est organisé comme suit: Une présentation du logiciel de simulation (section 2), les résultats des expériences réalisées (section 3), Une conclusion (section 4).

5.2 Simulateur Player/Stage

Player/Stage est un projet qui permet la simulation de robot sur un ordinateur avec le célèbre trio Player, Stage et Gazebo. Ce simulateur créé par les chercheurs de l'Université USC (University of Southern California)[61]. Il permet de simuler un très grand nombre de matériels y compris les caméras. La dynamique des robots peut être aussi simulée et le simulateur dispose de deux interfaces de visualisation: 2D et 3D. (*Figure 5.1, Figure 5.2*)

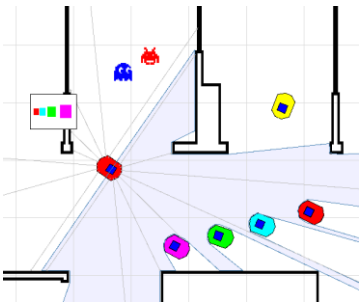


Fig.5.1 – *L'interface de Stage en 2D*

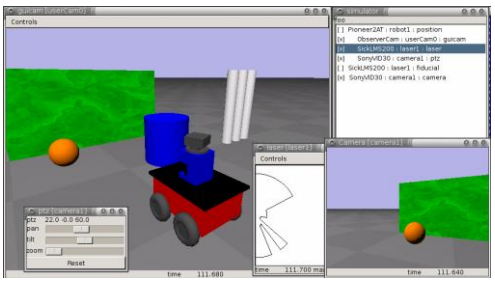


Fig.5.2 – *L'interface de Gazebo en 3D*

Player/Stage/Gazebo est un simulateur robotique avancé et une plate-forme d'interface. Elle donne à la fois des outils pour simuler, ainsi que pour communiquer avec un système robotique. Dans le modèle du simulateur 'Player/Stage/Gazebo', il y a trois composants principaux. Ce sont Player, Stage et Gazebo. Player est un serveur qui fournit une interface puissante et flexible à une variété de capteurs et d'actionneurs. Player fonctionne en créant plusieurs niveaux d'abstraction de hardware. Player permet de faciliter la programmation en comparaison avec la programmation robotique. Dans la programmation robotique (Figure 5.3), le programmeur doit être responsable à écrire un module pour retirer des données des capteurs, un module pour traiter des données obtenues, un module pour générer des commandes de contrôle des moteurs.

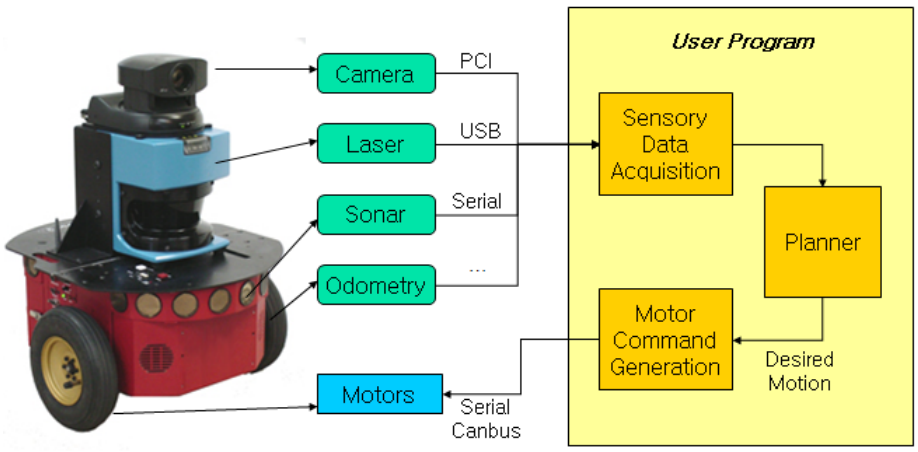


Fig.5.3 – *Le modèle de la programmation robotique*

Dans le modèle de la programmation par Player, on doit écrire seulement le module pour traiter des données obtenues par Player et donner des commandes au Player. Le module pour retirer des données des capteurs et le module pour générer des commandes de moteurs sont réalisés par Player.

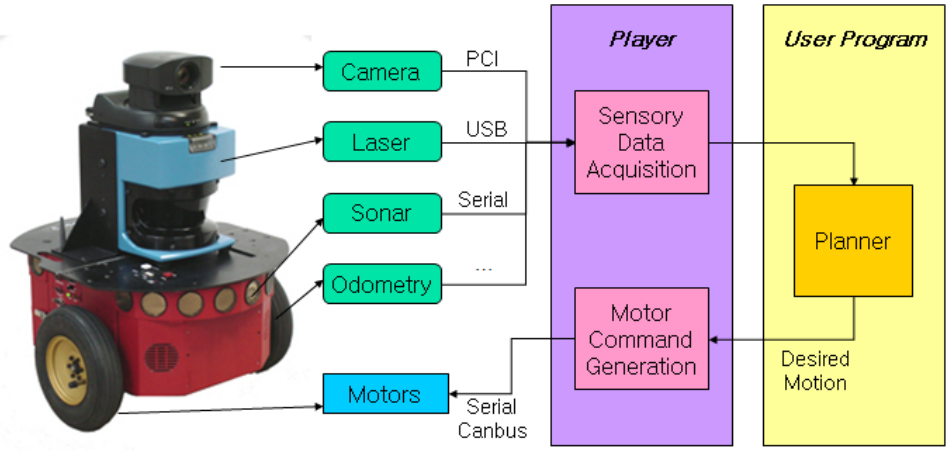


Fig.5.4 – *Le modèle de la programmation par Player*

Player fonctionne sous le modèle de Client/Server. Ce modèle a plusieurs avantages :

- Distribution : Ce modèle permet à un client d'accéder à des capteurs et à des actionneurs n'importe où dans le réseau. Un client peut accéder à plusieurs serveurs et un serveur permet à plusieurs clients d'accéder. Cela permet à un programme de contrôler le comportement de plusieurs robots et plusieurs programmes peuvent contrôler des aspects différents d'un seul robot.
- Indépendance : Des programmes de client peuvent être écrit dans n'importe quel langage pour n'importe quelle plate-forme de hardware qui soutient la programmation par le socket.
- Commodité : Le serveur fournit une *abstract interface* à des périphériques (devices). Le programme de client s'abonne (subscribe) à un ensemble de périphériques (devices) par une fréquence pour retirer des données. Les données des périphériques abonnés arrivent comme des paquets de données et à période de temps demandé.

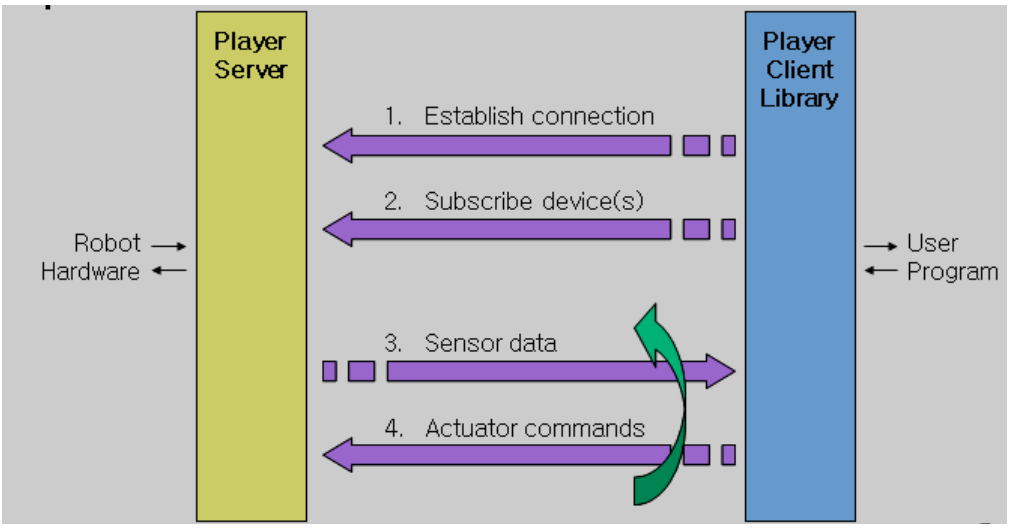


Fig.5.5 – *Le modèle Client/Server de Player*

Stage simule des robots mobiles, des capteurs et des objets en 2D. Gazebo simule des robots mobiles, des capteurs et des objets en 3D.

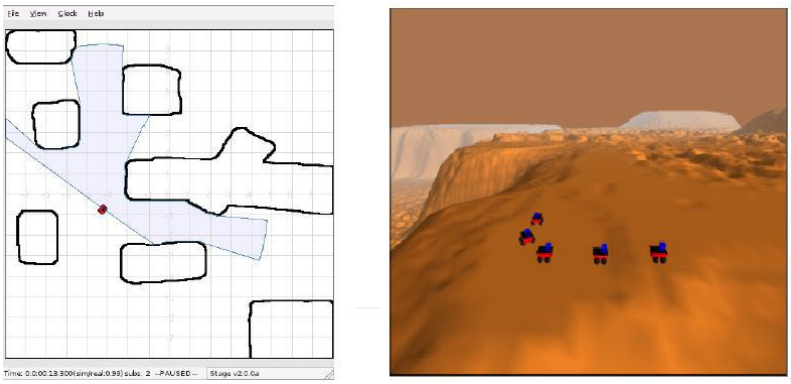


Fig.5.6 – *Le simulateur Stage et Gazebo*

5.2.1 Player

Player est en réalité une interface réseau de type client/serveur qui permet de réaliser la simulation de robots dans leur environnement. Il donne une possibilité d'appliquer un algorithme d'intelligence avec les trois langages proposés (Java, C et C++).

Il permet de recueillir et de traiter les informations portées par les différents capteurs que possède chacun des robots de la simulation. Et il permet de contrôler les robots via les ports (le port par défaut est 6665), alors chaque robot est contrôlé par un port différent. Le modèle client/serveur utilisé permet en outre de partager les données de la simulation en cours avec n'importe quel ordinateur mis en réseau avec celui qui est le siège de la simulation, ce qui, pour une équipe de recherche, peut se révéler particulièrement efficace et utile.

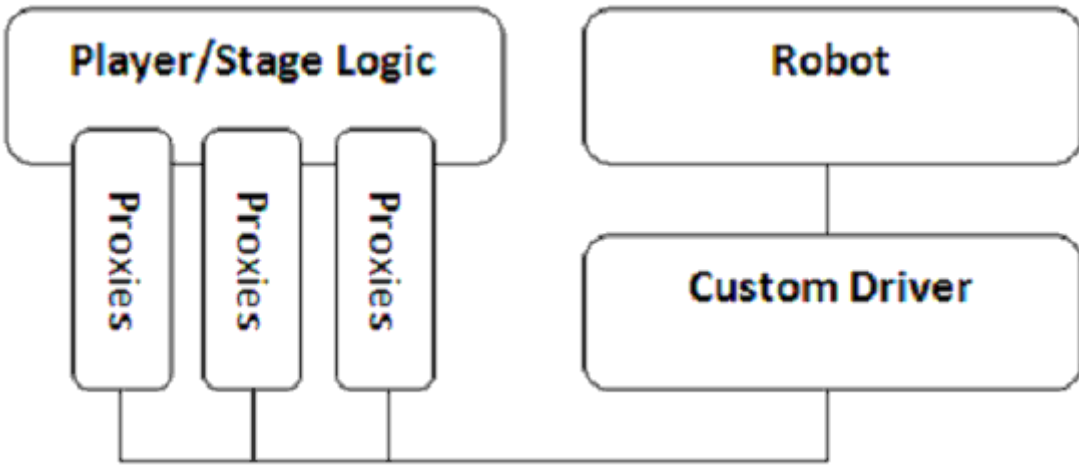


Fig.5.7 – *Le mécanisme de communication entre Player/Stage et le contrôleur de robot*

Player fonctionne par création de couches abstraites. Il cache les implémentations de bas niveau. Il fournit des proxies pour communiquer avec les robots en haut niveau. Player crée les communications et gère ces communications via TCP/IP.

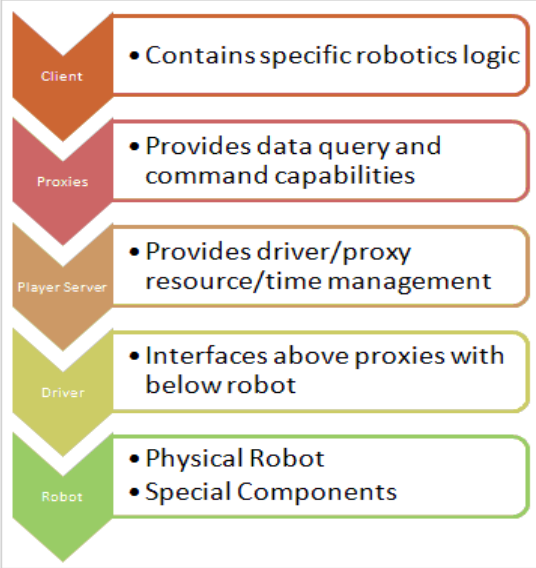


Fig.5.8 – *Le mécanisme de communication entre le programme client et le robot*

Un driver de Player permet de communiquer entre les matériels du robot et le code de client. Un driver a les méthodes pour communiquer aux proxies. Un proxy est un objet comme laser, sonar....

5.2.2 Stage

Stage est comme un driver de Player. Il fournit une interface graphique à la simulation en 2D, modélisant ainsi les déplacements des différents robots. À la vue de point des programmeurs, on peut exactement utiliser de même mécanisme pour communiquer entre Player et les robots réels. La différence est uniquement dans le fichier de configuration. Avec Stage, pour simuler les robots, on utilise le driver Stage avec plugin *libstageplugin*. Quand on travaille avec Stage, on s'intéresse au fichier de configuration. Ce qui décrit le model des robots et l'environnement de simulation.

