Stratégie d'exploration multirobot fondées sur le calcul de champs de potentiels

Antoine Bautin

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR STMIA

Stratégie d'exploration multirobot fondée sur le calcul de champs de potentiels

THÈSE

présentée et soutenue publiquement le 3 octobre 2013

pour l'obtention du

Doctorat de l'université de Lorraine

(spécialité informatique)

par

Antoine Bautin

Composition du jury

Président :	Raja Chatila, *Directeur de recherche, CNRS, ISIR*
Rapporteurs :	Simon Lacroix, *Directeur de recherche, CNRS, LAAS* Philippe Mathieu, *Professeur, Université Lille 1, LIFL*
Examinateurs :	Nicolas Bredeche, *Professeur, Université Pierre et Marie Curie, ISIR* Jean-Paul Haton, *Professeur émérite, Université de Lorraine, LORIA* Philippe Lucidarme, *Maître de conférences, Université d'Angers, LISA*
Directeurs :	François Charpillet, *Directeur de recherche, Inria, LORIA* Olivier Simonin, *Maître de conférences HDR, Université de Lorraine, LORIA*

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thloria.

Table des matières

Partie III Intégration robotique et expérimentations

Partie IV Conclusion

Chapitre 1

Introduction

Sommaire

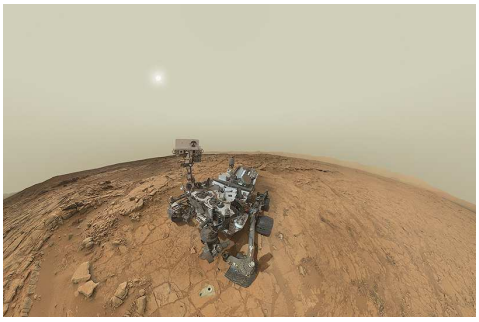
L'exploration d'un environnement est un problème fondamental en robotique mobile. Elle consiste, dans les cas les plus difficiles, à découvrir un environnement inconnu ou ayant subi des modifications et à en construire une représentation. La carte qui en résulte est en général nécessaire aux robots pour accomplir une tâche complexe de manière autonome. Ainsi l'**exploration** consiste à déterminer comment les robots cartographient un terrain inconnu pour en découvrir les caractéristiques. Durant l'exploration le problème de la **couverture** se pose. Celle-ci consiste à déterminer comment les robots se déplacent dans l'environnement afin d'observer exhaustivement toute zone de l'environnement ; c'est donc un sous-problème de l'exploration. La couverture est complète si la totalité de l'environnement a été observé.

L'exploration d'environnements inconnus nécessite la résolution de trois sous-problèmes, la localisation d'un ou plusieurs robots, la cartographie des zones observées et la couverture, au sens de la découverte d'un espace inconnu. Dans cette thèse, nous nous intéressons particulièrement à ce dernier problème. Plus précisément, nous cherchons une stratégie permettant à plusieurs robots de cartographier toutes les parties visibles de l'environnement le plus rapidement possible. Dans ce contexte, l'utilisation de plusieurs robots est avantageuse car elle permet d'augmenter l'efficacité en temps et en précision de la couverture. Cependant, comme nous le montrerons dans la suite, le gain est conditionné par le niveau de coopération entre les robots. La mise en œuvre d'une stratégie de coopération est donc primordiale pour une cartographie multirobot efficace.

1.1 Enjeux de l'exploration robotisée

Les avancées technologiques récentes ont permis de sortir les robots des environnements contrôlés du milieu industriel afin de réaliser des tâches de plus en plus variées. Parmi celles-ci, on trouve l'exploration d'environnements permettant la construction de cartes et la localisation et l'identification d'objets. Ces cartes peuvent ensuite être utilisées par d'autres robots pour y effectuer d'autres tâches ou servir aux humains pour des environnements inaccessibles ou dangereux pour l'homme. De nombreux exemples d'applications existent dont quelques-uns sont illustrés ci-dessous (en figure 1.1).

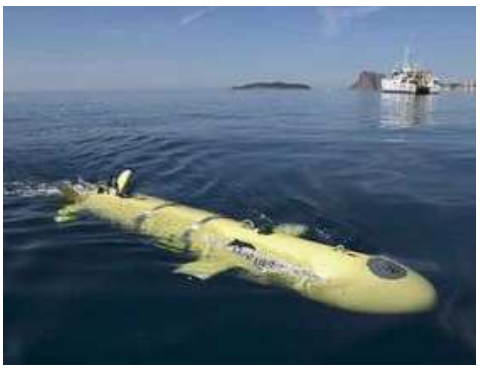
L'exploration d'environnement inconnu est utile pour les environnements lointains tels que Mars (figure 1.1a), pour la recherche et le secours après certaines catastrophes, pour l'inspection de zones dangereuses comme les zones radioactives (figure 1.1c) ou les terrains minés, et de manière plus générale pour les environnements hostiles, comme les mines souterraines (figure 1.1b) et les environnements sous-marins (figure 1.1d). Les applications de l'exploration robotique sont nombreuses également pour la surveillance des villes, des bâtiments industriels ou des maisons. Il y a aussi les applications internet tel que Google Maps qui commence à proposer des visites virtuelles de bâtiments comme des monuments historiques ou des centres commerciaux. La cartographie en trois dimensions de ces lieux publics est une des applications possibles des travaux présentés dans cette thèse.



(a) Autoportrait de Curiosity sur Mars
apod.nasa.gov/apod/ap130222.html

(b) Exploration de mine souterraine
www.frc.ri.cmu.edu/robots/

(c) Robot Packbot explorant la centrale nucléaire de Fukushima en 2011 après la fusion du cœur d'un réacteur
[Tokyo Electric Power Co.]



(d) Véhicule sous-marin autonome ASTER de l'IFREMER pouvant atteindre 3000 mètres de profondeur
flotte.ifremer.fr

FIGURE 1.1 – Exemples d'utilisation de robots pour l'exploration d'environnements hostiles aux humains

1.2 Motivation

L'exploration robotique est un sujet largement étudié depuis les débuts de la robotique mobile car il est considéré comme un des problèmes emblématique de la robotique mobile autonome. Les recherches se sont d'abord concentrées sur l'exploration monorobot (voir section 2.1). La miniaturisation des composants robotiques et la diminution de leurs coûts permet aujourd'hui d'envisager le déploiement effectif de flottilles de robots, pour réaliser toutes les tâches pour lesquelles l'utilisation de plusieurs robots constitue un réel avantage. Ainsi, l'exploration multirobot fait l'objet de nombreux travaux car les avantages par rapport au monorobot sont nombreux ; principalement au niveau de l'efficacité et de la robustesse (voir section 1.4). Les approches existantes sont cependant centralisées et elles reposent fortement sur la communication entre robots. La bande passante des dispositifs de communication sans fil étant limitée, ces approches ne passent pas à l'échelle d'un grand nombre de robots.

C'est pourquoi la principale motivation de ce travail est de proposer une approche décentralisée et distribuée qui limite naturellement les communications pour pouvoir passer à l'échelle. L'objectif est ainsi d'accroître les performances en temps d'exploration des approches multirobot actuelles tout en limitant leurs coûts de calcul. En effet, les systèmes multirobot sont, en général, composés de robots simples, peu coûteux et possédant une faible capacité de calcul.

L'expérimentation et la validation de l'approche proposée ont été effectuées à la fois en simulation et sur robots réels. Le contexte de cette thèse est le défi robotique CAROTTE (CArtographie par RObot d'un TErritoire) qui est un défi robotique national consistant à cartographier un environnement structuré et à y localiser des objets de manière complètement autonome (sans intervention humaine) et dans un temps contraint.

1.3 Exploration multi-robot d'un environnement inconnu

Dans ce contexte, nous nous intéressons à l'exploration d'espaces finis par le déploiement d'un ensemble de robots mobiles autonomes dotés de capacités d'observation leur permettant d'établir une carte locale de l'environnement et de se situer dans celle-ci. En pratique, ceci est possible en équipant les robots d'un capteur de distance (de type télémètre) et en utilisant un système de localisation externe (par exemple un GPS pour les milieux extérieurs terrestres) ou un algorithme de localisation et cartographie simultanée (SLAM, voir section 8.4). Afin de permettre la coopération entre les robots, il est essentiel de les munir de moyens de communication. Nous verrons en section 8.4.3 que cette communication ne peut pas être parfaite en pratique (pertes inhérentes aux communications sans-fil).

Le problème que nous traitons peut se formuler comme étant l'assignation à chaque robot d'une zone à explorer afin d'étendre la connaissance que possède la flottille de robot sur l'environnement. L'évolution de la carte implique une ré-assignation continue des robots vers les zones à explorer nouvellement identifiées. Il s'agit donc d'un problème **dynamique**.

1.4 Approche multirobot

Nous présentons, ici, les avantages et les inconvénients d'une approche multirobot.

1.4.1 Avantages

Notre choix d'étudier une approche multirobot repose sur les nombreux avantages qu'elle offre pour l'exploration. Ces principaux avantages sont la robustesse, la rapidité d'exploration et la précision. Nous les détaillons ci-après.

Robustesse

Un robot peut-être sujet à des pannes dont les causes peuvent être dues à un mauvais fonctionnement des batteries, des moteurs, des cartes électroniques ou des autres composants. Il peut aussi être bloqué contre un obstacle ou simplement être perdu. Leurs effets sont l'immobilisation d'un robot, la perte définitive de communication ou l'arrêt total du système. En monorobot une panne signifie généralement l'échec de la mission et la nécessaire réparation du robot afin de pouvoir continuer. En multirobot, la perte d'un ou plusieurs robots n'entraîne pas l'échec de la mission en cours, les robots encore opérationnels pouvant continuer. Aussi, si un robot est bloqué à cause d'une zone dangereuse mais continue de fonctionner, il pourra prévenir les autres robots du danger.

Approche décentralisée et distribuée

Du point de vue de la robustesse, une approche centralisée de la coordination et/ou de la collection/consultation de données fait reposer le succès de la mission sur un seul composant. En comparaison, un système décentralisé ne comporte pas de nœud plus important que d'autre et la perte d'un nœud ne compromet pas le succès de la mission. Du point de vue du calcul, la distribution sur plusieurs nœuds du problème est plus efficace et permet un meilleur passage à l'échelle. L'utilisation d'un calcul distribué pour la cartographie permet une plus grande efficacité de calcul et une moins grande utilisation de la mémoire. En effet, chaque robot n'a pas besoin de visiter et de construire une carte précise de tout l'environnement. Toutefois, le problème étant global, il doit disposer d'une information permettant de savoir quelles zones sont explorées et quelles sont les directions restant à explorer.

Rapidité d'exploration

L'utilisation de plusieurs robots permet de diviser l'exploration en plusieurs régions assignées à des robots différents. Cette exploration simultanée de différentes zones permet de découvrir plus rapidement l'environnement qu'avec un seul robot qui devra explorer séquentiellement chaque zone. Nous verrons que le nombre de robots à mettre en œuvre dépend du type d'environnement exploré. Par exemple, l'exploration d'un seul long couloir en partant d'une extrémité ne nécessite pas plus d'un robot ; les robots supplémentaires ne pouvant que suivre le premier. Le bénéfice apporté par l'ajout de robots n'est pas systématique, en effet, les gênes potentielles entre robots peuvent augmenter le temps d'exploration.

Le gain maximal théorique sur le temps d'exploration de l'utilisation de n robots, plutôt qu'un seul, est de n. Le temps d'exploration théorique avec n robots est donc égal au mieux au temps d'exploration avec un robot divisé par n. En effet, le gain d'information par l'utilisation de n robots est multiplié par n si tous les robots observent durant toute l'exploration des parties différentes de l'environnement. En revanche, quand ils observent les mêmes zones, les robots n'apportent pas plus d'information au niveau de l'exploration mais leurs apports d'information peut améliorer la précision de la carte sur ces zones déjà explorées. En pratique, le gain dépend de la topologie de l'environnement et de la position initiale des robots.