5.2.3 L'architecture de Player/Stage

Avec Player/Stage, il y a trois concepts principaux:

- **Interfaces :** Une spécification sur comment interagir avec une certaine classe de matériel comme un type de senseurs ou d'actionneurs particuliers. Les interfaces définissent la syntaxe pour émettre des commandes pour les actionneurs, ainsi que pour lire les messages provenant des capteurs.
- **Driver :** Le logiciel permettant de communiquer avec le matériel et permettant de traduire les entrées/sorties de celui-ci conformément à une ou plusieurs interfaces.
- **Device**

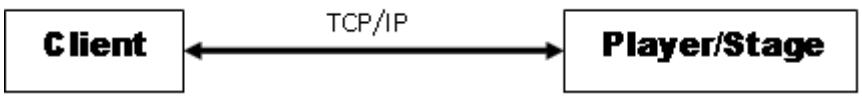


Fig.5.9 – *Communication par une Socket TCP/IP*

Player/Stage fonctionne sur un model de client/serveur en échangeant les messages. Les messages arrivent et le serveur les traite, puis envoie les messages correspondants. Il y a trois types de messages:

- Les messages des données
- Les messages de la commande
- Les messages de configuration

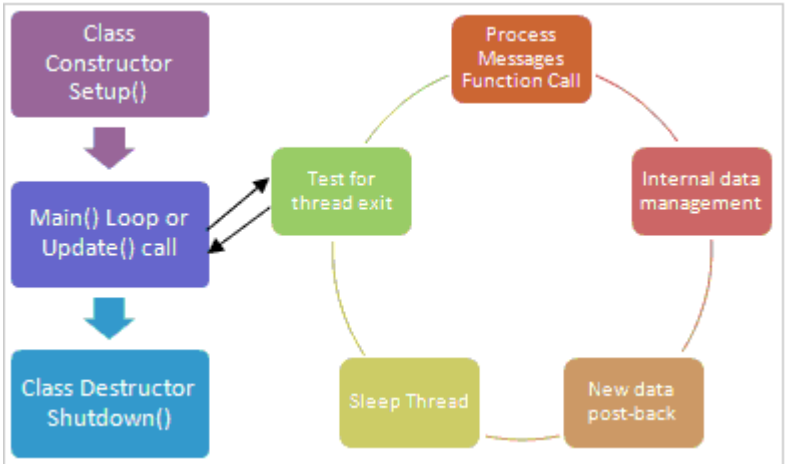


Fig.5.10 – *Le processus de Player*

Player offre la possibilité de simuler des robots, en leur appliquant des algorithmes de déplacement et de détection accompagnés de capteurs, et en permettant de suivre les résultats obtenus, tandis que Stage permet d'avoir une vision concrète de la simulation.

Dans la section suivante, nous exposerons les résultats de nos expériences.

5.3 Expérimentations et résultats de la simulation

Dans cette section, nous discutons quelques expériences réalisées avec notre système de navigation à base d'un capteur laser en utilisant l'environnement de simulation *Stage*.

Model du robot utilisé:

Fig.5.11 – ***Le robot Pioneer 2 DX***

Pour toutes les expériences le robot Pioneer 2 DX d'Activemedia est utilisé (figure 5.11). Le robot Pioneer 2 DX est équipé de deux roues motrices et se déplace à une vitesse qui peut atteindre 1.6 m/s et une charge utile de 20 Kg. Le robot est contrôlé à partir d'un PC à distance, en cours d'exécution sous Linux ou MS Windows, qui communique avec Pioneer par radio-modem à la vitesse de 9600 bauds. Il est équipé à bord d'un microcontrôleur Siemens 88C166 programmable qui gère les tâches de bas niveau, tels que le contrôle du moteur et capteurs.

Facilité d'utilisation et flexibilité:

Le robot est livré avec P2OS^{TM}, Un petit système d'exploitation pour le microcontrôleur à bord qui rend la programmation très facile des tâches de niveau supérieur, telles que la planification de chemin, la construction de cartes et la localisation. Chacune de ces tâches est effectuée sur l'ordinateur à distance et peut être facilement mise en œuvre dans le langage C / C + +.

Applications:

Avec sa grande vitesse (1,6 m / sec) et sa puissance de manoeuvre, Pioneer 2 DX peut accomplir beaucoup de tâches, en particulier dans les environnements intérieurs. Par exemple, il peut être utilisé pour la distribution du courrier entre les différents bureaux et ainsi de suite. Toutefois, il est surtout utilisé comme une plate-forme de test pour les algorithmes de navigation autonome et la planification de chemin.

Environnements utilisés:

Pour tous les environnements de simulation:

- Une fenêtre de simulation de taille [800 800] pixels avec un centre de coordonnées (0,0), une échelle de 0.025 et une résolution de 0.02
- Un robot de type Pioneer 2 DX équipé d'un capteur Laser Sick LMS-200

Paramètres de la méthode réactive VFH:

- Distance de sécurité: safety_dist_0ms = 0.1 m → Distance minimale à un obstacle lorsque le robot est en état d'arrêt.
- Distance de sécurité: safety_dist_1ms = 0.05 m → Distance minimale à un obstacle lorsque le robot se déplace à une vitesse de 1 m/s.
- Vitesse maximale: max_speed = 0.2 m/s
- Acceleration maximale: max_acceleration = 0.2 m/s/s
- Vitesse de rotation: min_turnrate = 10 deg/s → Vitesse minimale.
- Vitesse de rotation: max_turnrate_0ms = 70 deg/s → Vitesse de rotation maximale lorsque le robot est en état d'arrêt.
- Vitesse de rotation: max_turnrate_1ms = 70 deg/s → Vitesse de rotation maximale lorsque le robot se déplace à une vitesse de 1 m/s.
- Distance epsilon: dist_epsilon = 0.3 m → Distance au but considérée comme acceptable.
- Angle epsilon: angle_epsilon = 10 deg → Différence d'angle avec le but considérée comme acceptable.
- Coût d'orientation: weight_current_dir = 3 → Poids pour que le robot continue dans la direction actuelle.
- Coût d'orientation: weight_disired_dir = 5 → Poids pour que le robot aille dans la direction du but.

Paramètres pour le planificateur global "wavefront":

- Distance de sécurité: safety_dist = 0.01 m
- Rayon robot: robot_radius = O.1 m
- Rayon maximal: max_radius = 0.3 m → Les cellules avec une distance aux obstacles supérieure à celle-ci peuvent être prises pour la planification du chemin au but.
- Distance epsilon: dist_epsilon = 0.5
- Angle epsilon: angle_epsilon = 10 deg
- Distance de replanification: replan_dist_thresh = 2 m → changement dans la position du robot pour faire une replanification.

- Temps de replanification minimal: replan_min_time = 2 s → Temps minimal entre deux replanifications.
- Points de route: add_rotationl_waypoints = 1 → Mettre un waypoint si la différence entre la direction actuelle et le prochain waypoint est supérieure à 45°.

Pour les expériences 1, 3, 4 l'environnement utilisé est de type appartement de six chambres avec les paramètres suivants:

- Taille de l'image [16 10] cellule.
- Résolution de l'image simple_rooms.png est de 0.123333

Pour les expériences 2 et 5:

- Taille de l'image [16 16] cellule.
- Résolution de l'image cave.png est de 0.123333

Dans ce qui suit nous présentons les expériences réalisées afin de valider notre approche. Nous avons mené ces expériences pour résoudre deux types de problème:

(1) Evitement de situations de pièges et comportements cycliques.

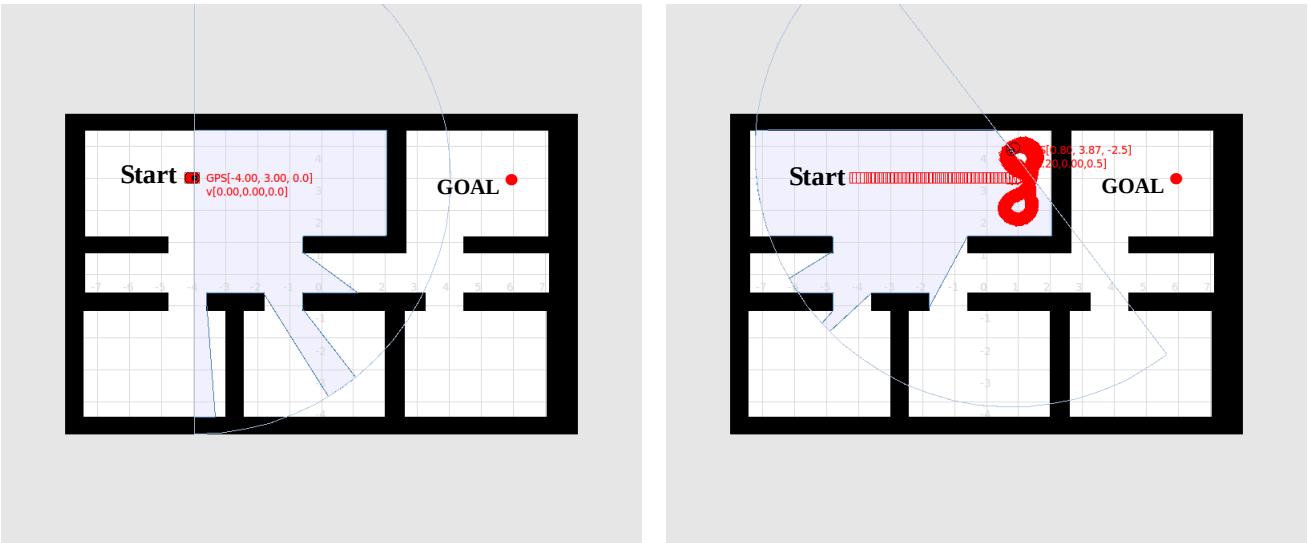
(2) Evitement d'obstacles et passages étroits dans un environnement dynamique

5.3.1 Evitement de situations de pièges et comportements cycliques

Dans cette expérience, le système de navigation résout plusieurs situations de piège et évite les comportements cycliques produits par la méthode réactive.

Experience1: Evitement d'une situation de piège sous forme de U (Minimum local)

La figure 5.12 montre une situation de piège sous forme de U dans laquelle la méthode réactive ne permet pas au robot de s'en sortir.



(a) **(b)**

Fig.5.12 – ***Situation de piège sous forme de* U *(minimum local)***

Dans cette situation très connue dans le monde de la robotique mobile et qui constitue un des points faibles des méthodes purement réactives, notre méthode réactive utilisée **VFH** échoue dans la mission d'atteindre le but et le robot se trouve bloqué infiniment dans un comportement cyclique comme l'indique la figure 5.12b (fenêtre du simulateur *stage*).

Le caractère local de la méthode lui confère un inconvénient majeur: les " minima locaux " de la fonction de sélection. En effet, un coût minimal peut être attribué à un déplacement répondant localement aux critères de choix désirés, mais qui, dans le futur, conduira le robot à une situation de blocage ou à une collision (figure 5.12b).

La figure 5.13 montre la solution du problème précèdent par notre approche hybride composée d'un planificateur globale qui utilise une fonction de navigation NF1 et une méthode réactive(VFH).

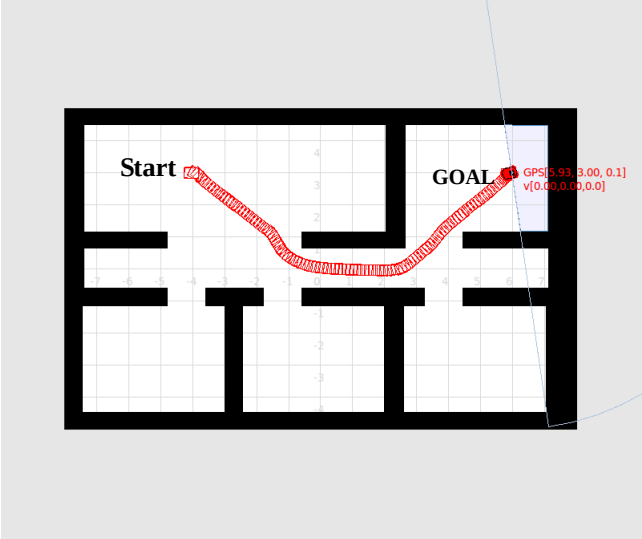


Fig.5.13 – ***Evitement de la situation de piège sous forme de U (minimum local) par notre approche hybride***

Les résultats donnés par le simulateur stage montrent clairement comment le robot a pu éviter la situation de piège en exécutant la trajectoire générée par le planificateur et en évitant les obstacles (obstacles fixes dans le cas de la figure 5.13). La méthode réactive a pu éviter les obstacles et plusieurs replanifications ont été faites pour remettre le robot sur son chemin vers le but comme le montre la figure 5.14.

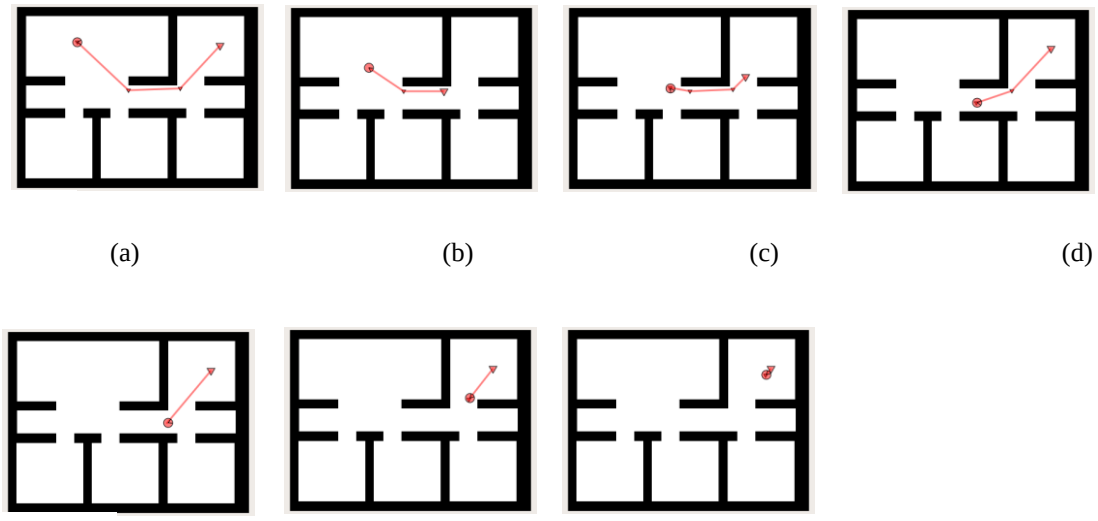


Fig.5.14 – ***Les différents plans générés lors de l'exécution de la trajectoire***

La figure 5.14 montre les différents plans générés lors de l'exécution de la trajectoire par la méthode locale. Lorsque le robot se trouve plus proche d'un obstacle ou s'éloigne de la trajectoire en exécution un autre chemin est planifié pour le remettre sur sa trajectoire vers le but.

Les plans calculés correspondants récupérés dans la fenêtre du simulateur sont les suivants:

new goal: 5.930000, 3.000000, 0.100000

computed global path: 0.058335

computed local path: 0.011497

computed local path: 0.012492

computed local path: 0.017301

computed local path: 0.013537

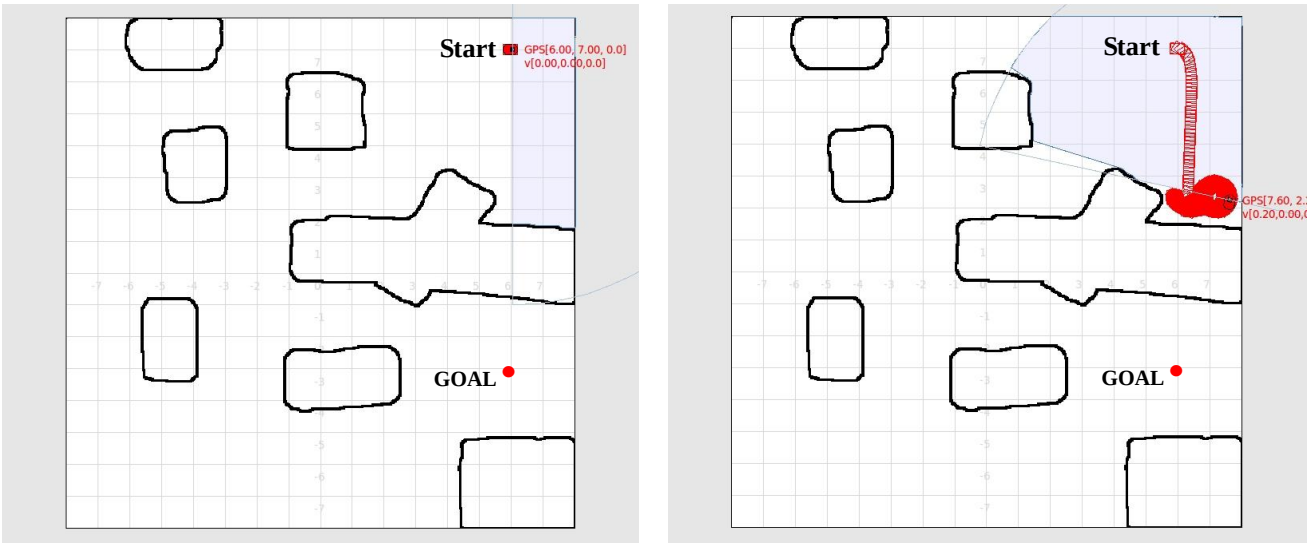
computed local path: 0.008414

computed local path: 0.004575

La première mesure concerne le chemin global, les autres mesures correspondent aux chemins générés localement.

Experience2: Evitement d'une situation de piège et comportement cyclique

La figure 5.15 montre une situation de piège qui ressemble à la situation précédente mais dans un environnement différent. En effet, la méthode réactive VFH dirige le robot directement vers le but qui se trouve en bas de l'obstacle mais le robot se trouve coincé infiniment dans un comportement cyclique et la méthode réactive échoue dans la sélection de la bonne direction qui permet d'éviter l'angle de l'obstacle qui gène le mouvement du robot.



(a) **(b)**

Fig.5.15 – ***Situation de piège et comportement cyclique***

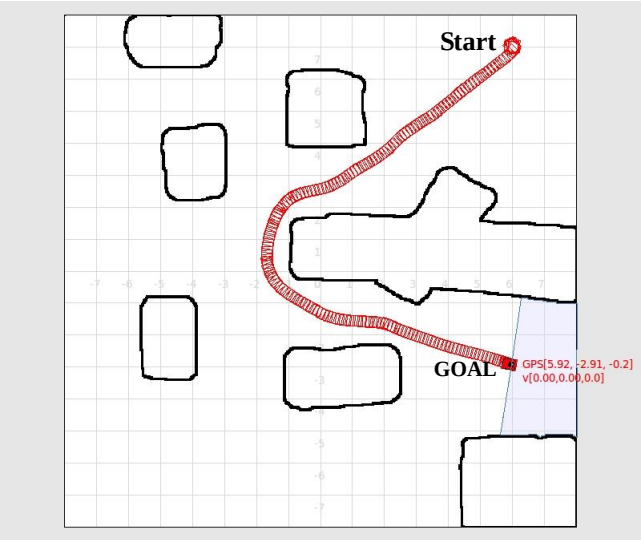
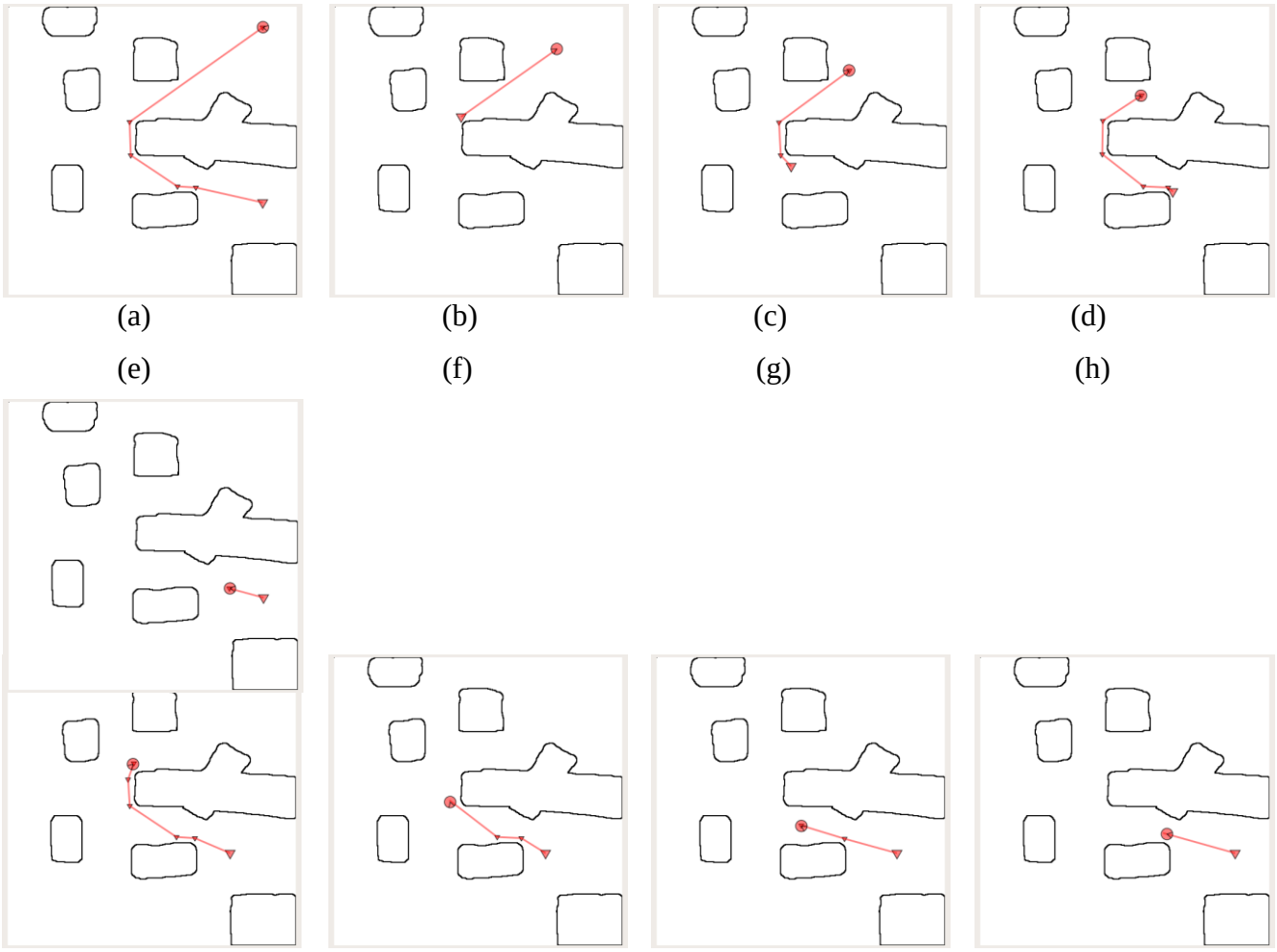


Fig.5.16 – ***Evitement de la situation de piège et comportement cyclique de la figure 5.15***



(a) (b) (c) (d)

(e) (f) (g) (h)

(i)

Fig.5.17 – ***Les différents plans générés lors de l'exécution de la trajectoire***

Les plans calculés correspondants récupérés dans la fenêtre du simulateur sont les suivants:
new goal: 5.730000, -2.700000, 2.034444

computed global path: 0.079180

computed local path: 0.032016

computed local path: 0.021466

computed local path: 0.038687

computed local path: 0.025611

computed local path: 0.039718

computed local path: 0.054837

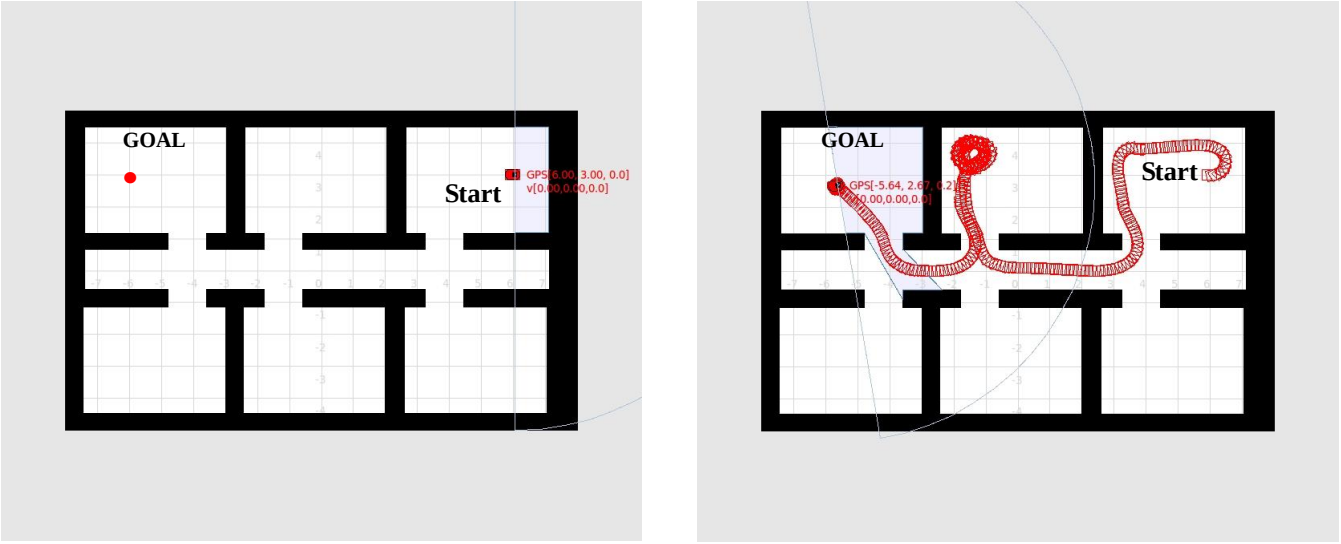
computed local path: 0.025148

computed local path: 0.017061

La figure 5.16 montre la trajectoire exécutée par le robot pour éviter la situation de la figure 5.15. En effet le robot a pu éviter la situation de piège en exécutant la trajectoire générée par le planificateur global tout en corrigeant les erreurs dans le mouvement généré et exécuté par la méthode réactive avec des replanifications des chemins locaux. La figure 5.17 montre 8 chemins locaux calculés pour remettre le robot dans la direction du but.

Experience3: Evitement d'un comportement cyclique et plusieurs portes

Dans cette expérience nous avons pu montrer un autre inconvénient de la méthode locale. En effet, malgré que le robot ait pu atteindre le but, il a trouvé des difficultés pour passer les portes et il est entré dans un comportement cyclique dans la deuxième chambre dans un environnement de type appartement. Après plusieurs cycles (Quatre dans notre cas) la fonction de sélection de la méthode réactive a pu trouvé une direction qui mène vers la sortie (figure 5.18b). Il faut noter que l'emplacement du but joue un rôle important dans le choix de la direction du mouvement par la méthode réactive. En effet, un emplacement vers le haut du but dans le cas de la figure 5.18 peut bloquer le robot dans un comportement cyclique infini dans la deuxième chambre.