Précision de la carte

Comme indiqué précédemment, plus la quantité d'informations récoltées sur la carte est importante, plus la précision de la carte construite sera grande. Aussi, si les robots coopèrent pour se localiser, cette précision peut encore augmenter. Les robots peuvent, par exemple, mutuellement s'observer afin de se localiser plus précisément par trilatération ou triangulation. La nécessité d'utiliser ces techniques dépend de la précision de la localisation des robots et de la précision voulue de la carte à construire.

1.4.2 Désavantages

L'utilisation d'une méthode multirobot comporte aussi un nombre limité d'inconvénients décrits ci-après.

La navigation de plusieurs robots dans un environnement nécessite une coordination car les robots peuvent parcourir les mêmes zones et se gêner, peut-être même entrer en collision. L'évitement de collisions ou la résolution d'inter-blocages occasionnent des détours durant l'exploration.

La cartographie avec plusieurs robots est plus complexe. L'utilisation d'un même type de capteur peut générer des interférences faussant les observations. De plus, un robot observant un autre robot peut le cartographier comme un obstacle faisant partie de l'environnement. Si la cartographie est distribuée, la perte d'un robot peut entraîner la perte des informations recueillies par ce robot. La perte de communication d'un robot peut entraîner la nécessité pour d'autres robots de ré-explorer la zone découverte par le robot défaillant.

Un système multirobot nécessite la construction d'une carte globale sur chaque robot permettant d'identifier les zones restant à explorer. Ceci nécessite la mise en correspondance des cartes de chaque robot pour pouvoir les fusionner. Si les robots partent d'une même zone, l'identification d'un repère commun et la mise en correspondance des cartes est assez simple. En l'absence d'un repère commun, il faudra identifier les transformations entre les repères des cartes de chaque robot. Des techniques de mise en correspondance de cartes existent (Birk and Carpin, 2006). Le résultat d'une fusion erronée, due à une mauvaise mise en correspondance, peut fausser sévèrement les cartes résultantes et bloquer les robots qui sont alors soit perdus (ne connaissant plus leur position) soit bloqués (tous les chemins étant virtuellement obstrués).

Les avantages importants d'une approche multirobot justifient son choix pour l'exploration, ses inconvénients pouvant être gérés par l'utilisation d'une bonne stratégie d'exploration.

1.5 Exploration

L'exploration consiste à découvrir l'environnement avec un ou plusieurs capteurs montés sur des robots. Les robots ne doivent pas couvrir physiquement tout l'environnement mais doivent se déplacer afin que leurs capteurs puissent observer la totalité de l'environnement et ainsi construire une carte complète.

L'environnement et les robots ne sont pas des paramètres modifiables, ils sont, la plupart du temps, fixés par le problème. Nous cherchons la meilleure stratégie étant donné le problème, mais l'environnement étant inconnu, la stratégie doit être assez générique pour être performante sur tous les types d'environnement. L'environnement est statique, il ne change pas au cours du temps. Pour un robot, seul les autres robots sont des obstacles mobiles. Il existe un nombre de robots optimal pour un environnement permettant une exploration rapide tout en évitant les

conflits entre robots. L'environnement étant inconnu, le nombre de robots doit être maximal ou choisi en fonction d'une estimation de la superficie de l'environnement.

Les robots disposent de capacités de perceptions locales de leur environnement et gardent en mémoire les zones perçues. L'environnement peut donc être séparé en deux zones, l'une connue - explorée, l'autre inconnue - inexplorée. L'exploration consiste à augmenter la taille de la zone connue jusqu'à ce qu'il n'y ait plus de zones inexplorées atteignables par les robots. La limite entre ces deux zones est appelée **frontière**. La figure 1.2 illustre une situation où deux robots ont exploré une partie de l'environnement définissant ainsi ces deux zones et des frontières.

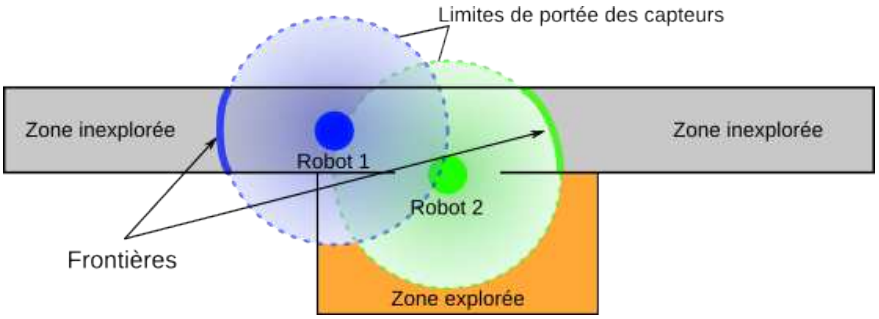


FIGURE 1.2 – Frontière : deux robots sortent dans un couloir et définissent, par la portée limitée de leur capteur, deux frontières.

1.6 Contributions

L'exploration multirobot nécessite une stratégie permettant une bonne coopération entre les robots. La principale contribution de cette thèse est une stratégie ayant de bonnes performances en temps d'exploration totale par rapport aux stratégies existantes. Elle est fondée sur la position relative d'un robot vers une frontière par rapport aux autres robots. Ainsi, chaque robot se dirige vers la frontière pour laquelle il est en meilleure position. Une variante de cette stratégie est aussi proposée. Elle consiste à affecter les frontières aux robots de manière gloutonne tout en prenant en compte le critère de position pour déterminer quelles frontières seront affectées.

Les implémentations proposées de ces stratégies sont très efficaces. Elles sont fondées sur la propagation synchronisée de fronts de vagues depuis les frontières. Ces propagations synchronisées réduisent fortement le coût de calcul des distances entre les robots et les frontières nécessaires pour l'affectation.

Nous proposons aussi un cadre de mise en œuvre de ces stratégies sur des robots réels. Plusieurs contributions ont été mises en œuvre dans ce cadre :

- une méthode de coopération entre robots limitant le volume d'information communiqué entre les robots,
- une stratégie d'exploration des frontières prenant en considération le champ de vision des capteurs,
- une planification de trajectoire fondée sur un A* à quatre dimensions utilisant une heuristique originale,
- une architecture logicielle mettant en œuvre la stratégie d'exploration de chaque robot de façon autonome.

1.7 Plan de la thèse

Cette thèse est composée de dix chapitres répartis en quatre parties.

- La première partie présente le problème et décrit le contexte dans lequel les travaux de recherche ont été effectués.

  - Le chapitre 2 présente le contexte de recherche de l'exploration multirobot en décrivant l'état de l'art. Dans ce chapitre, les approches existantes sont présentées selon les différents aspects liés aux objectifs de l'exploration, aux hypothèses de travail et aux méthodes d'exploration.

  - Le chapitre 3 présente plus précisément le problème abordé, en donne une définition formelle, ainsi que les contraintes que nous avons retenues. De plus, les critères d'évaluation d'une stratégie et d'une affectation sont présentés.

- La seconde partie est consacrée à l'étude des algorithmes d'affectation des frontières à explorer et à la proposition de nouveaux algorithmes.

  - Le chapitre 4 décrit les nouveaux algorithmes d'affectation de frontière *MinPos* et *MPG*. Leurs qualités et défauts sont étudiés qualitativement à partir d'exemples statiques.

  - Dans le chapitre 5, les performances de ces algorithmes sont évaluées en les comparant aux principaux algorithmes de la littérature. Les fréquences à laquelle ils peuvent être exécutés sont étudiées et l'impact de ces fréquences sur les différents algorithmes est évalué.

  - Le chapitre 6 décrit l'implémentation fondée sur les champs de potentiels. Nous évaluons ensuite son efficacité.

- La troisième partie aborde l'implémentation et l'évaluation de notre approche dans un contexte réel ; l'intégration robotique de la stratégie d'exploration y est décrite.

  - Le chapitre 7 décrit la méthode d'identification des frontières à partir de la carte construite. Une méthode pour l'exploration sous la contrainte de capteurs ayant un angle de champ de perception restreint et une portée limitée comme le capteur Kinect est proposée.

  - Dans le chapitre 8, nous présentons l'architecture physique et logicielle, utilisé pour le projet Cart-O-Matic, développée avec nos partenaires pour répondre au défi national robotique Carotte. Elle comprend notre approche multirobot de construction de carte, la planification ainsi que le suivi de trajectoires.

  - Le chapitre 9 décrit les résultats expérimentaux obtenus sur des robots réels, notamment durant le défi CAROTTE.

- Enfin, dans le dernier chapitre nous concluons et donnons les perspectives de ce travail.

Première partie

État de l'art et définition du problème

Chapitre 2

État de l'art

Sommaire

L'exploration multirobot est un sujet largement abordé pour les raisons citées en introduction 1.2. Dans ce chapitre nous présentons les principales approches existantes pour l'exploration d'environnements inconnus que nous classons selon les objectifs, les hypothèses et les méthodes utilisées.

2.1 Les pionniers

L'exploration robotique est un champ de recherche actif depuis le début de la robotique au milieu du 20ème siècle. Par exemple, les tortues de Bristol, construites par W.G. Walter en 1949, se comportent comme un animal capable d'explorer aléatoirement son environnement en évitant les obstacles pour chercher des sources lumineuses [1]. L'histoire de l'exploration robotique d'environnement inconnu est lié à celui de la cartographie en robotique. En effet, il est plus facile pour un robot d'explorer un environnement en disposant d'une carte des zones déjà visitées et inversement il est difficile pour un robot de cartographier sans explorer l'environnement. Au milieu des années 1980, le développement de la cartographie robotique (sous forme de grilles (Moravec and Elfes, 1985), de polygones convexes (Chatila and Laumond, 1985) ou de segments (Crowley, 1985)) a permis de diriger l'exploration vers les zones non-visitées. Simultanément, le développement de la robotique comportementale (Brooks, 1985) a permis de réaliser une tâche trop complexe pour les ordinateurs de l'époque (l'exploration et la cartographie) par l'interaction de plusieurs comportements intelligents simples.

En raison de l'imprécision des capteurs et de leur vitesse d'acquisition insuffisante, la localisation et la cartographie posaient encore, il n'y a pas si longtemps, de nombreux problèmes. Les cartes topologiques (Kuipers, 1978) aident à résoudre ce problème, elles modélisent les lieux par les nœuds d'un graphe et les chemins entre ces lieux par des arrêtes. Le travail pionnier de Kuipers and Byun (1991) propose d'explorer en combinant des comportements de navigation réactive dans les couloirs, d'une part, et d'identification et de reconnaissance de lieux distinctifs (tels que les intersections), d'autre part. Leur robot construit une carte topologique dont les nœuds sont des lieux distinctifs. Chaque direction non visitée d'un nœud est ajoutée à un agenda d'exploration. Le robot supprime les nœuds-directions de son agenda quand il quitte le nœud dans la direction correspondante ou quand il arrive sur le nœud de cette direction. Lorsque son agenda est vide, le robot a fini d'explorer son environnement.