(a) (b)

Fig.5.18 – ***Comportement cyclique et passages de portes***

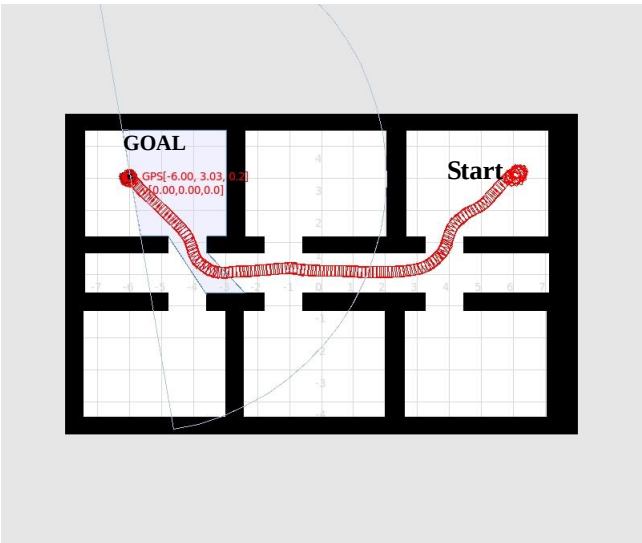
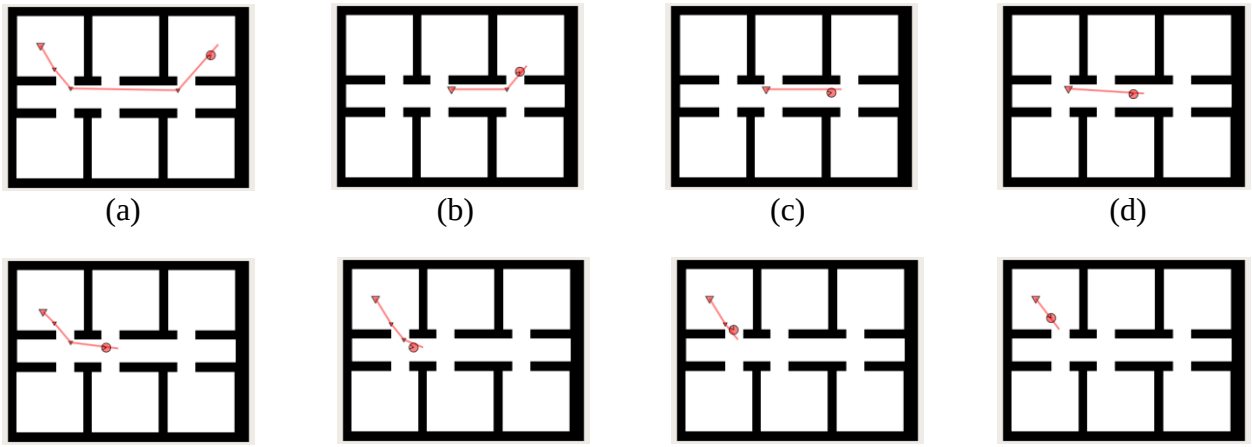


Fig.5.19 – ***Evitement du comportement cyclique et passages de portes de la figure 5.18***



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig.5.20 – ***Les différents plans générés lors de l'exécution de la trajectoire***

Les plans calculés correspondants récupérés dans la fenêtre du simulateur sont les suivants:

new goal: -5.840000, 3.440000, 0.000000

computed global path: 0.056819

computed local path: 0.010394

computed local path: 0.021615

computed local path: 0.012057

computed local path: 0.025442

computed local path: 0.024799

computed local path: 0.010723

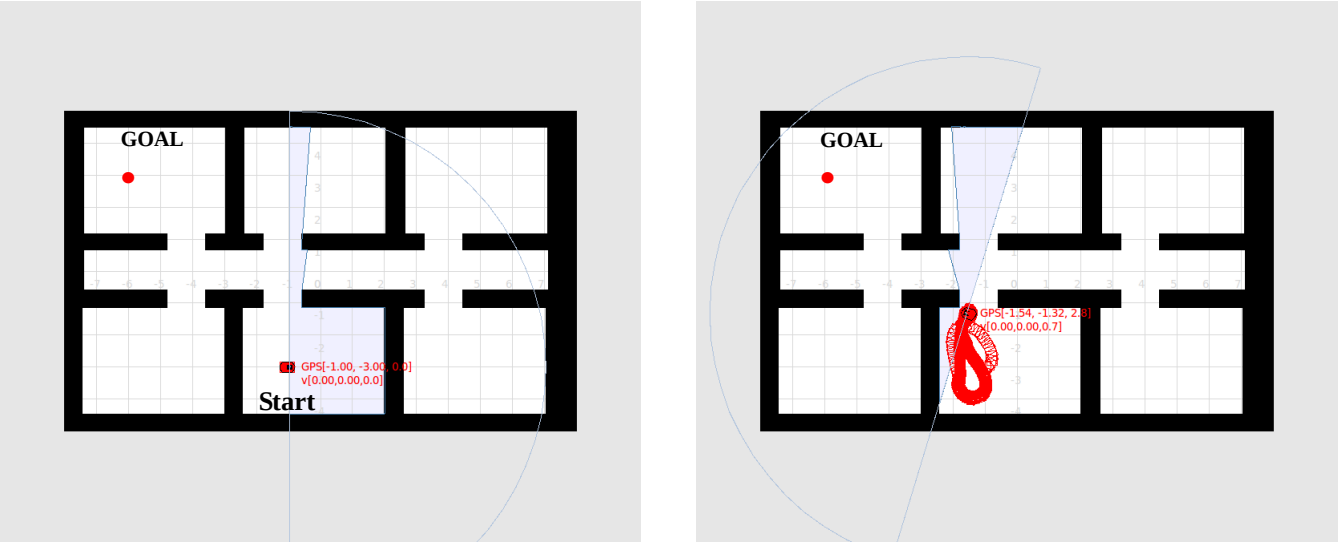
computed local path: 0.005875

La figure 5.19 montre comment le robot a pu éviter l'entrée dans la deuxième chambre et la longueur de la trajectoire a été réduite en suivant le chemin généré par le planificateur tout en faisant des replanifications chaque fois que le robot se trouve très proche d'un obstacle ou s'éloigne de la trajectoire globale comme le montre la figure 5.20. Cette expérience montre aussi comment le robot a pu atteindre le but dans un temps plus petit que celui de la méthode réactive.

Experience4: Solution du problème de passages et angles de portes

Dans cette expérience un autre inconvénient des méthodes réactives lors de passage des portes où le robot se trouve géné par l'angle comme le montre la figure 5.21b. En effet, la fonction de selection de la méthode reactive ne trouve pas la bonne direction pour faire sortir le robot qui se trouve bloqué par l'angle gauche de la porte et entre dans un comportement cyclique infini.

Cette situation de blocage a été évitée par notre approche hybride comme le montre la figure 5.22. Le robot passe par le milieu de la porte en suivant la trajectoire généré par le planificateur.



(a) (b)

Fig.5.21 – ***b) Le robot est gêné par l'angle gauche de la porte***

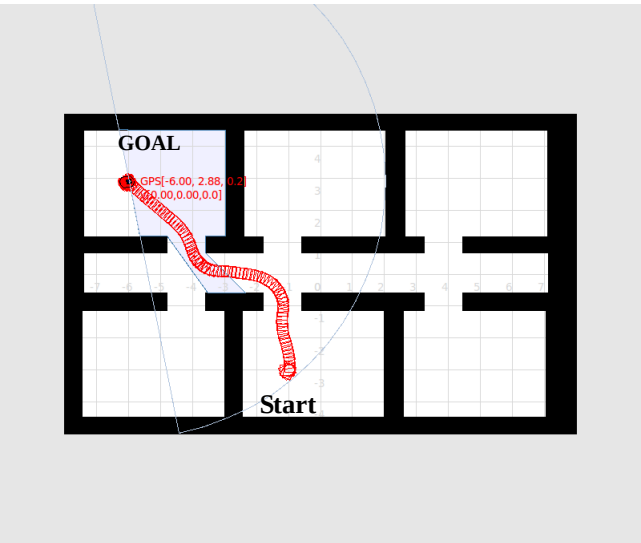
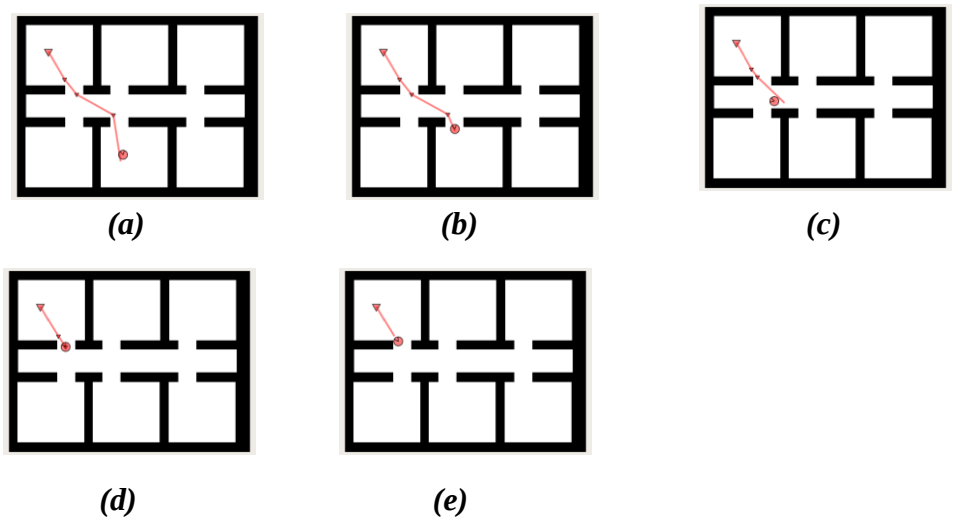


Fig.5.22 – ***Solution du problème de l'angle de porte par l'approche hybride***



(a) ***(b)*** ***(c)*** ***(d)*** ***(e)***

Fig.5.23 – ***Les différents plans générés lors de l'exécution de la trajectoire***

Les plans calculés correspondants récupérés dans la fenêtre du simulateur sont les suivants:

new goal: -5.920000, 3.640000, 0.000000

computed global path: 0.054481

computed local path: 0.016356

computed local path: 0.012569

computed local path: 0.009304

computed local path: 0.012945

La coopération entre les trois modules permet d'éviter le mouvement cyclique et les situations de piège.

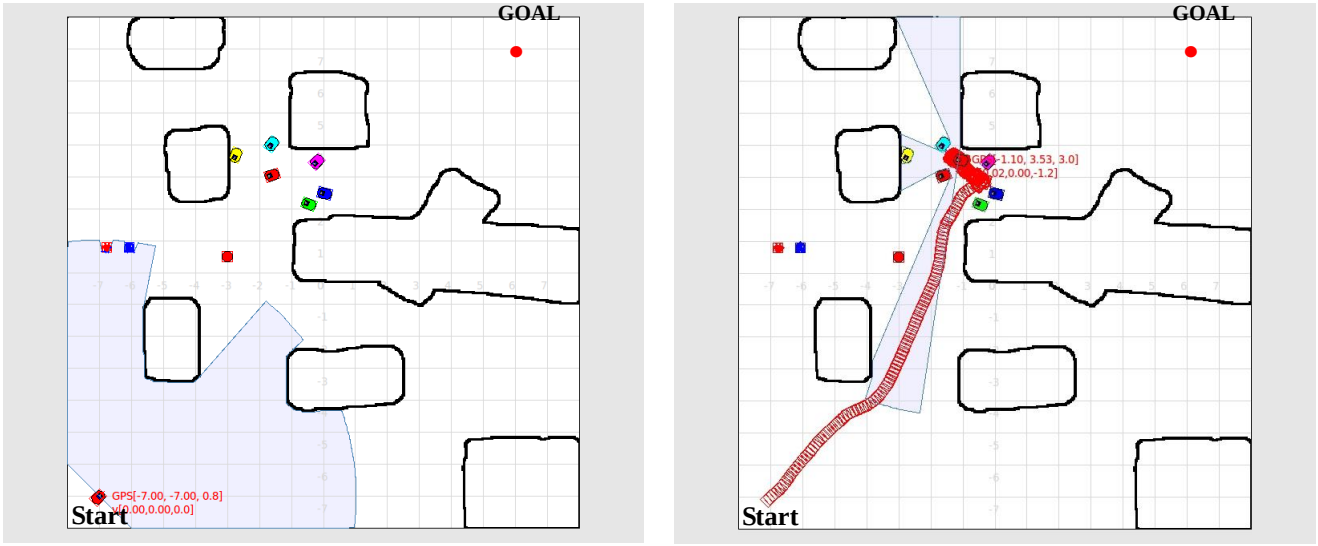
Premièrement, dans l'exécution de notre approche hybride nous n'avons pas rencontré de configurations d'obstacles qui produisent des situations de piège (quand un chemin existe au sein de la grille). C'est parce que la direction du mouvement calculé par le planificateur contient les informations nécessaires pour éviter ces situations (il faut noter que le mouvement est calculé par le module réactif).

Deuxièmement, les environnements symétriques ne produisent pas de mouvements cycliques puisque la direction du mouvement calculée par le planificateur fait la distinction entre les zones de mouvements possibles.

Pour conclure, le système a pu déplacer le robot dans des environnements différents en respectant les exigences pour un système de contrôle à base de capteur (voir section 3 du chapitre 4) en ce qui concerne la génération de *mouvements robustes* et *l'intégration de l'information*. *Les mouvements cycliques et les situations de piège ont également été évités.*

5.3.2 Evitement d'obstacles et passages étroits dans un environnement dynamique

Experience5: Le robot est entouré de plusieurs obstacles dynamiques (robots et autres objets)



(a) **(b)**

(c)

Fig.5.24 – ***b) Le robot est entouré de plusieurs robots et reste bloqué***

c) Le robot trouve un chemin vers le but après le déplacement du robot le plus haut vers la gauche

Cette expérience met en évidence notre approche hybride dans un environnement difficile (dynamique, complexe et très dense), où le robot navigue le long d'un passage très étroit avec des robots et objets distribués au hasard (Figure 4.24). La difficulté réside surtout dans le passage étroit entre les deux robots les plus à droite et compte tenu des distances de sécurité et de la dynamique du robot qui ne permettent pas au robot de suivre la trajectoire calculée par le planificateur, celui-ci se trouve entouré de plusieurs obstacles où il y a peu d'espace de manœuvre.

[Dans certains endroits le robot se trouve à une distance plus petite des obstacles distribués de chaque côté du robot (Figure 5.24b)]. Le robot entre dans un comportement cyclique infini.
La figure 5.24c montre comment le robot; après le déplacement du robot le plus haut vers la gauche; a pu atteindre le but en suivant un autre chemin après plusieurs replanifications (Figure 5.25). En effet, le déplacement du robot le plus haut vers la gauche laisse suffisamment d'espace au robot pour sortir de la situation de piège.

Les principaux aspects de cette expérience sont les performances individuelles et la coopération entre le module de modélisation et le module réactif. Les scans laser dans le module de modélisation ont été rapidement intégrés dans la grille. Ainsi, la méthode réactive évite les nouveaux obstacles dès qu'ils ont été perçus. Cette réactivité est essentielle pour déplacer le robot dans des environnements denses (car les retards dans le calcul du mouvement peuvent conduire à des collisions).