Au début des années 1990, le développement technologique des moyens de communication ainsi que leur plus grande disponibilité a permis un gain d'intérêt pour les approches multirobot. Les premières recherches utilisant une approche multirobot pour le déploiement étaient purement réactives. La carte était construite *a posteriori* à partir des informations récoltées par les robots. Par exemple, l'approche de Mantaras et al. (1997) utilise de petits robots à bas coûts qui explorent l'environnement de manière aléatoire (en tournant de $\pm 45°$ ou $\pm 90°$ à des moments aléatoires ou quand ils détectent un obstacle). Les robots retournent à leurs points de départ en suivant le chemin inverse parcouru (moins les boucles) et cela après un certain temps de parcours. La carte est construite en utilisant l'information sensorielle des robots revenus à la zone de départ. Celle-ci étant bruitée, une carte filtrée est construite en fusionnant les murs proches. Leur carte n'inclut que les murs orthogonaux. (Duckett and Nehmzow, 1997) montrent que les robots utilisant la technique du suivi de mur explorent plus rapidement qu'une exploration aléatoire. Leur environnement est construit pour disposer d'amers (points de repère fixes)

1. Cette notion d'exploration ne correspond pas à la définition de l'exploration retenue dans cette thèse qui implique la construction d'une carte.

uniques et non ambigus reconnaissables par les robots. La carte construite donne uniquement la position de ces amers.

Yamauchi (1997) propose une des premières approches qui dirige les robots vers les zones inexplorées, introduisant le concept de frontière. Dans cette approche, les robots construisent une carte commune de type grille dont les cellules sont vides, occupées ou inexplorées. Les robots sont attirés vers les cellules frontières (cellule vide jouxtant une cellule inexplorée) et mettent à jour la grille globale avec les nouvelles informations. Les robots se déplacent vers les frontières et collectent ainsi de nouvelles informations sur les zones inexplorées de l'environnement. L'exploration successive et exhaustive des frontières permet l'exploration complète de l'environnement.

L'exploration multirobot a été abordée avec des **objectifs**, des **hypothèses** et des **méthodes** différentes générant de nombreux modèles et architectures de systèmes très différents. Ces approches peuvent être classées selon cinq types que nous examinons dans ce qui suit.

2.2 Classement par hypothèses

Cette section aborde les différentes hypothèses sur l'environnement et sur les caractéristiques techniques des robots (capteurs et moyens de communication) avec lesquelles l'exploration multirobot a été abordée.

2.2.1 Environnement et initialisation

L'environnement influence toute la stratégie d'exploration (les objectifs, la coordination, la communication et l'identification des cibles). Nous distinguons principalement, les environnements intérieurs, des environnements extérieurs. Les environnements intérieurs sont structurés alors que les environnements extérieurs présentent davantage d'espaces vides. La localisation en extérieur est plus difficile car il y a moins de repères. Pour remédier à ce problème, les robots d'extérieur sont souvent équipés d'un système de localisation externe par exemple un *Global Positioning System* (GPS). En intérieur et dans bien d'autres environnements (sous-marins, extra-terrestres, etc) le GPS n'est pas disponible.

Comme nous l'avons vu en introduction, afin de collaborer, les robots doivent mettre en correspondance les informations récoltées. Pour cela, les robots peuvent utiliser un repère commun. Lorsque les robots partent d'une même zone, il peuvent disposer d'un repère commun ou le définir. S'ils démarrent de zones différentes, sans repère commun, la mise en correspondance de leur carte respective peut se faire par la recherche de la transformation entre leurs cartes (par exemple avec l'approche de Birk and Carpin (2006) avec des grilles d'occupation), l'utilisation d'amers communs, uniques et non ambigus, ou par l'observation des positions relatives entre les robots. Les positions relatives peuvent être obtenues par observation mutuelle avec, par exemple, des marqueurs de couleurs.

2.2.2 Caractéristiques des robots

La puissance de calcul influence aussi le choix de la stratégie. Une stratégie trop coûteuse en calcul sur un robot par rapport à ses capacités nécessitera des temps d'arrêt pour calculer la prochaine destination et les moyens de l'atteindre. L'autonomie du robot dépend de l'énergie embarquée et de sa consommation. Concernant la communication entre les robots plusieurs caractéristiques sont à prendre en compte : la bande passante disponible, la portée et la fiabilité.

Capteurs

La nature des capteurs influence la stratégie de coopération et le choix des cibles à affecter aux robots. Leur précision et leur portée peut déterminer l'objectif : ainsi, si un robot seul peut cartographier assez précisément avec la résolution souhaitée alors les approches cherchant à maximiser la précision ne seront pas utiles. La portée des capteurs doit être adaptée à l'étendue des zones libres de l'environnement. Quand les environnements sont suffisamment petits pour pouvoir observer d'une position des traits discriminants de l'environnement, la localisation est plus simple. Il est plus difficile d'obtenir des informations métriques précises dans de grands environnements ouverts sans un capteur ayant une grande portée. Enfin, le bruit des capteurs influence la méthode de cartographie. Si l'on souhaite créer une carte en trois dimensions, il est nécessaire d'avoir les capteurs adaptés i.e. permettant d'acquérir des informations en 3D. Les caractéristiques des capteurs sont nombreuses : précision, résolution, fréquence d'acquisition, quantité de données générées. Chacune de ces caractéristiques influence directement les méthodes d'exploration, de localisation et de cartographie.

Quantité d'information communiquée

La quantité d'information communicable entre les robots influence directement le choix de la méthode de coordination. En effet, plus le nombre de robots est grand plus la quantité d'informations échangées risque d'être importante, en particulier si les communications se font de robot à robot. Sur tous les réseaux, la bande passante est limitée, une stratégie fondée sur la communication de grand volume d'information entre robots ne sera donc pas applicable avec un grand nombre de robot. Pour un passage à l'échelle d'un grand nombre de robots, il est donc nécessaire d'utiliser une stratégie fondée sur un échange d'informations limité.

Portée de la communication

Certaines techniques utilisent des techniques de rendez-vous pour la confirmation des coordonnées de références. Ceci requiert l'abandon des actions en cours (Roy and Dudek, 2001) (de Hoog et al., 2010). D'autres techniques contraignent les robots à rester à portée de communication (Vazquez and Malcolm, 2004). Ces techniques restreignent la dispersion des robots dans l'environnement et limitent leur exploration vers les différentes zones. Certains robots doivent alors jouer le rôle de relai de façon à maintenir une chaine entre tous les robots (Le, 2010).

Fiabilité de la communication

Pour prendre en compte l'instabilité des communications sans fils, les approches fondées sur la décision autonome du robot sont préférables. En effet, si un robot attend une communication pour décider de sa prochaine cible ou de son chemin mais que celle-ci est perdue, le robot attendra jusqu'à la reprise des communications.

2.3 Classement par objectifs

L'exploration étant définie comme la découverte d'un environnement inconnu, les objectifs de l'exploration multirobot peuvent être différents selon les applications. Pour la recherche d'objet, de personne ou d'incendie par exemple, l'exploration se concentrera en priorité sur les plus grandes zones offrant une meilleure couverture pour augmenter la probabilité de trouver les

objets plus rapidement. En revanche, pour la surveillance, il faudra explorer la totalité de l'environnement plusieurs fois. Dans d'autres scénarios la construction de la carte la plus précise sera souhaitée sans prendre en compte le temps d'exploration, ceci est particulièrement vrai pour les robots utilisant des capteurs très bruités.

2.3.1 Minimisation du temps d'exploration

Parmi les approches qui cherchent à minimiser le temps d'exploration, nous pouvons distinguer deux approches, celles qui cherchent la plus grande couverture rapidement de celles qui cherchent l'exhaustivité de l'exploration le plus tôt possible. Les premières cherchent à explorer rapidement les plus grandes zones en priorité ; l'exhaustivité n'intervenant qu'au terme de l'exploration comme objectif secondaire. Les secondes peuvent, par exemple, chercher à explorer des zones pour éviter d'y revenir ultérieurement. Un problème proche de la couverture exhaustive est la patrouille qui consiste à répéter des couvertures en minimisant le délai entre deux visites de la même zone. C'est un problème connexe qui est en général abordé en fournissant la carte aux robots. S'ils ne la connaissent pas, un premier temps est consacré à sa création et donc à l'exploration. La couverture désigne aussi le problème statique du calcul des configurations des robots permettant l'observation de la plus grande surface possible (par exemple (Renzaglia and Martinelli, 2009)). Ce dernier problème est toutefois plus éloigné de la problématique que nous abordons car il ne considère pas l'exploration totale de l'environnement.

2.3.2 Minimisation de l'énergie

Les robots embarquent une quantité d'énergie limitée. Il peut donc être important d'en limiter la consommation pour l'exploration d'environnements de grande taille (Mei et al., 2006). Toutefois, l'utilisation des moteurs est en général la principale source de consommation d'énergie ; auquel cas l'énergie consommée est proportionnelle à la distance parcourue elle même proportionnelle au temps d'exploration.

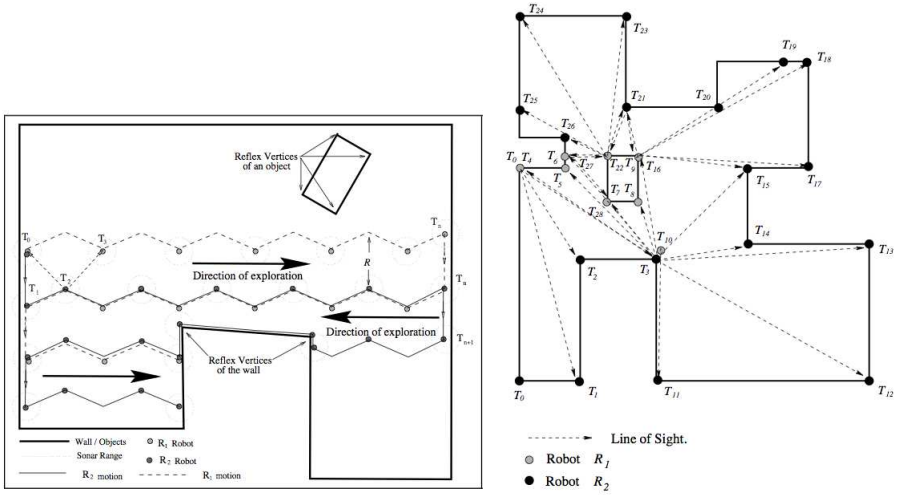
2.3.3 Maximisation de la précision

L'objectif peut être de construire la carte la plus précise possible en utilisant plusieurs robots. Deux approches se distinguent pour satisfaire cet objectif.

La première consiste à améliorer la précision de la localisation et donc de la cartographie en localisant les robots les uns par rapport aux autres. L'approche de Rekleitis et al. (1997) couvre l'environnement avec un robot arrêté qui en observe un autre en déplacement, l'intérieur du triangle ainsi formé entre le robot à l'arrêt, le point de départ et d'arrivée du robot en déplacement est cartographié comme inoccupé. L'exploration est effectuée au niveau local par un balayage de bandes d'espace inoccupé (voir figure 2.1a). Au niveau global, ces bandes sont connectées en utilisant une stratégie de conservation de lignes de mire entre les robots basée sur les coins formant des angles aigus (voir figure 2.1b). Les auteurs montrent ainsi que l'exploration avec plusieurs robots est plus rapide et plus précise qu'avec un seul. Cette approche ne considère pas de limite de distance pour la communication entre les robots et un robot ne peut en observer qu'un autre en mouvement. Rekleitis et al. (2001) utilisent une collaboration entre robots capables de mesurer leurs distances et angles relatifs : un ou plusieurs robots à l'arrêt observent le déplacement d'un autre robot. Dans ce système, la moitié des robots de la flottille se déplace pendant que l'autre moitié observe la position des robots en mouvement. Les robots statiques sont mieux localisés et peuvent donc transmettre précisément les positions finales ob-

servées aux robots mobiles. Les robots alternent entre ces comportements pour progresser dans l'environnement.

Cette approche présente comme inconvénient de maintenir à l'arrêt la moitié de la flottille à tout moment, ce qui ralentit fortement la progression de l'exploration.