Deuxièmement, les dernières mesures sensorielles restées dans la grille ont été utilisées par le module réactif pour la tâche d'évitement. Cela a été important car il arrive que le capteur ne perçoive pas les obstacles les plus proches en raison de contraintes de visibilité. Toutefois, étant donné qu'ils étaient perçus à quelques instants avant, ils sont restés dans la grille et ont été évités (Figure 5.24).

Cette expérience a été particulièrement difficile du point de vue de la génération du mouvement en raison de l'espace étroit, car la plupart des techniques existantes ont des limitations intrinsèques qui pénalisent le calcul du mouvement en vertu de ces circonstances. Toutefois, la méthode VFH conduisait le robot sans collisions, même au milieu des obstacles très proches. Le mouvement est libre d'oscillations (voir le chemin exécuté dans la figure 5.24c après déplacement du robot le plus haut vers la gauche) et évite tous les pièges qui peuvent être crées en raison de la densité d'obstacles. Les exigences importantes respectées ici sont la *génération de mouvement robuste* et *l'intégration de l'information*.

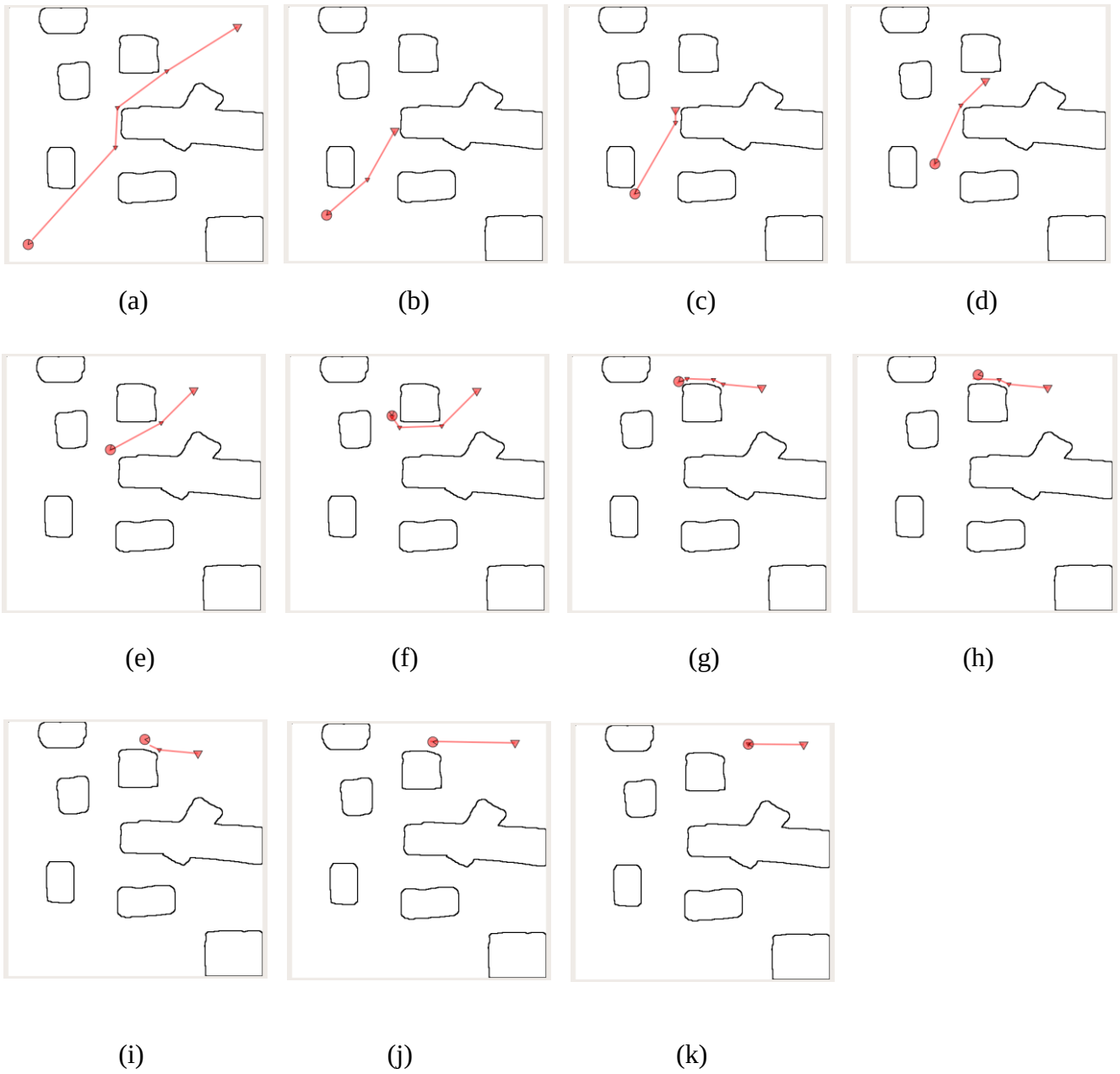


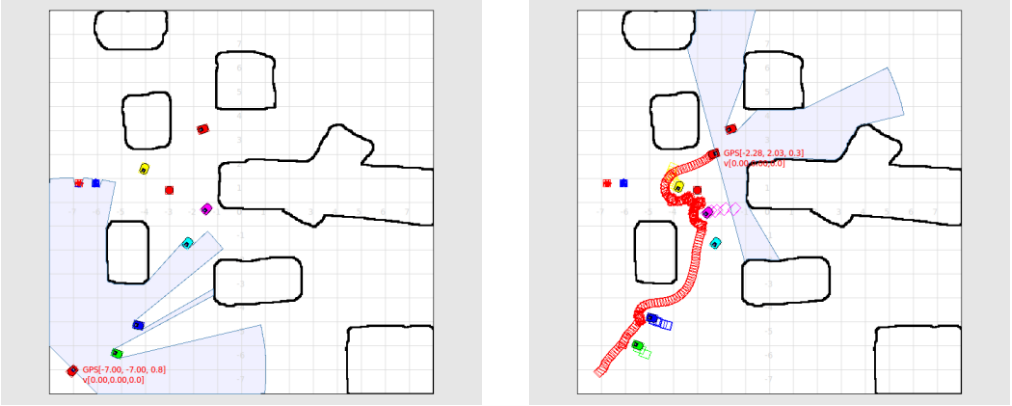
Fig.5.25 – ***Les différents plans générés lors de l'exécution de la trajectoire***

Les plans calculés correspondants récupérés dans la fenêtre du simulateur sont les suivants:

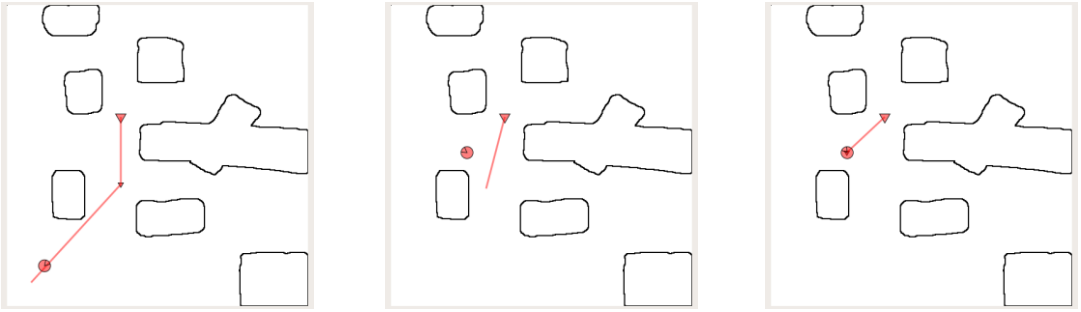
new goal: 5.970000, 6.360000, -2.677945
computed global path: 0.074612
computed local path: 0.019442
computed local path: 0.027036
computed local path: 0.030754
computed local path: 0.028488
computed local path: 0.025985
computed local path: 0.024023
computed local path: 0.017073
computed local path: 0.015287
computed local path: 0.018491
computed local path: 0.015216

Experience6: évitement d'obstacles mobiles.

Dans cette expérience qui complète l'expérience précédente, nous montrons comment le robot a pu éviter des obstacles mobiles tout en respectant les exigences pour un système de contrôle à base de capteur. (Figure 5.26, Figure 5.27)



(a) **(b)**



(c) **(d)** **(e)**

Fig.5.26 – ***b) Le robot évite d'autres robots (obstacles mobiles)***
c) plan global généré d)e) plans locaux

Les plans calculés correspondants récupérés dans la fenêtre du simulateur sont les suivants:

new goal: -1.830000, 1.920000, 0.000000

computed global path: 0.078582

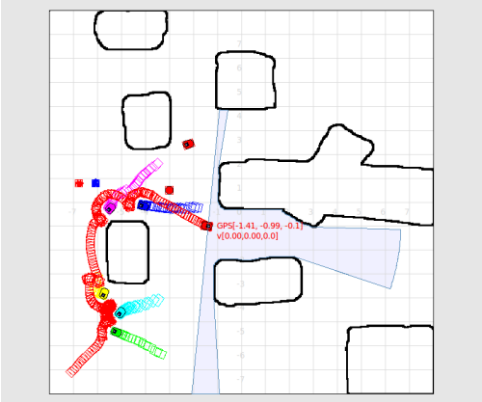
computed local path: 0.021080

computed local path: 0.026373

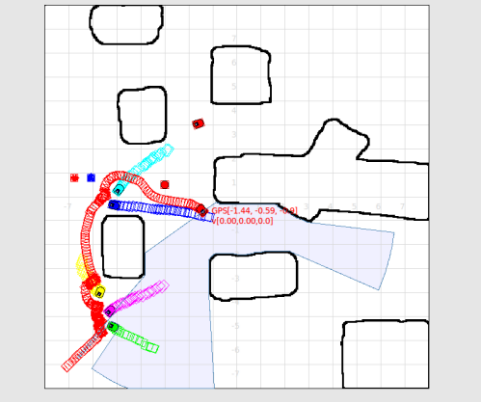
computed local path: 0.038802

computed local path: 0.037469

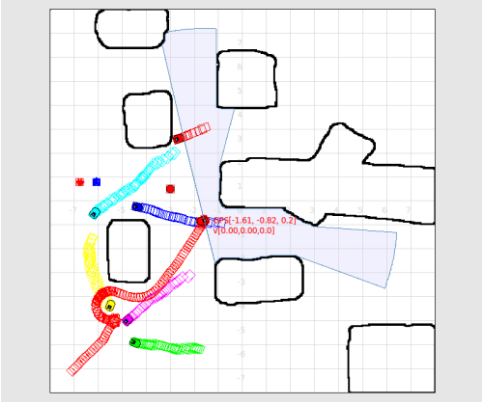
computed local path: 0.036209



(a)



(b)



(c)

Fig.5.27 – ***Evitement d'obstacles mobiles***

Discussion et comparaison avec d'autres méthodes

Notre approche à base d'un capteur Laser a été conçue suivant une architecture hybride à trois couches (Modélisation, Planification et Réaction). L'interaction entre modules est une configuration synchrone Planificateur-Réacteur où le planificateur calcule les informations nécessaires pour diriger la réactivité. Notre approche diffère des autres approches utilisées dans l'architecture d'intégration et dans les techniques utilisées dans chaque module. Nous discutons ici des conceptions alternatives à notre approche hybride qui n'utilisent pas le module réactif pour faire une planification "any-time", et l'impact des techniques utilisées pour mettre en œuvre chaque module en comparaison avec d'autres systèmes hybrides.

Comme nous l'avons mentionné dans la section 2 du chapitre 4, les systèmes hybrides tentent de combiner les capacités de planification avec la meilleure réactivité, robustesse, adaptation et flexibilité. Ces systèmes combinent un planificateur (délibération) et un réacteur (exécution). La stratégie commune consiste à calculer un chemin et l'utiliser pour diriger le module réactif. Une autre solution

architecturale alternative aux systèmes hybrides est celle fondée sur deux modules, un pour construire le modèle et l'autre pour planifier (conduire le robot). Ils effectuent des planifications "any-time" au lieu d'utiliser le module réactif pour calculer le mouvement. Dans certains d'entre eux, la modélisation et la planification sont synchrones, Roy et Thrun (2002); Stachniss et Burgard (2002), alors que dans d'autres, l'étape de planification est un processus asynchrone qui ne s'exécute que dans les zones qui influence le progrès vers le but, Stenz et Hébert (1995), Murphy et al. (1996); Urmson et al. (2003). Ces systèmes n'ont pas les avantages d'un comportement réactif, tels que l'adaptation rapide aux changements et la flexibilité dans des circonstances imprévisibles. Tous ces systèmes utilisent des planificateurs qui calculent généralement des trajectoires qui longent les obstacles, ce qui semble en contradiction avec la tâche d'évitement. [Murphy et al.(1996) a présenté un planificateur qui permet une planification de trajectoire loin des obstacles]. Cependant, dans notre approche cette situation est gérée de façon naturelle par le module réactif. Une autre difficulté de la plupart des systèmes est la dépendance à une trajectoire menant au but, qui n'est pas toujours disponible dans des scénarios réalistes ce qui conduit à une défaillance dans le Système de mouvement.

En outre, ces systèmes ne sont pas compatibles avec l'idée qu'un robot ne doit pas être contrôlé directement par un planificateur, Gat (1998), et il serait difficile d'obtenir le temps de calcul pour les mettre en pratique, Fikes et Nilsson (1971).

Récemment, les deux systèmes (hybride et "any-time planning"), ont été regroupés par Ranganathan et Koenig (2003) en utilisant les deux paradigmes pour adapter l'opération à la progression du robot dans l'environnement. Cependant, les conclusions ci-dessus sont tirées uniquement lorsque le planificateur est utilisé pour conduire le robot.

Pour les systèmes hybrides, une autre question est l'impact des techniques utilisées pour implémenter le planificateur, le modèle et l'évitement d'obstacle. En ce qui concerne le planificateur, Ulrich et Borenstein (2000) utilisent une vérification avant d'exécuter l'algorithme réactif, Borenstein et Koren (1991b). Les situations de piège locales sont évitées par la méthode réactive avant que l'algorithme soit exécuté. La validité de cette stratégie dépend du nombre d'étapes précédentes (respect de la distance maximale). Cette solution est bien adaptée pour les robots avec des capacités limitées de calcul car de bons résultats de navigation sont obtenus en réduisant la distance maximale inspectée.

Notre approche utilise un planificateur global qui assure les informations nécessaires pour éviter les situations de piège. Notre modèle d'approche est similaire à Brock et Khatib (1999), Arras et al. (2002). Cependant, leur modèle représente un espace de configuration qui augmente avec la distance parcourue. L'avantage est qu'une connaissance globale est incorporée lorsque le robot progresse, et la fonction de navigation du planificateur n'est pas à recalculer tant que le scénario reste le même. Cependant, notre modèle représente une portion locale de l'espace de travail, donc il ne dépend pas de la distance parcourue. Autrement dit, la mémoire et le temps de calcul sont fixés, alors que dans les

autres méthodes, ils augmentent avec l'évolution du robot ce qui rend le modèle plus large. Dans notre modèle, l'information autour du robot est toujours disponible. Dans les autres méthodes elle dépend de la disponibilité de temps et de mémoire. En outre, notre modèle représente un espace de travail qui peut être rapidement mis à jour lorsque l'environnement change, ce qui n'est pas le cas dans les méthodes qui représentent l'espace de configuration.

Un autre aspect fondamental est le module réactif. Brock et Khatib (1999); Arras et al. (2002); Hebert et al. (1997); Ulrich et Borenstein (2000); Gat (1998) entre autres, sont des systèmes qui utilisent des planificateurs réactifs qui ont des problèmes de conduite dans un environnement dense, complexe et gênant. Ce qui n'est pas le cas dans notre modèle, car les limitations typiques d'autres méthodes sont évitées, comme les situations de piège locales, les oscillations dans des environnements denses, et l'impossibilité d'obtenir une direction de mouvement vers les obstacles ou vers les zones à grande densité d'obstacles. Il en résulte un mouvement sûr et robuste dans des environnements qui restent toujours gênant pour de nombreuses méthodes existantes. La conclusion, notre système est doté de meilleures capacités de manoeuvrabilité spécialement en environnements denses, complexes et gênants.

5.4 Conclusion

Notre approche a les avantages d'un comportement réactif, tels que l'adaptation rapide aux changements et la flexibilité dans des circonstances imprévisibles. Elle permet d'éviter les limitations typiques d'autres méthodes, comme les situations de piège locales, les oscillations dans des environnements denses, et l'impossibilité d'obtenir une direction de mouvement vers les obstacles ou vers les zones à grande densité d'obstacles. Il en résulte un mouvement sûr et robuste dans des environnements qui restent toujours gênant pour de nombreuses méthodes existantes. En conclusion, notre approche est dotée de meilleures capacités de manoeuvrabilité spécialement en environnements dynamiques, denses, complexes et gênants.

Conclusion et Perspectives Générales

Nous nous sommes intéressés, au cours de ce mémoire, aux concepts et outils algorithmiques permettant à un robot mobile de se déplacer de manière autonome vers un but, et plus particulièrement nous avons considéré ce problème pour le cas où l'environnement est dynamique et incertain.

Deux tâches sont généralement identifiées pour ce travail: le calcul préalable d'une trajectoire de référence à l'aide de méthodes de planification; puis l'exécution de cette trajectoire à l'aide d'une méthode réactive qui permet de prendre en compte les petits imprévus et de rattraper les erreurs de suivi.

Dans le cas présent, l'absence de connaissance complète a priori sur l'environnement ne permet pas de garantir la validité de la trajectoire sur un temps indéfini, et celle-ci doit être recalculée régulièrement. Or, la présence d'obstacles mobiles nécessite que le robot reste réactif, et par conséquent que ces calculs soient réalisés dans un temps court et borné. La dimension élevée des espaces manipulés ne le permet pas et seul un déplacement local peut généralement être recherché, sans certitude que le but pourra être atteint.

Nous avons présenté dans ce mémoire une approche hybride à base d'un système de contrôle de mouvement qui utilise un capteur laser et qui constitue une partie d'un système complet de navigation. Les principales contributions sont les aspects fonctionnels et informatiques des modules et leur intégration dans le système, en plus de la forte validation par simulation. En conséquence, ce système hybride est capable de déplacer le robot avec une grande robustesse dans des environnements dynamiques et difficiles.

Nous avons cherché, dans un premier temps, une méthode réactive pour l'évitement d'obstacles. Après étude des approches existantes, nous avons choisi la méthode VFH+ qui est le résultat de plusieurs améliorations par rapport à la méthode VFH originale. Tout d'abord, en utilisant un seuil, la trajectoire du robot devient plus fiable. Deuxièmement, la méthode VFH+ prend explicitement en compte la largeur du robot, et par conséquent, cette méthode peut être facilement implémentée sur des robots de tailles différentes. Cette amélioration élimine également le temps de réglage du filtre passe-bas utilisé précédemment. Troisièmement, la méthode VFH+ prend en compte la trajectoire du robot en masquant les secteurs qui sont bloqués par les obstacles. En conséquence, le robot ne peut être dirigé vers un obstacle, comme il a été possible avec la méthode VFH originale. Enfin, en appliquant une fonction de coût pour la sélection de la direction, la performance de l'algorithme d'évitement d'obstacle devient plus efficace et plus fiable. La fonction de coût donne aussi la possibilité de changer entre les comportements en modifiant simplement la fonction de coût ou ses paramètres. Le problème

qui reste de la méthode VFH + est sa nature locale, ce qui amène parfois le robot mobile en impasses qui pourraient être évitées. C'est le problème qui nous a amené à coupler la méthode réactive avec un planificateur global.

La planification globale de trajectoire implique la recherche d'un chemin sûr à partir de la position initiale du robot jusqu'au but dans un environnement connu. Il existe de nombreuses approches qui tentent de résoudre le problème. L'une d'elles est basée sur la fonction de navigation lorsque le problème de planification de trajectoire est transformé en un problème de descente de gradient de la position initiale jusqu'au but. La fonction de navigation nécessite un environnement connu et statique. Un chemin libre et continu peut être trouvé à l'avance par l'analyse de la connectivité de l'espace libre. La méthode trouve toujours un chemin libre et continu s'il existe.

La stratégie de notre approche consiste à calculer un chemin par le planificateur global basé sur une fonction de navigation (*wavefront*) et l'utiliser pour diriger la méthode réactive. Notre approche a les avantages d'un comportement réactif, tels que l'adaptation rapide aux changements et la flexibilité dans des circonstances imprévisibles. Elle permet d'éviter les limitations typiques d'autres méthodes, comme les situations de piège locales, les oscillations dans des environnements denses, et l'impossibilité d'obtenir une direction de mouvement vers les obstacles ou vers les zones à grande densité d'obstacles. Il en résulte un mouvement sûr et robuste dans des environnements qui restent toujours gênant pour de nombreuses méthodes existantes. En conclusion, notre approche est dotée de meilleures capacités de manoeuvrabilité spécialement en environnements dynamiques, denses, complexes et gênants.

Ce travail a ouvert de nouvelles perspectives dans le domaine de la navigation autonome d'un robot mobile. Cependant, il reste encore des points à étudier et développer plus profondément:

- Le premier point concerne les situations pour lesquelles l'évolution dans le temps de la position des obstacles mobiles est supposée connue. Dans ce cas, la prise en compte de ces obstacles peut être faite dés la phase de planification.
- Deuxièmement, dans certaines situations, le robot a la possibilité d'exécuter plusieurs tâches. Par exemple, lorsque le robot se déplace d'un endroit à un autre, il pourrait tenir une conversation avec quelqu'un sur son chemin, diffuser une présentation ou même exécuter une tâche de surveillance.
- Troisièmement, l'utilisation d'autres sources de données. Notre approche utilise un capteur laser à balayage horizontal. L'utilisation de stéréo-vision devrait être envisagée car elle apporterait à notre approche du suivi réactif de trajectoire des données proximétriques plus riches (volumes en 3 dimensions) et nous pourrions ainsi éviter, par exemple, des angles de bureau saillants.
- Parmi les perspectives aussi, nous prévoyons d'ajouter d'autres méthodes pour la localisation, la cartographie, qui font parties du système complet de navigation autonome.

Annexe A

Planification de Chemin en Environnement Statique Connu

La planification d'un chemin géométrique en environnement statique consiste en deux étapes consécutives:

- une étape de modélisation de l'espace libre dans $\boldsymbol{C}$
- une étape de construction d'un chemin dans cet espace

La construction du chemin se résume généralement à un parcours d'arbre et ne présente pas un intérêt particulier pour notre étude. En revanche bien que définies pour des environnements statiques, les méthodes de modélisation permettent d'avoir un aperçu des possibilités de partitionnement d'un espace, et leurs propriétés. Nous n'en citons ici que les principales, classées de la manière suivante:

- Les Roadmaps
- Graphe de visibilité
- Diagramme de Voronoï
- Méthode des autoroutes
- Méthode des silhouettes
- Les méthodes de décomposition cellulaire
  - Exactes
  - Approchées
- Les champs de potentiels

A.1 Roadmaps

Le terme "roadmap" désigne une carte des routes que le robot peut emprunter pou se rendre d'un endroit à l'autre de son espace de travail. Cela permet de ne considérer qu'un nombre limité de possibilités de chemins entre deux points. L'information est représentée sous forme de graphe. Les noeuds sont les "carrefours" et les branches les a "routes" libres entre ces derniers. Chaque carrefour est associé à une région (un sous-espace de $\boldsymbol{C_{free}}$) et réciproquement. Étant donnée une position de départ et une d'arrivée, le problème consiste à trouver les deux carrefours qui leur sont associés, puis à rechercher la route qui les relie.

A.1.1 Graphe de visibilité

La première méthode apparue porte sur des espaces de configuration de dimension 2 seulement. Cela signifie un robot mobile circulaire, ou un robot ayant une orientation fixe ([Lozano-Perez, 81]). L'algorithme consiste à construire un graphe reliant deux à deux tous les sommets des obstacles polygonaux qui peuvent être connectés entre eux par une ligne ne traversant pas un autre obstacle

(figure A.1). Le graphe obtenu porte le nom de graphe de visibilité ([Nilsson, 69]) du fait qu'il permet de connaître les sommet visibles depuis sommet donné.

Dans le cas d'un robot simple comme un polyèdre se déplaçant uniquement en translation, Lozano-Pérez a montré que cette construction pouvait se faire par la différence de Minkowski avec une complexité de $O(mnlog(mn))$, avec m et n les nombres de sommet respectifs du robot et des obstacles.

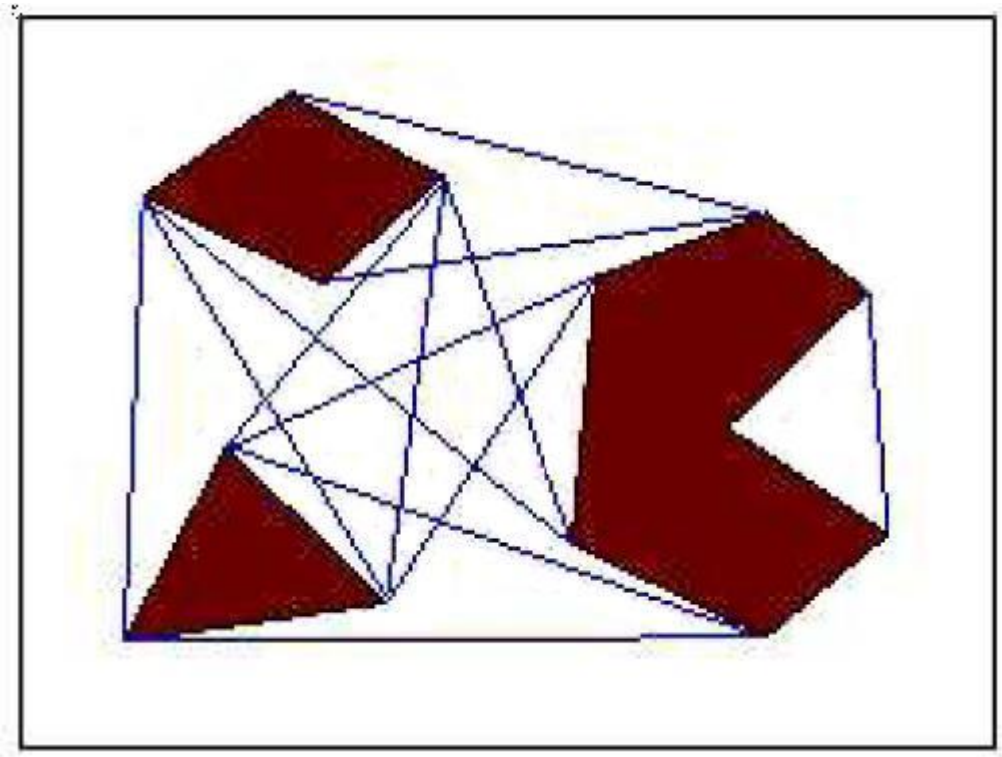


Fig. A.1*- Diagramme de Visibilité*

Cette approche présente cependant l'inconvénient de modéliser des déplacements frôlant les obstacles. De tels déplacements, libres en théorie, peuvent poser problème en pratique à cause des erreurs d'exécution (de suivi) ou des incertitudes sur la position des obstacles et du robot. Par mesure de sécurité, il est donc prudent de "grossir" les obstacles, au risque de masquer des solutions.