(a) Découpe en bandes d'espace inoccupé (b) Conservation des lignes de mire

FIGURE 2.1 – Stratégie locale et globale Rekleitis et al. (1997)

La seconde approche consiste à combiner le SLAM (*Simultaneous Localization and Mapping*) avec l'exploration. Le SLAM est une technique permettant de construire une carte de l'environnement tout en localisant le robot dans cette carte. Ceci est fait de manière répétitive à chaque observation. Ces approches sont monorobot mais sont applicables au multirobot. Par exemple, Makarenko et al. (2002) alternent entre l'exploration et le retour vers des régions connus pour réduire l'incertitude. Feder et al. (1999) proposent de guider le robot vers des amers de façon à réduire l'incertitude. Enfin, des approches telle que l'Active SLAM forcent les robots à retourner vers des zones connues avec une grande certitude quand l'incertitude sur leur localisation franchit un certain seuil. Dans (Stachniss et al., 2005), la fermeture de boucle (confirmation de l'hypothèse que le robot est revenu dans un lieu déjà visité) se prolonge tant que l'incertitude n'est pas passée en dessous d'un seuil, avant l'exploration de nouveaux espaces.

Les approches visant à augmenter la précision de la carte nécessitent l'abandon de la tâche d'exploration afin de mieux localiser les robots. Nous nous concentrons par la suite sur la tâche d'exploration. Celle-ci nécessite l'affectation de cibles aux robots pour qu'ils découvrent l'environnement. La section suivante décrit les différentes méthodes existantes d'identification des cibles.

2.4 Classement par méthodes d'identification des zones à explorer (cibles)

Lors de l'exploration, l'identification des cibles à atteindre est fortement liée à la dimension de la représentation de l'environnement (2D, 3D) et à son type (carte topologique, grille d'occupation, triangulation). Nous décrivons ici succinctement les grandes familles d'identification des cibles. Nous verrons plus en détail leur identification dans le chapitre 7.

2.4.1 Approches par frontière

Les frontières sont des zones d'intérêt pour l'exploration. Elles délimitent les zones connues des zones inconnues. L'affectation des robots aux frontières nécessite que ceux-ci se coordonnent afin que la répartition des robots aux frontières soit la plus appropriée possible (présentée en section 2.5). Les frontières sont définies comme un ensemble de cellules contiguës. Avec une représentation de carte sous forme de grille, l'identification des frontières s'effectue en regroupant les cellules frontières voisines en une frontière unique (Balakirsky et al., 2007). Les approches fondées sur une représentation topologique de la carte identifient une frontière par un nœud topologique, comme illustré figure 2.2.

Ces approches, comme celles fondées sur l'utilité (voir section 2.4.3), visent à éviter l'assignation de plusieurs robots à des cellules frontières proches l'une de l'autre.

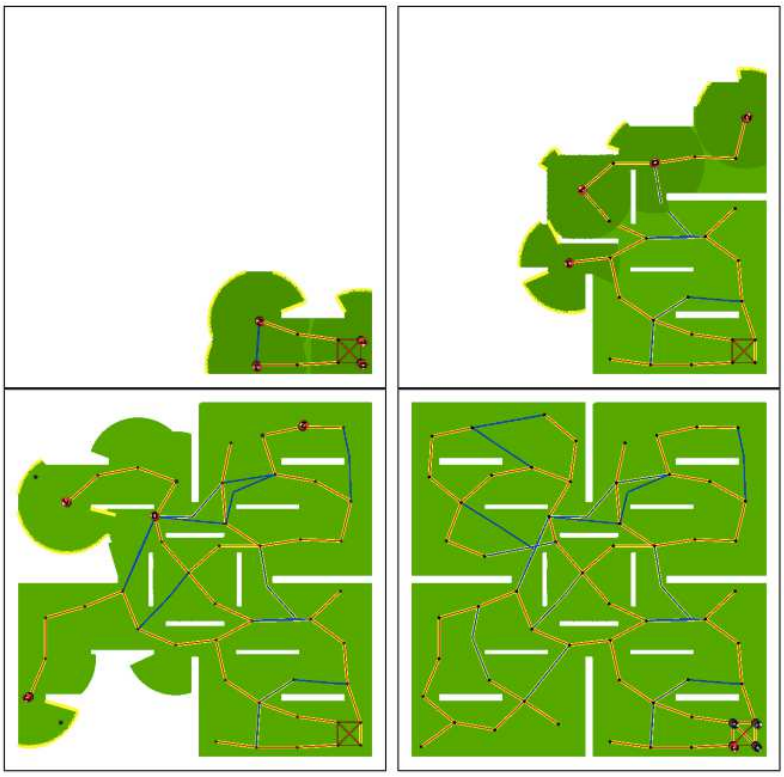


FIGURE 2.2 – Graphe d'exploration de frontière (Franchi et al., 2009)

L'approche par frontière est simple à mettre en œuvre et efficace. L'approche proposée dans cette thèse est fondée sur une telle approche.

2.4.2 Approches *Next Best View*

González-Banos and Latombe (2002) échantillonnent de manière probabiliste l'environnement proche des frontières pour déterminer la prochaine meilleure position d'observation (*Next Best View*) de la frontière. La figure 2.3 illustre l'évaluation d'une position candidate. Cette méthode est difficilement applicable en multirobot car elle nécessite d'échantillonner plusieurs positions et de les évaluer. Ce procédé est déjà coûteux en monorobot et l'analyse combinatoire

des positions échantillonnées la rend trop complexe. De plus, Holz et al. (2010) évaluent qu'en monorobot cette approche est moins efficace en temps d'exploration que les approches frontières.

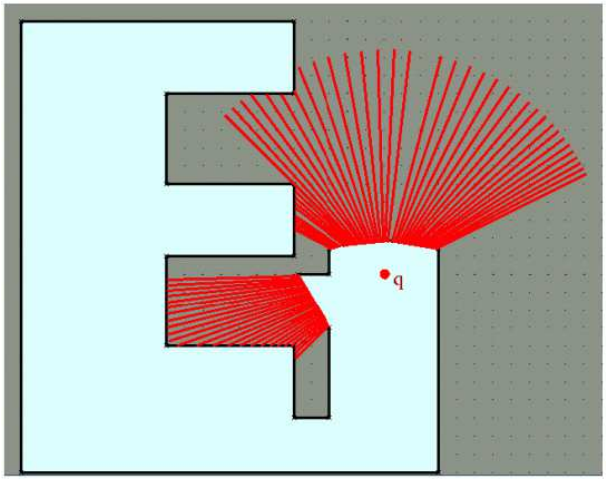


FIGURE 2.3 – Evaluation d'une position candidate comme cible d'exploration de frontière. L'aire visible en dehors des zones explorées est évaluée par un lancer de rayons depuis la position candidate (González-Banos and Latombe, 2002)

2.4.3 Approches par Utilité

Le concept d'utilité, initialement proposé par Simmons et al. (2000), est défini comme la différence ou le rapport entre le coût et le gain d'information lié à une observation. Il réduit par exemple l'exploration de frontières voisines à celles déjà assignées, car elle n'apporte que peu d'information. En effet, un robot qui observe une frontière observera aussi son voisinage. Ainsi, l'utilité de l'exploration d'une frontière est inversement proportionnelle à sa distance aux frontières assignées. L'utilité des frontières est ainsi réduite si elles sont visibles (non séparées par un obstacle et à une distance inférieure au rayon de perception) depuis la frontière assignée. Si des frontières sont assez proches, un seul robot pourra les explorer lors d'une même observation. L'utilité permet d'obtenir une meilleure répartition des robots dans l'environnement que les approches frontières ne comportant pas de regroupement des cellules frontières voisines.

2.4.4 Autres approches

Les approches de Wurm et al. (2008) et de Gossage et al. (2006) discrétisent la carte partiellement construite en cellules de Voronoï et assignent les cellules frontières aux robots. La figure 2.4 illustre la construction du graphe topologique fondée sur la discrétisation de Voronoï. Les nœuds de la carte topologique sont les nœuds associés aux cellules de Voronoï. Les robots sont affectés aux cellules inexplorées. Ces méthodes sont proches des méthodes utilisant l'utilité et le regroupement des cellules frontières car elles évitent aussi l'assignation de plusieurs robots à des frontières proches.

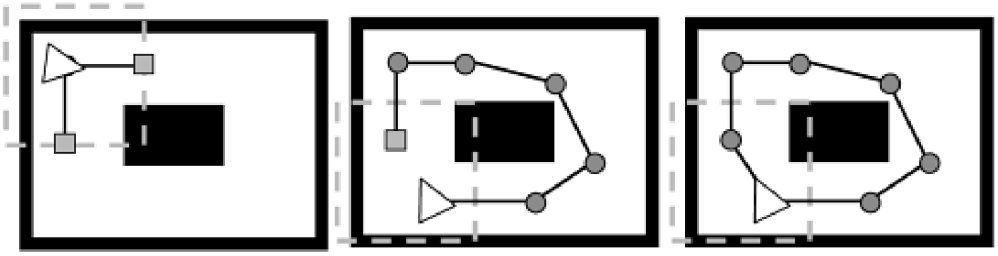


FIGURE 2.4 – Graphe fondé sur la discrétisation de Voronoï (Gossage et al., 2006). Le triangle représente le robot, les carrés sont les nœuds inexplorés et les disques sont les nœuds explorés. Les pointillés montrent la perception du robot.

L'approche de Puig et al. (2011) effectue une segmentation de l'environnement avec l'algorithme *K-means* en autant de zones que de robots. Ainsi, une zone sera explorée par chaque robot. La figure 2.5 illustre la segmentation. Cette méthode répartit les robots dans l'environnement pour une exploration plus rapide. Son inconvénient est qu'elle nécessite de connaître la taille de l'environnement *a priori*. De plus, elle ne prend pas en compte la topologie de l'environnement, ce qui posera beaucoup de problème si l'environnement est structuré car les robots devront éventuellement explorer une zone affectée à un autre robot avant de pouvoir accéder à la zone qui lui a été affectée.

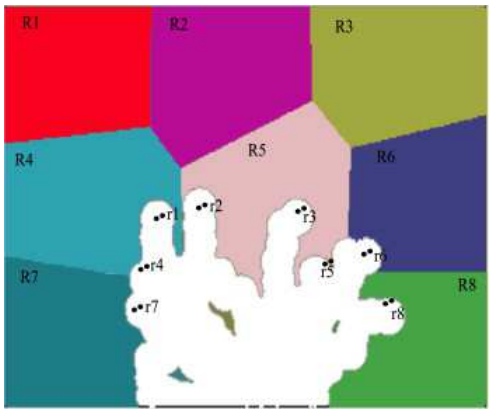


FIGURE 2.5 – Segmentation *a priori* de l'environnement avec l'algorithme *K-means*, K étant égal au nombre de robots. Une zone à explorer est ensuite affectée à chaque robot. (Puig et al., 2011)

2.5 Classement par méthode d'affectation de cibles

Les cibles peuvent être affectées de manières différentes aux robots suivant la méthode d'exploration. Nous résumons ici les principales méthodes.

2.5.1 Frontière la plus proche

En monorobot , la méthode proposée par Yamauchi (1997), affectant le robot à la frontière la plus proche, est la méthode la plus efficace des approches frontière (voir(Holz et al., 2010)).