A.1.2 Méthode des autoroutes

La méthode des autoroutes ([Brooks, 83]) a une approche opposée de la précédente dans la mesure où elle consiste à passer le plus loin possible des obstacles. Pour ce faire, un couloir

correspondant à une zone libre, est calculé pour chaque couple de polygones. L'axe de ce couloir ainsi que sa largeur sont conservés. Les intersections des axes correspondent à des changements de direction éventuels du robot, et matérialisent la connectivité de deux couloirs (comme des carrefours). Ces points constituent les nœuds d'un graphe dont les arêtes représentent les couloirs et donc les déplacements autorisés. (figure A.2)

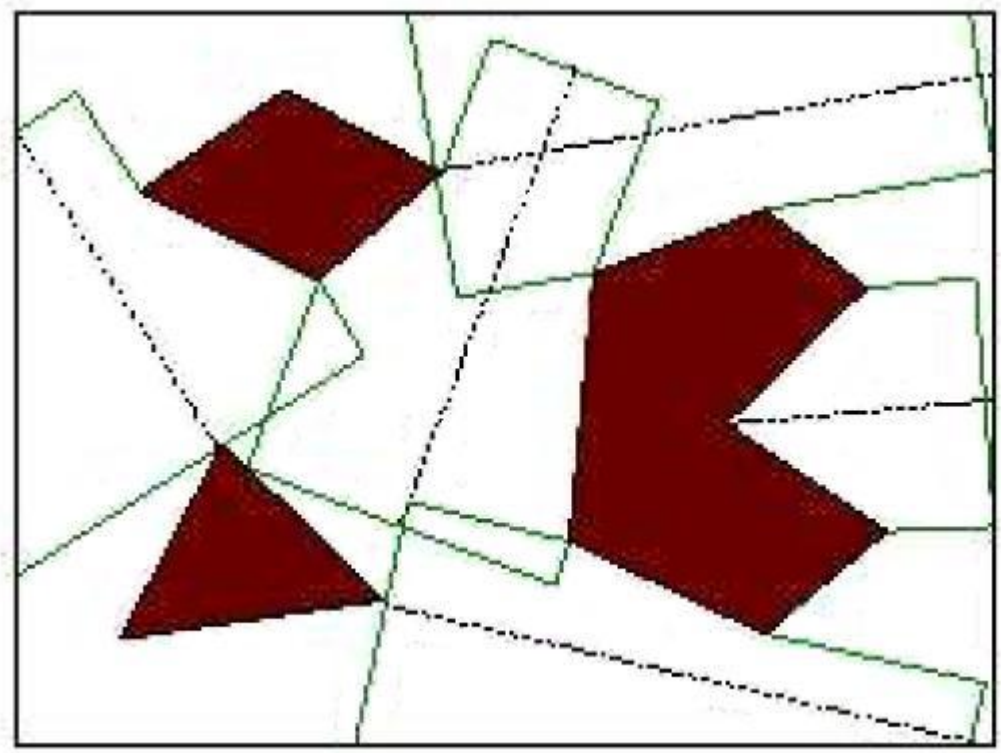


Fig. A.2 *- Autoroute*

Cette approche permet au robot de naviguer le plus loin possible des obstacles. Elle serait donc une bonne candidate pour limiter les conséquences des incertitudes. En contrepartie, cela conduit à des trajectoires plus longues et pas toujours très naturelle.

A. 1.3 Méthode des silhouettes

La méthode des silhouettes ([Canny et al., 88]) s'applique à tout espace de travail pouvant être représenté par un ensemble mathématique semi algébrique. Il s'agit du seul algorithme de planification de chemina ayant une complexité en temps exponentielle dans la dimension de ***C***. Le principe de la méthode consiste à balayer l'espace de configurations avec un hyperplan de dimension m-l, pour créer une " silhouette " de ***C***.

Ce principe est appliqué récursivement puis les " silhouettes " obtenues sont connectées entre elles dans un graphe pour représenter la connectivité de l'espace libre.

A.1.4 Graphe de Voronoï

Selon le même principe que la méthode des autoroutes, le diagramme de Voronoï, (figure A.3) ([O'Dunlaing et Yap, 82]) consiste à s'éloigner des obstacles, et en l'occurrence, en trouvant la courbe unidimensionnelle correspondant aux positions équidistantes des obstacles. En dimension 2, Yap a montré que le calcul de ce diagramme peut se faire en $O(nlog(n))$ où n est le nombre total de sommets des obstacles. La frontière (à égale distance d'au moins deux objets) des cellules obtenues constitue un squelette, matérialisant les points de passage privilégiés pour le robot.

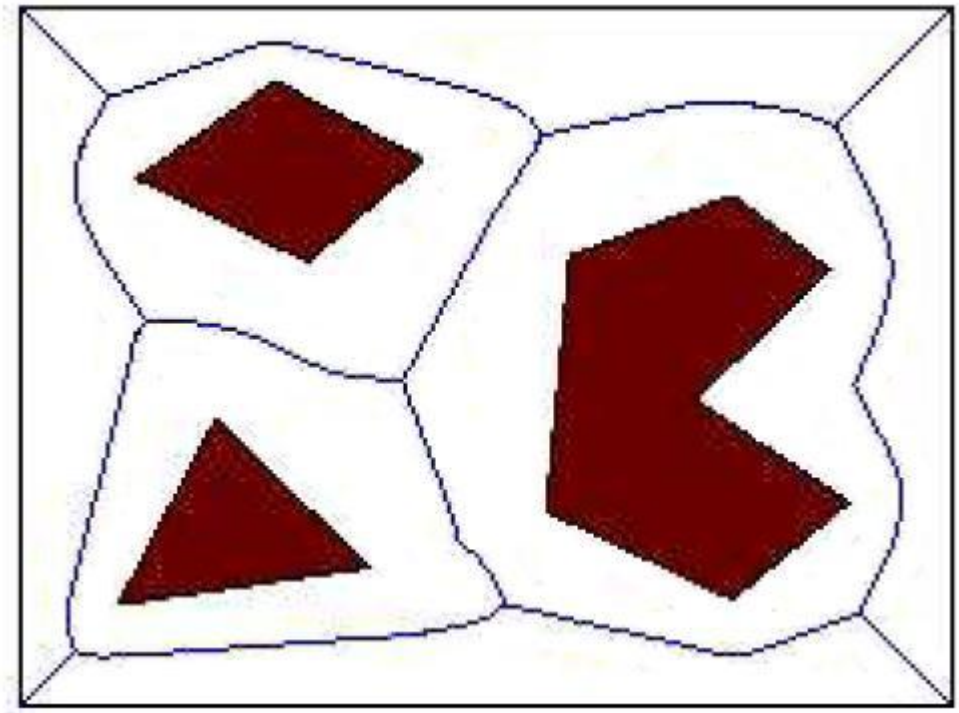


Fig. A.3 - *Voronoï*

Cette méthode est la plus couramment utilisée des méthodes polygonales. Les diagrammes de Voronoï sont des constructions classiques et bien maîtrisées,

A.1.5 Discussion

Les inconvénients des roadmaps sont leur grande dépendance au nombre total d'arêtes constituant les obstacles et la nécessité de travailler dans des espace de petite taille.

Leur avantage réside dans le fait qu'une pré interprétation de l'espace libre est réalisée par ces méthodes, puisque les informations représentées sont déjà des déplacements libres du robot. Cela peut également être un inconvénient du fait de la restriction importante des possibilités.
D'une manière générale, ces approches seront surtout utiles pour proposer un déplacement de référence du robot, calculé hors ligne à partir des informations connues a priori.

A.2 Décompositions cellulaires

A.2.1 Exactes

Principe: La décomposition cellulaire exacte a pour but d'obtenir une représentation polygonale aussi exacte que possible de l'espace libre, sous la forme d'un ensemble de cellules polygonales. La connectivité des cellules est exprimée à l'aide d'un graphe appelé *graphe de connectivité*.

Le principe général consiste à découper l'espace (***C***) peuplé de polygones (les ***C***-obstacles), en primitives géométriques telles que des triangles ou des trapèzes.

Discussion: Les techniques de décomposition cellulaire exacte sont généralement purement géométriques (triangulation de Delaunay par exemple) et possibles dans des espaces de faible dimension (2 ou 3). Pour des dimensions plus élevées, une méthode générale ([Schwartz et Shark, 83]) existe pour calculer ces ensembles, mais en pratique, son coût oblige à se limiter à des espaces de dimension 2, correspondant par exemple à l'espace des configurations d'un robot circulaire ou ayant une orientation figée (voir [Avnaim et al., 88]).

D'une manière générale, les calculs associés à ces techniques sont complexes et ne permettent pas de modéliser des environnements complexes[1] dans des temps raisonnables pour permettre la modélisation de l'environnement en ligne.

En revanche, lorsque cela n'est pas nécessaire (lorsque l'environnement est connu) la modélisation peut être réalisée hors ligne avec l'une de ces approches.

Les représentations polygonales des obstacles permettent alors de conserver une résolution importante de $\boldsymbol{C}_{free}$ dans ***C***.

A.2.2 Approchées

L'idée première consiste à casser la complexité des décompositions cellulaires exactes ([Schwartz et Shark, 83]) ou des roadmaps en considérant une discrétisation plus grossières de l'espace ([Brooks et Lozano-Perez, 85]). Cette discrétisation, matérialisée par un découpage de l'espace en cellules de même forme, peut être figée (cellules de dimension fixe) ou adaptative (cellules de dimension variable).

Grilles homogènes: La version la plus simple de grille d'occupation est la grille homogène. Son principe consiste à diviser l'espace en cellules rectangulaires (ou carrées) régulières. Un contenu booléen permet de savoir si un obstacle occupe la zone correspondante de l'espace, et donc son appartenance à $\boldsymbol{C}_{col}$ ou $\boldsymbol{C}_{free}$.

---

[1] *La complexité de l'environnement dépend pour ces approches géométriques, du nombre total de segments constituant les obstacles*

L'avantage principal de cette méthode est sa simplicité, tant sur le principe que sur les calculs mise en oeuvre. En revanche son principal défaut est sa forte dépendance vis-à-vis de la résolution de la grille: des cases trop grandes dans un environnement contraint peuvent aboutir à $C_{free} = \varphi$ (figure A.4).
Il faut dans ce cas augmenter le nombre de cases, ce qui signifie augmenter la quantité de calculs et faire face à des problèmes de taille de stockage des informations.

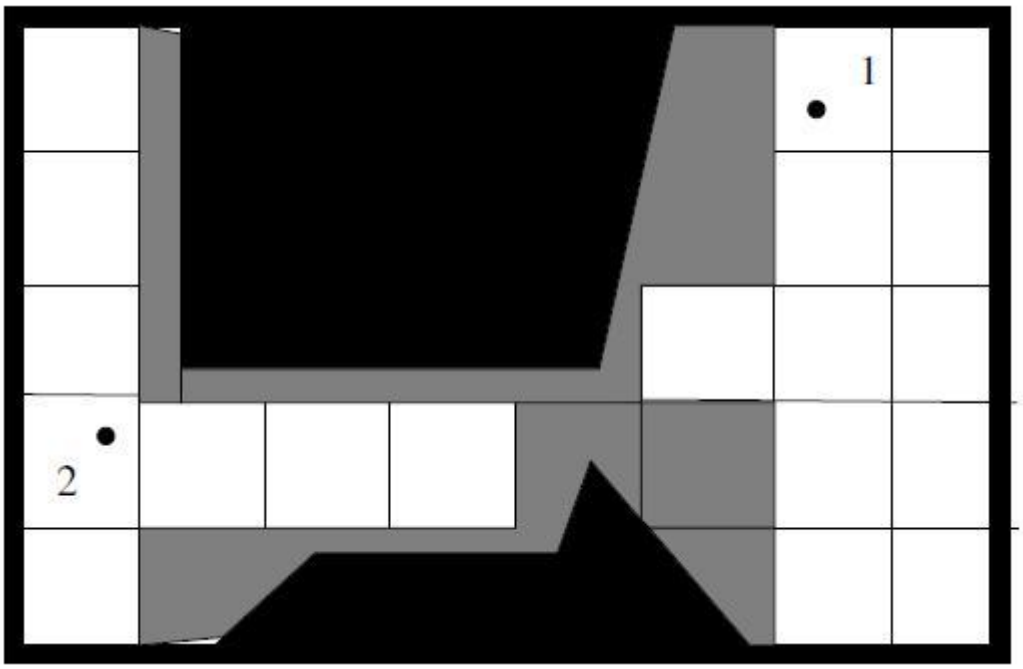


Fig. A.4 - $C_{free} = \varphi$

Décomposition adaptative: Au lieu de considérer des cellules de même taille, il est également possible de travailler avec des cases de taille variable en fonction de la topologie des obstacles. Cela permet d'éviter une taille de cellule trop petite et l'explosion du nombre de cellules qui l'accompagne, lorsque l'environnement comporte des passages étroits.
L'algorithme le plus répandu est appelé quadtrees. Il s'agit d'un découpage récursif d'une cellule rectangulaire en quatre sous cellules de même dimension:
- soit jusqu'à ce qu'une cellule corresponde à un espace tout en collision ou tout libre,
- soit jusqu'à ce que la condition précédente soit fausse, mais que la cellule ail atteint la taille minimale donnée.

Cette méthode permet une discrétisation hiérarchique de l'environnement, permettant de choisir le niveau de discrétisation auquel l'on désire travailler.
Comme pour toutes les méthodes basées sur des grilles, l'inconvénient de la méthode porte sur le nombre de cellules pour des environnements très contraints, mais dans une mesure moindre.

Une approche semblable consiste à représenter dans un arbre, une hiérarchie de boites englobantes des obstacles (alignées sur les axes principaux ou ceux des objets) Cette approche est séduisante en particulier pour des " boites " de forme circulaire en 2D ou sphériques en 3D ([Quinlan, 94]). Elle permet en effet d'approximer l'environnement par un ensemble d'obstacles circulaires, avec une résolution variable (en fonction du niveau de profondeur choisi dans l'arbre). Les tests de collisions sont alors réduits à des calculs de distance entre le robot (représenté par un point dans ***C***) et le centre de ***C***-obstacles.