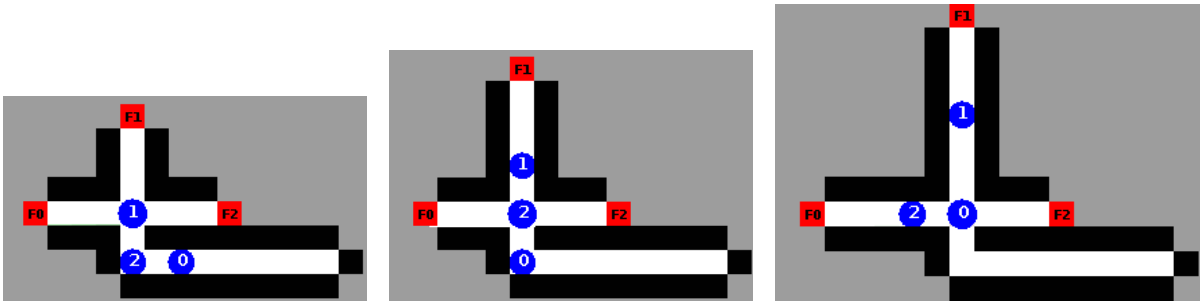


FIGURE 2.6 – Coordination implicite : trois robots sortent d'un couloir exploré et se répartissent de manière équilibrée dans les trois couloirs. Les frontières sont de couleur rouge, les robots de couleur bleu. Chaque robot se dirige vers la frontière la plus proche de lui.

Yamauchi (1998) étend cette méthode au cas multirobot i.e. chaque robot se dirige vers sa frontière la plus proche. Lors de son application, si la carte partiellement construite est partagée, une coopération implicite s'effectue (voir la figure 2.6) mais elle est limitée car les robots peuvent choisir des frontières identiques, ne tirant pas avantage de leur nombre.

2.5.2 Glouton

L'algorithme Glouton tente d'optimiser le coût total de l'exploration en fixant à chaque itération la paire robot-frontière ayant le coût minimum. Comme l'itération est effectuée sur les robots, l'équilibre de la répartition des robots sur les frontières est garanti. Cet algorithme est en général appliqué de manière centralisée. Toutefois, chaque robot, s'il dispose des coûts de déplacement de tous les robots vers chaque frontière, peut exécuter l'algorithme jusqu'à obtenir son assignation (voir la section 4.1.2 pour le détail de l'algorithme). La figure 2.7 illustre une allocation de frontière avec l'algorithme Glouton. La plupart des approches d'allocation de frontières sont fondées sur cet algorithme (Burgard et al., 2005), (Simmons et al., 2000) et (Zlot et al., 2002).

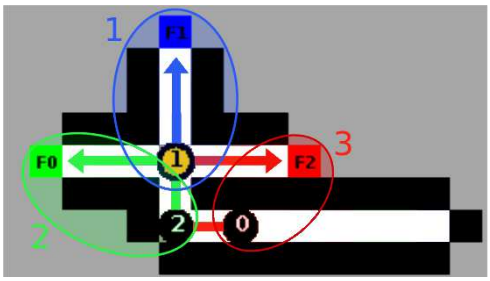


FIGURE 2.7 – Illustration de l'allocation de frontière avec l'algorithme glouton. Les couples robot-frontière sont assignés par ordre de coût du plus faible au plus élevé

2.5.3 Glouton sur l'utilité des frontières

Burgard et al. (2005) utilisent l'algorithme Glouton en ajoutant un critère d'importance à chaque frontière. En effet, l'algorithme est semblable au Glouton avec la mise à jour du critère d'importance après chaque assignation. Le but de cette approche est de minimiser le temps

d'exploration en prenant en compte simultanément le coût pour atteindre le cible ainsi que son apport en gain d'information (Burgard et al., 2002).

Le principe général de l'algorithme est le suivant :

1. initialiser l'utilité de chaque frontière à 1 ;
2. trouver la paire robot-frontière ayant la différence utilité-coût maximum et assigner le robot à la frontière correspondante ;
3. diminuer l'utilité de toutes les frontières en fonction de leur distance à la frontière précédemment assignée (plus elles sont proches plus leur utilité est diminuée) ;
4. aller à l'étape 2 s'il reste un robot sans frontière.

Cette méthode permet de mieux distribuer les robots mais elle est peut difficilement être décentralisée car l'affectation d'un robot influe sur l'affectation d'un autre robot. De plus, l'affectation de tous les robots doit se faire simultanément, posant un problème de synchronisation.

2.5.4 Méthode hongroise

La méthode hongroise (Kuhn, 1955) est un algorithme d'optimisation combinatoire classiquement utilisé pour l'affectation de tâche. Il s'exécute directement sur une matrice de coût. Il permet de résoudre optimalement la minimisation de la somme des coûts d'exploration de chaque tâche. Sa complexité est en $O(n^3)$ où n est le nombre maximum entre le nombre de robot et le nombre de frontières. Il est adapté pour l'affectation de frontières aux robots et a souvent été utilisé pour l'exploration multirobot (Wurm et al., 2008) (Ko et al., 2003).

2.5.5 Affectation optimale

La minimisation de la somme des coûts effectuée par les algorithmes Glouton et Hongrois ne permet pas l'affectation optimale des robots sur les frontières (voir figure 2.8). Pour obtenir de meilleurs résultats, il peut être nécessaire d'énumérer les solutions car c'est un problème combinatoire concurrentiel.

Lorsque le nombre de robots est petit (inférieur à dix), il est possible de générer toutes les combinaisons robot/frontière possibles et de déterminer laquelle est optimale. Lorsque le nombre de robot est plus important, l'utilisation du deuxième algorithme proposé dans Burgard et al. (2005) est préférable, celui-ci consistant en une recherche aléatoire combinée à une méthode d'optimisation locale. Toutefois le nombre de robots doit rester raisonnable suivant la puissance de calcul de l'ordinateur central ; d'après les auteurs, cette méthode calcule l'affectation de vingt robots en quelques secondes et en plus d'une minute pour quarante robots.

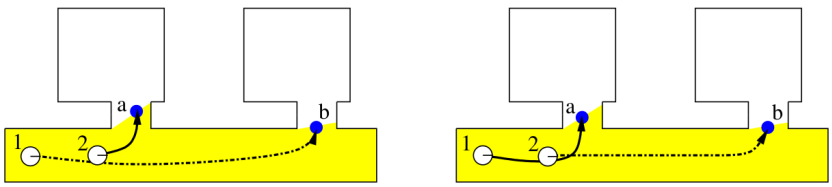


FIGURE 2.8 – Situation dans laquelle la minimisation de la somme des coûts comporte deux solutions. Les robots exécutant leurs tâches en parallèle, le temps nécessaire pour découvrir les deux frontières (a) et (b) est déterminé par la trajectoire la plus longue. L'affectation à gauche est donc sous-optimale. (Burgard et al., 2005)

L'implémentation de ces méthodes d'affectation de cibles ne peut s'effectuer sans communication entre les robots. La section suivante décrit ces méthodes de communications.

2.6 Classement par méthode de communication

Les premiers robots n'étant pas capables de communiquer sans fils la coordination entre robots était difficile à mettre en place. Le développement des communications sans-fils a permis de développer de meilleures stratégies de coordination que nous allons étudier dans cette section.

2.6.1 Sans communication

La communication est essentielle pour garantir la complétude de la couverture. Les méthodes d'exploration aléatoire et de suivi de mur (voir section 2.1) n'utilisent pas de communication mais il n'y a pas de coopération entre les robots et rien ne permet de déterminer si l'exploration est complète. En pratique, la méthode de Mantaras et al. (1997) utilise une communication entre les robots qui se croisent pour ne pas perdre les informations récoltées par les robots qui ne reviennent pas au point de départ. Pour la couverture d'environnement, Howard et al. (2002) utilisent une technique fondée sur les champs de potentiels (Khatib, 1986). Ces champs repoussent les robots entre eux et des obstacles. Un schéma moteur (Arkin, 1989) est utilisé pour générer une action en fonction des champs de potentiels auquel le robot est soumis. La seule hypothèse de l'approche est que les robots perçoivent la distance et l'angle des obstacles et des robots proches grâce à leurs capteurs. En partant de la même zone, les robots se dispersent dans l'environnement pour maximiser la couverture (voir figure 2.9). Ces méthodes ne construisent pas de carte ou construisent la carte *a posteriori*. Elles sont peu efficaces pour l'exploration, en effet, il est nécessaire de diriger les robots vers les zones inexplorées. Pour effectuer cela en collaboration, la communication est indispensable. Les robots qui ne communiquent pas devront chacun explorer la totalité de la carte pour déterminer que l'exploration est achevée.

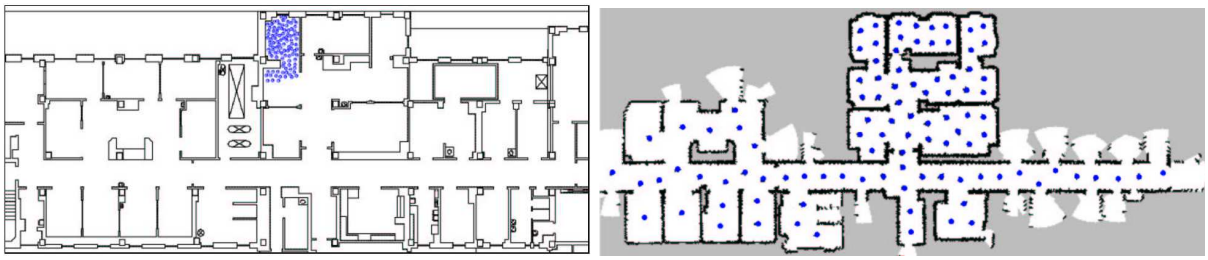


FIGURE 2.9 – À gauche, configuration initiale des robots. À droite, configuration finale maximisant la couverture (Howard et al., 2002)

2.6.2 Communication indirecte

Les approches sans communication directe entre robots sont fondées sur le marquage de l'environnement, par exemple, avec des phéromones digitales (Glad et al., 2009) en déposant des marques/balises. L'environnement étant un graphe prenant souvent la forme d'une grille de cellules, les robots laissent des marques contenant une valeur dans les cellules visitées. L'exploration s'effectue avec des agents-fourmis (agents très simples avec perception très limitée). Par exemple, les méthodes LRTA* (*Learning Real-Time A**) (Korf, 1990) et *Node counting* (Wagner

et al., 1999) utilisent une stratégie identique : les agents se dirigent vers la cellule voisine ayant la valeur la plus faible. Ces algorithmes diffèrent sur la méthode de mise à jour des valeurs des marqueurs. LRTA* guide les robots vers les cellules inexplorées les plus proches d'une manière similaire à celle de (Yamauchi, 1998). Avec l'algorithme *Node counting*, chaque cellule de l'environnement tient à jour le nombre de fois où la cellule a été visitée par un robot, les robots se dirigent donc vers la celulle voisine la plus anciennement visitée. Koenig et al. (2001) montrent que LRTA* garantit d'explorer l'environnement en temps polynomial tandis que *Node counting* a une complexité en temps exponentiel. Ces approches sont simples à simuler mais difficiles à mettre en œuvre sur des robots. Par ailleurs, les agents ne construisent pas de carte mais remplissent une grille ; ils ne peuvent pas percevoir si l'exploration est terminée (perception limitée et pas de mémoire).

2.6.3 Communication explicite

La communication explicite d'informations et de données entre les robots peut prendre différentes formes. En effet, les robots peuvent échanger explicitement des informations sur l'environnement, leur permettant de se coordonner implicitement. Ceux-ci peuvent également échanger directement des informations de coordination.

Partage de la carte / Coopération implicite

Dans (Yamauchi, 1998), la coordination est qualifiée d'implicite car elle n'utilise pas d'échanges de messages de coordination entre robots. Elle résulte de la construction collaborative de la carte globale. Les robots partagent les informations locales récoltées afin que chacun d'entre eux produise une carte similaire, fournissant une liste de frontières similaires. Chaque fois qu'un robot effectue une observation, il en diffuse (*broadcast*) le résultat à l'ensemble de la flottille. Aucune communication n'est utilisée pour coordonner explicitement les robots. C'est un système asynchrone et distribué. Chaque robot décide de façon autonome dans quelle direction il doit se diriger.

Communication explicite pour la coordination

Le cas d'une communication explicite entre plusieurs robot a donné lieu au développement d'un certain nombre de méthodes dont les principes sont expliqués ci-après.

Système d'enchères Les systèmes d'enchères fonctionnent en deux temps, tout d'abord une phase d'enchères basée sur des communications soit entre robots soit avec un serveur central, suivie d'une phase d'assignation. L'assignation de frontière utilise généralement le même principe que l'algorithme Glouton. En pratique, chaque robot émet une enchère sur chaque frontière, il remporte une frontière pour laquelle il a la meilleure enchère. Zlot et al. (2002) et Chaimowicz et al. (2002) ont développé une méthode distribuée où les robots qui découvrent une frontière sont commissaires-priseurs pour celle-ci, les autres robots sont enchérisseurs.

Stratégie par groupes de robot Dans l'approche de Franchi et al. (2009), la coopération s'effectue en partageant une carte topologique de l'environnement. Les robots choisissent de façon autonome leur cible mais se coordonnent lorsque les cibles ou les chemins sont en conflit. La coordination se déclenche entre les robots dont l'inter-distance entre leurs cibles est inférieure à deux fois le rayon de perception. La coordination s'effectue alors parmi la chaine de robots répondant au critère d'inter-distance de la façon suivante : lors de la phase de coordination, les robots arrêtent leur déplacement, se synchronisent et décident dans ce sous-groupe de la stratégie de coordination. Si les chemins menant aux cibles sont en conflit, un leader du groupe est élu

et résout le conflit en assignant les cibles ou en laissant certains robots à l'arrêt. La figure 2.10 illustre un groupe de robots se coordonnant.

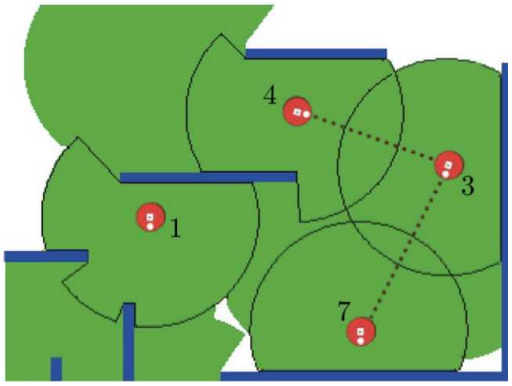


FIGURE 2.10 – Constitution du groupe de robots engagé dans une phase de coordination (Franchi et al., 2009)

Approches centralisées

De nombreuses approches utilisent une unité centrale qui récolte les informations provenant de chaque robot, construit la carte partielle, calcule les coûts pour chaque robot et distribue enfin des tâches à chaque robot (Wurm et al., 2008). Ainsi, la communication est centralisée vers le seul et unique robot leader. Durant l'exploration, le rôle de leader peut être attribué à différents robots.

D'autres approches distribuent une partie du calcul. Ainsi, Simmons et al. (2000) proposent une approche fondée sur l'exploration de frontières et un système d'enchères simple. Chaque robot évalue le profit de l'exploration de chaque frontière en déterminant son coût de déplacement et une estimation du futur apport d'information. Il envoie ensuite une enchère sur les frontières évaluées à un agent central qui, une fois les enchères de tous les robots reçues, assigne les frontières aux robots de manière gloutonne en évitant la superposition de la couverture de chaque robot. Stachniss et al. (2008) utilisent le même genre d'approche mais fournissent des informations *a priori* sur la topologie de l'environnement permettant de prendre en compte le type de région à explorer (couloir, pièce, etc). Notons que ces dernières approches sont partiellement centralisées mais elles comportent les mêmes défauts qu'une approche centralisée car elles contiennent un point de centralisation pouvant être une source de défaillance du système.

2.7 Conclusion sur l'état de l'art

L'exploration multirobot a été abordée avec des objectifs différents : précision de la carte construite, couverture maximale et exploration complète en un temps minimum. Quelque soit l'objectif recherché, nous avons vu que pour développer une stratégie d'exploration efficace, il faut considérer la portée des capteurs et la portée des communications entre les robots. Les stratégies centralisées permettent une exploration efficace grâce à une coopération efficace mais elles sont difficilement applicables à grande échelle. Les approches sans communication ou avec une communication indirecte permettent une coopération mais ne construisent pas de carte et ne peuvent pas garantir la complétude de l'exploration. Les stratégies fondées sur la coordination explicite induisent un coût de communication qui ne permet pas de passer à l'échelle d'un

grand nombre de robots. La coordination implicite, proposée par Yamauchi (1998), est un bon compromis entre communication et coopération, mais ses performances sont bien en dessous des approches utilisant une coordination explicite. Nous nous intéressons dans cette thèse à l'exploration complète d'environnements en un temps minimum et nous proposons une stratégie fondée sur la coordination implicite, ayant des performances similaires aux approches de coordination explicite.

Le problème abordé est très riche dans ses variantes et contraintes. Dans le chapitre suivant nous précisons l'énoncé du problème que nous traitons et ses variantes.

Chapitre 3

Problèmes de l'exploration : contraintes et critères d'évaluation

Sommaire

Ce chapitre définit le problème de l'exploration d'un environnement inconnu borné avec un ensemble de robots mobiles autonomes. Nous détaillons les étapes nécessaires à sa résolution ainsi que les contraintes que nous prenons en compte, notamment celles du défi robotique Carotte. Nous donnons les hypothèses retenues sur les robots et la communication, enfin, nous définissons les critères d'évaluation des stratégies d'exploration.

3.1 Exploration par construction de carte

En général, l'exploration consiste à naviguer dans un environnement *a priori* inconnu de façon à le percevoir et à en construire une représentation/carte exhaustive. Les étapes successives nécessaires à la réalisation d'une carte d'un environnement sont illustrées par la figure 3.1. Lors de la phase de perception les informations disponibles sur l'environnement dans l'état actuel du ou des robots sont récupérées. Celles-ci sont ensuite fusionnées avec les informations précédemment récoltées dans l'étape de construction de carte (cartographie). La carte partiellement construite guide le choix du prochain mouvement qui permettra d'obtenir de nouvelles informations de perception.

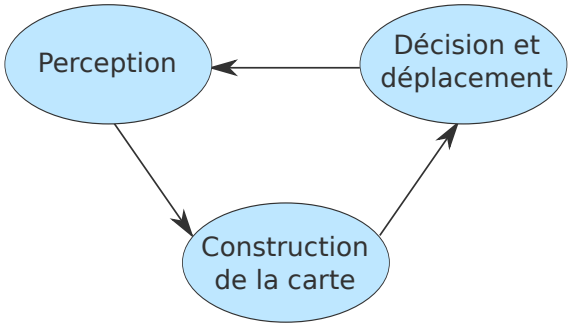


FIGURE 3.1 – Schema général de l'exploration par un ou plusieurs robots

En général, l'exploration d'un environnement inconnu requiert la résolution de trois sous-problèmes : la cartographie, la localisation et la décision de déplacement.

- La **cartographie** résulte de l'intégration des données collectées par les capteurs du ou des robots.
- La **localisation** consiste à estimer la position du robot.
- La **décision de déplacement** doit guider le robot sur une trajectoire ou vers un lieu donné.

Le diagramme 3.2 illustre ces trois sous-problèmes et leurs intersections. L'exploration est la composition de deux sous-tâches, la cartographie et le contrôle du mouvement intégrant la stratégie de déploiement spatial des robots. Le SLAM est une estimation simultanée de la carte et de la position dans cette carte. La localisation active guide le robot vers les lieux lui permettant de mieux se localiser. Au contraire l'exploration guide le robot vers des lieux inconnus pour construire une carte. Le centre du diagramme représente les approches abordant simultanément les trois problèmes.

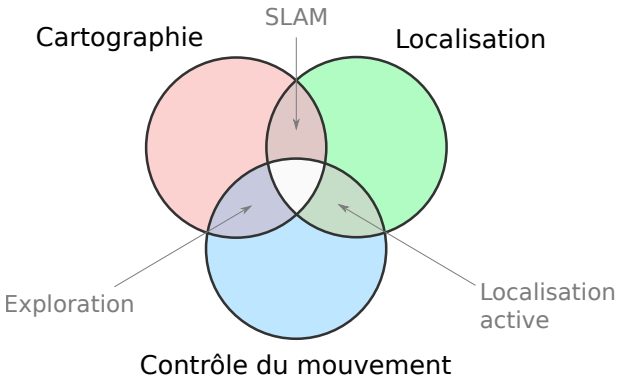


FIGURE 3.2 – Les trois taches nécessaires à la création d'une carte : localisation, cartographie et mouvement. Leur superposition représente la combinaison des tâches Stachniss et al. (2005).

Dans cette thèse nous abordons plusieurs de ces sous-problèmes :

- la stratégie de déploiement ou d'exploration (chapitre 4),
- la cartographie (7 et la section 8.4),
- le contrôle du mouvement (chapitre 8).

Afin de générer une carte exhaustive de l'environnement, les robots doivent explorer toutes les zones inconnues de celui-ci. Pour cela, les mouvements des robots résultent du choix des cibles à atteindre et leurs affectations. Comme nous l'avons vu précédemment, afin d'assurer le succès

et la sureté de la mission, les mouvements effectués doivent permettre l'évitement d'obstacles et de collision.

La localisation n'est pas abordée à proprement dit dans la thèse. Nous supposons qu'un module de SLAM est disponible. La cartographie nécessite que le robot soit localisé. Sans système de localisation externe, le robot doit se localiser pour pouvoir cartographier et doit disposer d'une carte pour se localiser. Ces problèmes sont donc souvent abordés simultanément. Les robots découvrant un environnement inconnu doivent construire une carte de cet environnement tout en s'y localisant de façon incrémentale à l'aide des nouvelles informations perçues à chaque étape. Cette technique est nommée SLAM en anglais pour 'Simultaneous Localization and Mapping' et est détaillée en section 8.4. Dans cette première partie, nous considérons que chaque robot peut se localiser et cartographier indépendamment des autres robots. Chaque robot construit sa propre carte précise et communique uniquement les informations nécessaires aux autres robots pour qu'ils sachent quelles zones sont explorées et quelles directions restent à explorer. Le SLAM multirobot utilise les ressources disponibles sur chaque robot. On peut différencier les approches en les classant selon le niveau auquel le partage est effectué et par la nature, la quantité et la qualité (les données pouvant être pré-filtrées) des informations fournies au filtre qui estime la carte globale.

3.2 Stratégie d'exploration

L'exploration d'un environnement inconnu par plusieurs robots nécessite l'élaboration d'une stratégie de déploiement. Celle-ci répond à l'optimisation d'au moins un critère, dans notre cas nous cherchons à minimiser le temps d'exploration de l'environnement. Pour cela la répartition des robots dans l'environnement permet l'exploration simultanée de zones distinctes. Plus précisément, la stratégie détermine des cibles à atteindre pour chaque robot permettant l'exploration efficace de zones non-visitées distinctes.

Par souci de clarté, nous abordons, dans un premier temps, la problématique de l'exploration d'un point de vue central - une seule instance décide de la direction des robots - et montrons ensuite comment cette décision peut être prise de manière décentralisée - sur chaque robot (en particulier dans le chapitre 5).

Pour pouvoir explorer un environnement de façon autonome, les robots doivent pouvoir naviguer de façon sûre, c'est-à-dire sans heurter les obstacles ou se bloquer. De plus, ils doivent pouvoir se localiser et garder en mémoire les zones explorées.