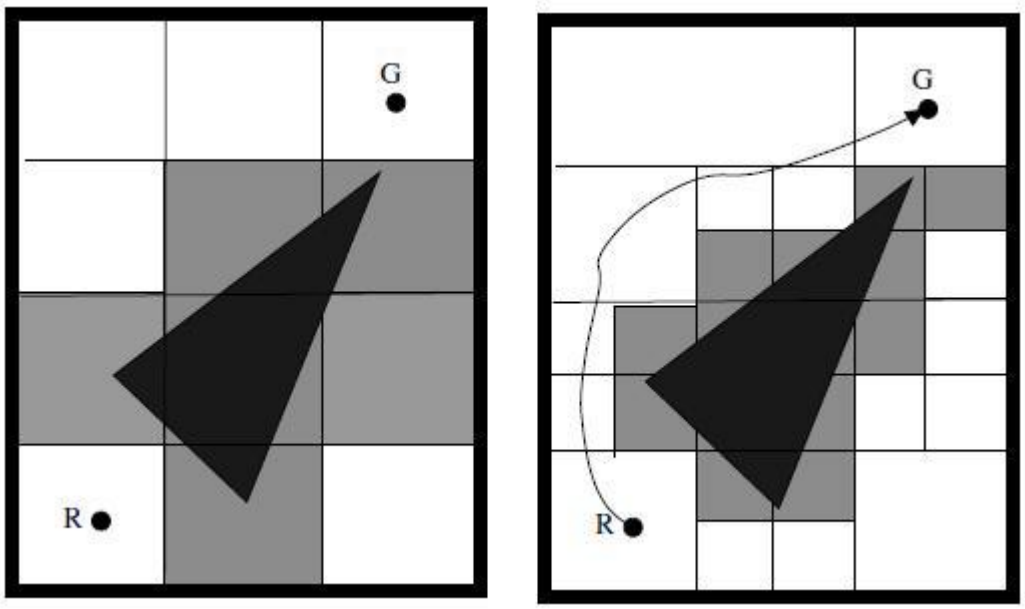


Fig. A.5 - *Différents Niveaux de Résolution*

Discussion sur l'utilisation de ces approches pour la navigation: En raison de la faible dimension des espaces manipulés, les méthodes de décomposition cellulaire approchées, et en particulier les approches adaptatives hiérarchiques, permettent une représentation suffisamment précise de $\boldsymbol{C_{free}}$ sans avoir la complexité des approches exactes.

Ces approches sont parfaitement utilisables par une méthode de navigation pour la plupart des environnements de la vie courante. De plus, en cas de modification d'une partie de l'environnement, la structure hiérarchique de cette représentation permet ne pas avoir à tout recalculer mais seulement la région modifiée.

A.3 Champs de potentiels

La méthode des champs de potentiels associe des forces attractives au but et des forces répulsives aux obstacles pour créer des champs de forces. En suivant les valeurs décroissantes de ce dernier le robot se dirige vers son but en évitant les obstacles.

L'inconvénient majeur de cette approche est le fait que la fonction de potentiel qui permet de diriger le robot vers le but, n'est pas strictement décroissante vers ce but, et admet des minima locaux. L'approche initiale ([Khatib, 80] [Khatib, 86]) est sujette aux problèmes de minima locaux et ne garantit donc pas que le robot sera en mesure d'atteindre son but en suivant la courbe des potentiels décroissants. La première solution envisagée pour y remédier vise à se sortir des minima en appliquant des déplacements aléatoires lorsque ces derniers sont atteints ([Barraquand et Latombe, 89]). La seconde solution est de garantir l'absence totale de minima locaux. Des fonctions de navigation ont donc été étudiées pour satisfaire cette contrainte: La première propose de séparer le gradient menant vers le but de celui associé aux obstacles (figure A.6). La fusion des deux conserve une pente vers le but en tout point, et minimise les minima locaux ([Koditschek, 87] [Rimon et Koditschek, 92]) sans toutefois les supprimer totalement. L'autre solution, meilleure mais nécessitant des calculs plus complexes, fait appel aux fonctions harmoniques ([Connolly et al., 90]) et se présente comme une analogie avec les fluides ([Feder et Slotine, 97]) ou l'électricité ([Connolly et Grupe, 94]).

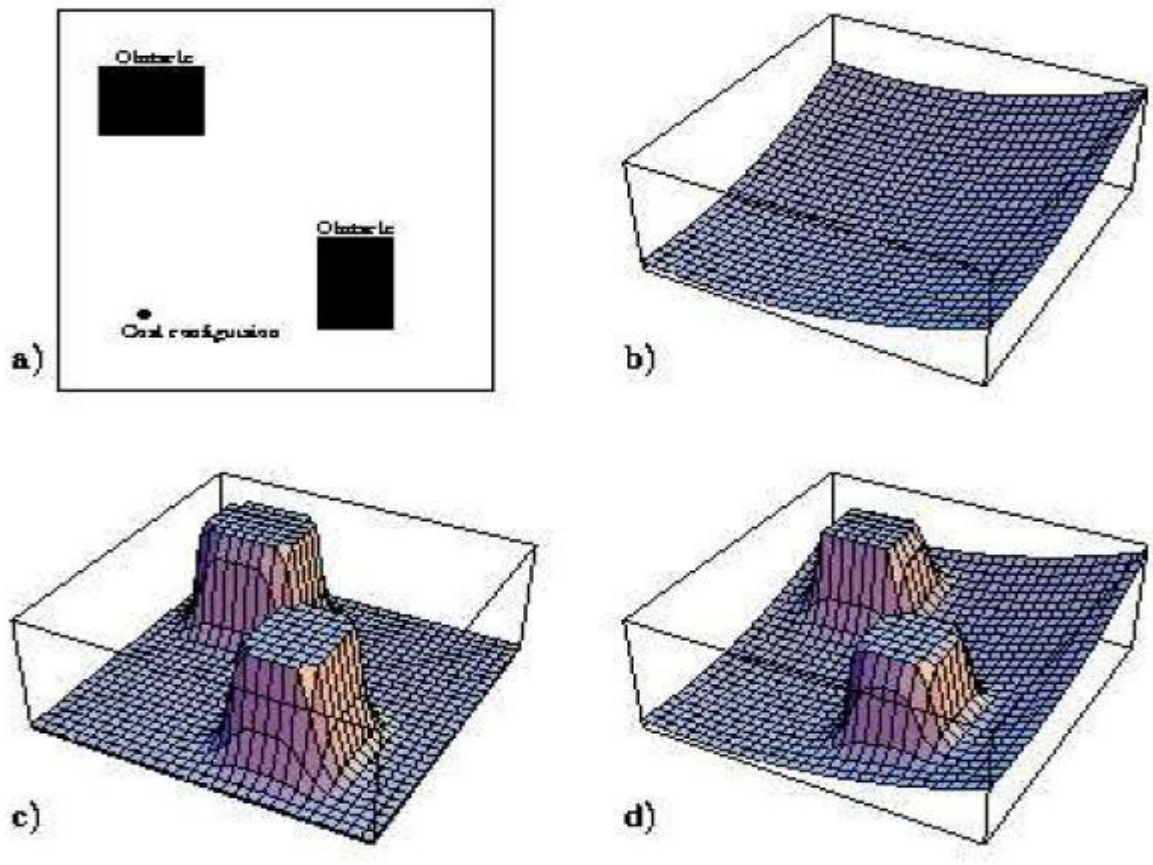


Fig. A.6 *- Champs de Potentiel*

A.3.1 Discussion sur l'utilisation de ces approches pour la navigation

Pour des raisons d'efficacité, le calcul des potentiels est généralement discrétisé selon une grille régulière définie dans l'espace de travail du robot ou celui des configurations. Par commodité, les obstacles peuvent être discrétisés selon la même grille selon le principe des grilles homogènes. Des versions analytiques non discrétisées ont également été proposées notamment par Faverjon, pour permettre une meilleure résolution.

Dans ces conditions et pour des environnements de taille raisonnable, les champs de potentiels peuvent en théorie être utilisés en ligne par une méthode de navigation. Ils présentent de plus l'avantage de fournir un déplacement éloigné des obstacles à l'image des autoroutes ou du diagramme de Voronoï cités précédemment.
En pratique, cela dépendra de notre capacité à s'affranchir des problèmes de minima locaux, à l'aide d'une méthode appropriée de choix d'un déplacement.

Bibliographie

[1] Roland Siegwart and Illah R. Nourbakhsh, Introduction to Autonomous Mobile Robots, the MIT press. ISBN: 0-262-19502-X.

[2] Eric Beaudry, Planification de tâches pour robotique mobile, thèse de doctorat, université de Sherbrooke, Canada, Mai 2006

[3] C. Durieu, algorithmes de localisation d'un robot mobile dans un milieu balisé par mesure de distance ou d'angle de gisement en tenant compte des mesures aberrantes. Algorithmes de calibration et de recalage du champ de balise, thèse de doctorat, université de Paris sud 1989.

[4] Robin R. Murphy : Introduction to AI Robotics. MIT Press, Cambridge, MA, USA, 2000.

[5] Rodney A. Brooks : A robust layered control system for a mobile robot. Rapport technique, Cambridge, MA, USA, 1985.

[6] Ronald C. Arkin : An Behavior-based Robotics. MIT Press, Cambridge, MA, USA,1998.

[7] Frédéric Large, Navigation d'un robot mobile en environnement dynamique et incertain, thèse de doctorat, université de Savoie, 2003

[8] E Gauthier, Utilisation des réseaux de neurones pour la commande d'un véhicule autonome, Ph.D thèse, Inst. Nat. Polytechnique de Grenoble, 1999

[9] O Khatib, Real-Time obstacle avoidance for manipulators and mobile robots, International Journal Robotics research 5(1), 90-98, 1986

[10] Borenstein and Koren, The vector field histogram – fast obstacle avoidance for mobile robots, IEEE Transactions on Robotics and Automation 7(3), 278-288, 1991

[11] Ulrich and Borenstein, VFH+ : Reliable Obstacle Avoidance for Fast Mobile Robots, In IEEE int. conf. on Robotics and Automation, Leuven, Belgium, 1998

[12] I. Ulrich, J. Borenstein, VFH* : Local Obstacle Avoidance with Lookahead Verification In IEEE int. conf. on Robotics and Automation, San Francisco, USA, 2000

[13] Simmons R., The Curvature Velocity Method for Local Obstacle Avoidance, In IEEE Int. conf. on Robotics and Automation, Minneapolis, USA, 1996

[14] N. Y. Ko and R. Simmons, The Lane-Curvature Method for Local Obstacle Avoidance, In Proc. Of the IEEE/RSJ Int. conf. on Intelligent Robots and Systems (IROS), 1998

[15] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. In IEEE Robotics & Automation Magazine, 4(1), 1997.

[16] J. Minguez, L. Montano. Nearness Diagram Navigation (ND): A new Real Time Collision Avoidance Approach. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2000. Takamatsu, Japan.

[17] J. Minguez, L. Montano, N. Siméon, R. Alami, Global Nearness Diagram Navigation; In Proc. of the IEEE Int. conf. on Robotics and Automation, Washington DC, 2002.

[18] LaValle, Rapidly-Exploring Random Trees: A new Tool for Path Planning, Technical Report No. 98-11, Dept. of Computer Science, Iowa State University, 1998

[19] Brock and Khatib, Real-Time Replanning in High-dimensional Configuration Spaces Using Sets of Homotopic Paths, In IEEE int. conf. on Robotics and Automation, San Francisco, USA, 2000

[20] 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.

[21] A. Arleo andW. 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.

[22] 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.

[23] V. Braitenberg. Vehicles. MIT Press, Cambridge MA, 1984.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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, 2001.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

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

[35] J.-P. Laumond. Robot Motion Planning and Control. Lectures Notes in Control and Information Sciences 229. Springer, 1998.

[36] 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.

[37] T. S. Levitt and D. T. Lawton. Qualitative navigation for mobile robots. Artificial Intelligence, 44 :305–360, 1990.

[38] R. R. Murphy. Introduction to AI Robotics. The MIT Press, 2000.

[39] 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.

[40] S. Thrun. Learning metric-topological maps maps for indoor mobile robot navigation. Artificial Intelligence, 99(1) :21–71, 1999.

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

[42] 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.

[43] O. Trullier and J. A. Meyer. Biomimetic navigation models and strategies in animats. AI Communications, 10 :79–92, 1997.

[44] 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.

[45] 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.

[46] G. Von Wichert. Mobile robot localization using a self-organised visual environment representation. Robotics and Autonomous Systems, 25 :185–194, 1998.

[47] Arkin, R., 1989. Towards the unification of navigational planning and reactive control. In: Working Notes of the AIII Spring Symposium on Robot Navigation. Stanford University, pp. 1–6.

[48] Arkin, R., 1987. Motor schema based navigation for a mobile robot: An approach to programming by behavior. In: IEEE International Conference on Robotics and Automation. Raleigh, USA, pp. 264–271.

[49] Agree, P., Chapman, D., 1990. What are the plans for. Journal for Robotics and Autonomous Systems 6, 17–34.

[50] Lyons, D., Hendriks, A., 1992. Planning, reactive. In: Saphiro, S., Wiley, J. (Eds.), Encyclopedia of Artificial Intelligence. pp. 1171–1182.

[51] Arkin, R., 1999. Behavior-Based Robotics. The MIT Press.

[52] Minguez, J., Montano, L., Simeon, N., Alami, R., 2001. Global Nearness Diagram Navigation (GND). In: IEEE International Conf. on Robotics and Automation. Seoul, Korea, pp. 33–39.

[53] Quinlan, S., Khatib, O., 1993. Elastic Bands: Connecting Path Planning and Contrôle. In: IEEE Int. Conf. on Robotics and Automation. Vol. 2. Atlanta, USA, pp. 802-807.

[54] Choi, W., Latombe, J., 1991. A reactive architecture for planning and executing robot motion with incomplete knowledge. In: IEEE/RSJ International Workshop on Intelligent Robots and Systems. Osaka, Japon, pp. 24–29.

[55] Minguez, J., Montano, L., 2004. Nearness Diagram (ND) Navigation: Collision Avoidance in Troublesome Scenarios. IEEE Transactions on Robotics and Automation 20 (1), 45–59.

[56] Minguez, J., & Montano, L. (2005). Sensor-based robot motion generation in unknown, dynamic and troublesome scenarios. *Robotics and Autonomous Systems, 52*(4), 290–311.

[57] Foley, J., Dam, AV, Feiner, S., Hughes, J., 1990. Computer Graphics, principles and practice. Addison Wesley edition 2nd.

[58] Moravec, H.P., “Sensor fusion in certainty grids for mobile robots”, AI Magazine, Summer 1988, pp. 61-74.

[59] Elfes, A., “Using occupancy grids for mobile robot perception and navigation”, Computer Magazine, June 1989, pp. 46-57.

[60] Kramer, J.,Scheutz, M., “Development environments for autonomous mobile robots: A survey”, C _Springer Science + Business Media, LLC 2006.

[61] http://playerstage.sourceforge.net/wiki/Main_Page.