Pour pouvoir coopérer pendant l'exploration de l'environnement, les robots doivent pouvoir communiquer entre eux. En effet, comme nous l'avons vu dans le chapitre précédent, sans communication chaque robot devra explorer la totalité de l'environnement puisqu'il n'aura pas d'information sur les zones déjà explorées par les autres. Malheureusement, dans de nombreuses applications la communication n'est pas fiable et la bande passante est limitée. Ceci peut être dû à des interférences qui engendrent des coupures de communication et à des distances entre les robots qui peuvent être grandes. Ainsi, pour la réalisation d'un système robuste, il est important que les robots ne soient pas dépendants de la communication pour prendre des décisions. Pour assurer la communication, il faut limiter la bande passante nécessaire afin d'éviter les congestions et de maintenir au maximum la stabilité de la communication.

3.3 Énoncé formel de l'exploration

Dans cette section, nous donnons un énoncé formel de la problématique de l'exploration multirobot. Pour cela, nous donnons un ensemble de définition des différents paramètres pris en compte, ensuite nous donnons les notations utilisées dans le reste du document et, enfin, nous illustrons celles-ci par un exemple.

3.3.1 Définitions

Soit $\mathcal{E}$ l'environnement, $\mathcal{E}_{exp}$ la partie explorée de l'environnement, et $\mathcal{E}_{inexp}$ la partie inexplorée de l'environnement. Soit $\mathcal{R}$ l'ensemble des robots, $\mathcal{R}_1 ... \mathcal{R}_n$ avec n le nombre de robots $|\mathcal{R}|$.

L'exploration est terminée lorsque :

$$\mathcal{E}_{exp} = \mathcal{E} \; et \; \mathcal{E}_{inexp} = \varnothing$$

Mais ceci n'est vrai que dans le cas où les robots peuvent observer la totalité de l'environnement or l'intérieur d'un obstacle n'est pas observable. Il peut également exister une partie de l'environnement qui n'est pas accessible pour les robots, par exemple derrière une vitre. De plus, les robots ne connaissent pas la taille de l'environnement qu'ils explorent, ils ne peuvent donc pas déterminer à quel moment l'exploration est terminée en utilisant la définition donnée ci-dessus. Pour un observateur extérieur, la définition ci-dessus est valide si l'environnement $\mathcal{E}$ est défini comme l'ensemble des parties observables de l'environnement. Toutefois les robots doivent déterminer à quel moment l'exploration est terminée.

Les robots effectuent des observations de l'environnement. Notons $X_i(t)$ la configuration du robot i au temps t et $\mathcal{E}_{obs}(X_i(t))$, la partie observée par le robot i au temps t dans la configuration $X_i(t)$. La partie explorée de l'environnement est alors égale à l'union de toutes les parties observées par tous les robots durant toute l'exploration :

$$\mathcal{E}_{exp} = \bigcup_{i=0}^{n} \bigcup_{t=0}^{T} \mathcal{E}_{obs}(X_i(t))$$

Nous considérons que l'exploration est complète lorsque toutes les zones observables l'ont été c'est-à-dire qu'il n'y a plus de configuration atteignable par un robot qui permette d'observer une zone inexplorée.

$$\nexists X_i \mid \mathcal{E}_{obs}(X_i) \cap \mathcal{E}_{inexp} \neq \varnothing$$

3.3.2 Métriques de performance

Afin d'évaluer la qualité et la validité des méthodes proposées, nous utilisons les métriques décrites ci-après, basées sur la durée et l'exhaustivité de l'exploration.

Temps d'exploration

La métrique utilisée pour mesurer la performance d'une observation est le temps d'exploration c'est-à-dire, du début de l'exploration quand les robots commencent à naviguer jusqu'à la fin de l'exploration quand tout l'environnement a été exploré.

Exhaustivité de l'exploration

La méthode d'exploration doit garantir la complétude. Aussi, chaque robot doit connaître l'état d'avancement de l'exploration pour savoir quand il doit rentrer à la zone de départ. Cette information est globale et nécessite le partage d'information entre les robots.

$$Minimiser\ t \mid \nexists X_i(t) \mid \mathcal{E}_{obs}(X_i(t)) \cap \mathcal{E}_{inexp} \neq \varnothing$$

3.3.3 Contraintes du multirobot

En monorobot, la seule étape spécifique à l'exploration est l'identification de la cible ou de l'état qui permettra d'observer de nouvelles zones de l'environnement et d'apporter de nouvelles informations sur ces zones au module de construction de carte.

En multirobot, l'exploration comporte plusieurs étapes spécifiques illustrées figure 3.3. La première est l'identification des cibles qui peut être différent du choix en monorobot. La deuxième est l'affectation des cibles aux robots pour qu'ils se répartissent dans l'environnement. La troisième est le calcul des trajectoires permettant à chaque robot d'atteindre sa cible tout en minimisant les interactions (gène) entre les robots. Concernant cette troisième spécificité, nous l'aborderons peu car notre stratégie de déploiement dans l'environnement force une dispersion spatiale évitant globalement les conflits de trajectoire entre robot.

Lors de la navigation, les robots doivent aussi s'éviter, en effet même si l'affectation de frontières leur permet de se répartir dans l'environnement il est possible que plusieurs robots se dirigent vers la même zone ou bien vers des zones distinctes uniquement accessibles par un chemin commun. La planification de trajectoire devra donc prendre en compte la position des autres robots qui sont des obstacles dynamiques. La planification de trajectoire multirobot est complexe, particulièrement sans un planificateur centralisé. Pour la simplicité du système, il faut donc soit prendre en compte les trajectoires déjà planifiées et communiquées par les autres robots, soit être réactif à la découverte d'obstacles imprévus lors de l'exécution de la trajectoire.

En résumé, une stratégie d'exploration comprend trois étapes :

1. fixer une cible ou un état à atteindre,
2. planifier un chemin ou trajectoire,
3. suivre le chemin ou exécuter la trajectoire.

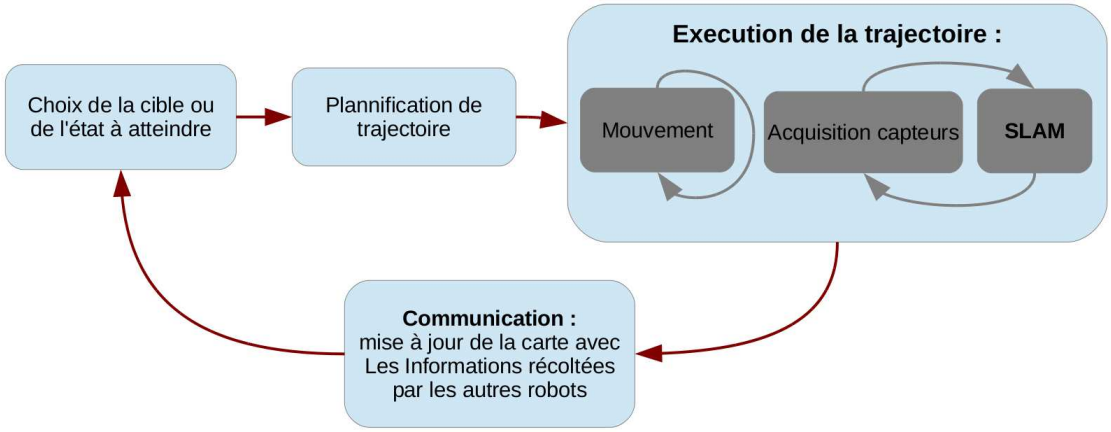


FIGURE 3.3 – Schéma général de l'exploration multirobot

3.3.4 Notations

Dans le reste du document, nous utiliserons les notations suivantes :
- soit $\mathcal{R}$ l'ensemble des robots, $\mathcal{R}_1 ... \mathcal{R}_n$ avec n le nombre de robots $|\mathcal{R}|$
- soit $\mathcal{F}$ l'ensemble des frontières, $\mathcal{F}_1 ... \mathcal{F}_m$ avec m le nombre de frontières $|\mathcal{F}|$
- soit $\mathcal{C}_{ij}$ le coût associé à l'assignation du robot $\mathcal{R}_i$ à la frontière $\mathcal{F}_j$
- soit $\mathcal{A}$ une matrice d'assignations composée de $\alpha_{ij} \in [0, 1]$ tel que :

$$\alpha_{ij} = \begin{cases} 1 & \text{si le robot } \mathcal{R}_i \text{ est assigné à } \mathcal{F}_j \\ 0 & \text{sinon} \end{cases}$$

3.3.5 Exemple d'environnement et de matrice de coûts

La figure 3.4 illustre un exemple d'exploration d'un environnement à l'aide de trois robots. Cet environnement est constitué de quatre frontières (de couleur rouge, bleu, verte et violette). Chaque robot se dirige vers la frontière la plus proche.

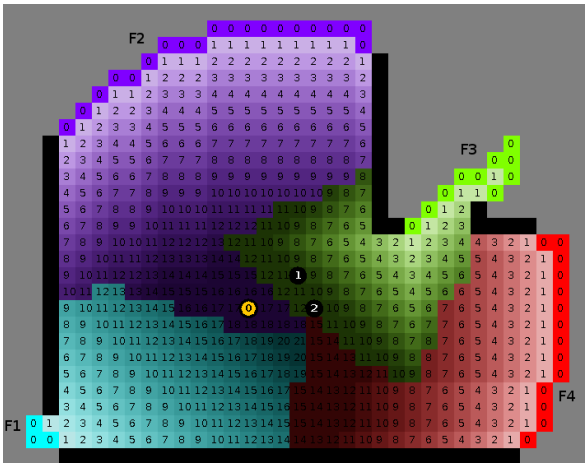


FIGURE 3.4 – Trois robots dans un environnement comportant quatre frontières. Le nombre inscrit dans chaque cellule est la distance à la frontière la plus proche.

Pour cela, dans un premier temps, une matrice coût, associée à la configuration illustrée en figure 3.4, est calculée :

		Frontière			
		$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$	$\mathcal{F}_4$
Robot	$\mathcal{R}_0$	20	20	15	19
	$\mathcal{R}_1$	25	15	10	16
	$\mathcal{R}_2$	24	19	11	15

Cette matrice nous permet de déterminer les assignations robot/frontière suivantes optimisant le critère de coût de l'exploration :

		Frontière			
		$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$	$\mathcal{F}_4$
Robot	$\mathcal{R}_0$	1	0	0	0
	$\mathcal{R}_1$	0	1	0	0
	$\mathcal{R}_2$	0	0	0	1

Cette dernière table, nous permet de déterminer les directions d'exploration de chaque robot.

3.4 Critères d'évaluation d'une stratégie d'exploration

Dans un premier temps nous considérons que les observations sont continues par rapport au déplacement. La fréquence d'acquisition du capteur est largement supérieure à la fréquence de planification. Ces hypothèses font que planifier la prochaine observation ou planifier la direction de navigation en comptant sur la prochaine planification est équivalent. En effet, la prochaine planification prendra en compte les nouvelles observations.

Pour évaluer ces assignations il faut connaître le coût de parcours des robots vers les frontières. Un chemin doit donc être calculé entre chaque robot et chaque frontière. Cette opération fournit une matrice de coût associant frontières et robots.

Avant de présenter des approches existantes, nous précisons les notations employées ainsi que les critères de qualité d'assignation.

3.4.1 Choix et affectation des frontières aux robots : coopération

Le problème d'exploration consiste à visiter successivement l'ensemble des frontières. Ce problème est hautement dynamique car les frontières sont découvertes, d'autres disparaissent quand elles sont explorées. L'exploration d'une frontière peut-être vu comme une tâche ayant une durée inconnue, nous ne pouvons donc pas associer une priorité ou une espérance de durée à ces tâches. Sachant que toutes les frontières doivent être explorées l'objectif à minimiser est le temps d'exploration maximum de chaque frontière existante à un moment donné c'est à dire à minimiser le temps où l'ensemble des frontières seront explorées. Notons $\mathcal{TF}_i$ le temps pour que la frontière i soit atteinte par un robot. Il faut alors minimiser $\mathcal{TF}_{max}$ le temps d'exploration maximum parmi toutes les frontières :

$$\mathcal{TF}_{max} = \max_{\forall i} \mathcal{TF}_i \tag{3.1}$$

Pour cela, il faut affecter un robot à chaque frontière. Lorsque le nombre de robots est inférieur au nombre de frontières, il faudra affecter plusieurs frontières à un robot et minimiser le temps de parcours entre les frontières qui lui sont affectées. Ce dernier problème est celui du voyageur de commerce et est NP-complet.

Le problème d'allocation d'une seule frontière à chaque robot est déjà un problème d'optimisation combinatoire NP-difficile. Plus précisément, nous cherchons une assignation parmi tous les arrangements sans répétition soit $\frac{m!}{(m-n)!}$ assignations possibles quand $n \leq m$ où $n = |\mathcal{R}|$ est le nombre d'agents et $m = |\mathcal{F}|$ le nombre de frontières. Cette quantité rend une recherche exhaustive des solutions rapidement intraitable quand le nombre de frontières et de robots croît. De plus, la fonction à optimiser (définie ci-après) n'est pas convexe et ne peut être décomposée ; une approximation est donc nécessaire.

Nous cherchons à affecter des cibles aux robots afin que l'exploration soit la plus rapide possible. Suivant la taille et la topologie de l'environnement le nombre optimal de robots pour

l'explorer est différent. Ici, comme l'environnement est inconnu ce nombre ne peut être déterminé a priori. Nous utilisons donc la totalité de la flottille. Affecter plusieurs frontières à un robot est complexe et n'est pas forcement utile car la plupart du temps la frontière est repoussée, autrement dit, une nouvelle frontière est créée un peu plus loin que celle qui vient d'être explorée et le robot qui l'a explorée est souvent le mieux placé pour explorer la nouvelle frontière. Ainsi, pour déployer les robots dans l'environnement nous affectons à chaque robot exactement une frontière à explorer. Cette contrainte s'exprime par :

$$\forall i \quad \sum_{j=1}^{m} \alpha_{ij} = 1 \tag{3.2}$$

L'opération d'affectation est effectuée à chaque changement d'état du système (exploration, disparition ou création d'une frontière). Les trois critères détaillés ci-après peuvent servir à évaluer la qualité d'une assignation.

1. Équilibre de la répartition des robots sur les frontières

Lors de l'exploration nous rencontrons trois cas de figures :

- *Nombre de robots égal au nombre de frontières.* Cas le plus simple, une frontière devra être affectée à chaque robot.
- *Nombre de frontières supérieur au nombre de robots.* Les robots devront chacun être affectés à des frontières distinctes pour répartir les robots vers des zones inexplorées différentes et ainsi éviter la redondance des informations récoltées.
- *Nombre de robots supérieur au nombre de frontières.* Après l'affectation d'un robot à chaque frontière, il faut déterminer une stratégie pour les robots non affectés. Au regard du critère de minimisation du temps total d'exploration, laisser des robots immobiles n'est pas une bonne stratégie, en effet, si un robot découvre une vaste zone inexplorée de l'environnement plusieurs robots seront nécessaires pour l'explorer plus rapidement. Comme il n'y a pas d'information, a priori, sur ce qui se trouve derrière les frontières, une répartition équilibrée du nombre de robots par frontière est donc souhaitable.

Ce critère d'affectation équilibré peut se résumer par :

$$\lfloor n/m \rfloor \ \leq \ \forall j \sum_{i=1}^{n} \alpha_{ij} \ \leq \ \lceil n/m \rceil \tag{3.3}$$

le nombre de robots par frontière est égal à une différence d'un robot près.

2. Minimum de la somme des coûts

Il s'agit de minimiser le coût de l'exploration en minimisant la somme des coûts (le coût de parcours de chaque robot) :

$$C(\mathcal{A}) = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_{ij} \ \ \mathcal{C}_{ij} \tag{3.4}$$

3. Minimum du maximum des coûts

Le respect du critère 3.4 ne garantit pas une solution unique. Par exemple, la figure 3.5 illustre une situation où deux solutions existent respectant les critères 3.3 et 3.4 mais celle affichée à droite s'effectuera en un temps d'exploration inférieur. En effet, le temps d'exploration de toutes les frontières affectées aux robots est déterminé par le temps d'exploration individuel maximum.

Un troisième critère d'évaluation est donc la minimisation du coût maximum :

$$C_{max}(\mathcal{A}) = \max_{\forall i} \sum_{j=1}^{m} \alpha_{ij}\, \mathcal{C}_{ij} \tag{3.5}$$

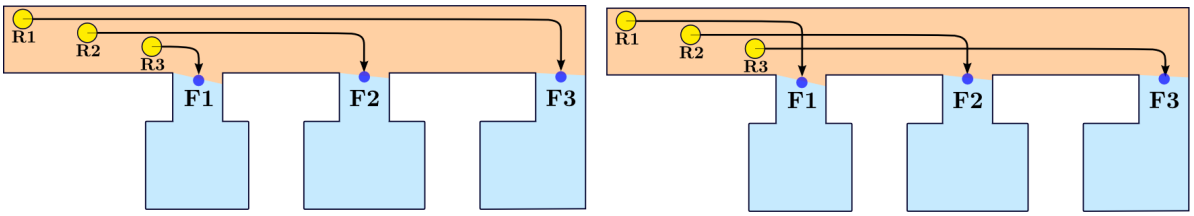


FIGURE 3.5 – Illustration du critère 3 : le somme des coûts de chaque assignation est identique mais le temps pour réaliser l'exploration de chaque frontière sera inférieur dans l'assignation à droite.

Il faut nuancer ce dernier critère car minimiser le coût maximum tout en respectant l'équilibre de la répartition peut augmenter la somme des coûts. Nous montrons, de manière intuitive, la complexité du problème sur un cas simple avec deux robots et deux frontières illustré figure 3.6. Dans cet exemple, minimiser le maximum des coûts augmente la somme des coûts car le robots R2 doit revenir en arrière pour retourner dans le couloir de explorer F2. Sur la figure à gauche, le robot R2 n'est pas très avancé dans le couloir de F1 et, il semble que le min du max soit une meilleure affectation. En revanche, sur la figure à droite, R2 est plus avancé dans le couloir et il semble trop coûteux qu'il revienne en arrière. Si l'exploration se terminait après la découverte des deux frontières, il serait toujours intéressant de minimiser le maximum des coûts, mais comme une zone inconnue existe derrière la frontière, il faut trouver un compromis entre minimiser la somme des coûts et minimiser le coût maximum.

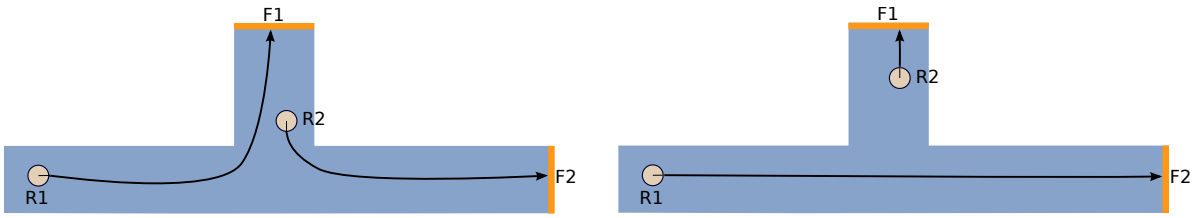


FIGURE 3.6 – Illustration comparant l'affectation utilisant le critère 3 (à gauche) et l'utilisation du critère 2 (à droite).

Les contraintes et critères posés ci-dessus sont discutables : la contrainte 1 (équation 3.2) - le choix d'affecter une seule frontière à chaque robot - est une simplification du problème qui permet de le traiter. Affecter l'ensemble des frontières constitue une alternative intéressante.

Nous proposons par la suite une stratégie d'exploration respectant les hypothèses et contraintes présentées dans ce chapitre.

3.5 Choix et contraintes liées au défi CAROTTE

Cette thèse s'inscrit dans le cadre du défi robotique Carotte. Dans un premier temps, nous présentons les différentes contraintes liées à ce défi. Dans un deuxième temps nous précisons les choix de notre approche.

3.5.1 Le défi robotique CAROTTE

Présentation

Le défi robotique français CAROTTE (CArtographie par RObot d'un TErritoire) s'est déroulé sur trois ans, de 2009 à 2012. Il a été réparti en trois phases se terminant chacune par une compétition. Les équipes participantes devaient réaliser un système robotisé autonome, capable de construire une carte en deux et trois dimensions d'un environnement intérieur inconnu. La carte devait intégrer des annotations sémantiques sur la topologie de l'environnement.

Les épreuves du concours reposaient sur des missions de cartographie en temps limité. Elles consistaient à naviguer dans une arène (une zone fermée constituée d'un ensemble de pièces) afin de générer une carte complète et de reconnaître automatiquement des objets présents dans l'arène, ainsi que la nature des sols et des parois la constituant. Le système robotisé devait accomplir sa mission de manière autonome i.e. sans aucune intervention ni contrôle extérieur (en particulier humain) et en temps limité.

Analyse des contraintes

Le défi CAROTTE fixe de nombreuses contraintes et spécifie le problème de l'exploration. La zone de départ des robots est donnée car elle sert à fixer le repère pour l'évaluation des cartes produites. Pour les robots, cette zone sert aussi à fixer un repère global entre eux. Cette hypothèse n'est pas irréaliste pour d'autres applications car en général les robots sont amenés sur une zone à explorer et démarrent donc leur exploration au même endroit (par exemple en descendant d'un camion). Toutefois les robots pourraient être parachutés sur la zone à explorer et devraient alors mettre en correspondance leur carte. L'environnement est de taille finie mais nous n'avons pas d'information a priori sur sa taille. C'est un environnement intérieur, c'est-à-dire structuré par des murs formant des pièces et des couloirs. Certaines approches existantes utilisent cette information pour en déduire, au cours de l'exploration, la topologie de l'environnement. Cela n'a pas été notre cas car cela induit une certaine perte de généricité. Dans ce contexte, nous utilisons les hypothèses suivantes : la prise de décision décentralisée et l'homogénéité de la flottille.

À ces contraintes, nous ajoutons quelques hypothèses sur les robots composant le système.

3.5.2 Choix sur la flottille

Homogénéité de la flottille

Nous considérons dans cette étude que la flottille est homogène (les robots la composant sont identiques). C'est une hypothèse simplificatrice qui amène de la robustesse : tout robot peut en remplacer un autre. Elle simplifie l'affectation des tâches aux robots, ainsi tous les robots sont également adaptés à effectuer une tâche. De plus, avec une flottille hétérogène, le calcul du coût d'affectation d'une tâche à un robot serait différent pour chaque robot pour prendre en compte ses propriétés cinématiques et dynamiques.

Calcul centralisé *versus* décentralisé

Les robots étant autonomes et la communication inter-robot étant peu fiable, nous avons privilégié un contrôle décentralisé. La première hypothèse afin de décentraliser la décision, est que chaque robot puisse connaître quelles zones ont déjà été explorées. Les robots coopèrent pour la construction de la carte globale en diffusant (*broadcast*) leur carte périodiquement afin de partager les observations individuelles. Cette méthode est détaillée au chapitre 5.

Un critère à prendre en compte dans le choix de la méthode d'assignation est le coût de communication. En effet, selon que le calcul sera centralisé ou décentralisé la méthode n'aura pas le même coût ni la même robustesse. Sous l'hypothèse de la disponibilité de la matrice de coût, l'assignation de frontière peut être effectuée soit par un agent central (robot dans l'environnement ou serveur central) soit en décentralisant la décision à travers des agents communicants. Cette dernière possibilité laisse les agents décider de manière autonome, résultant en un système décentralisé sans communication pour la coopération.

Décision synchrone *versus* asynchrone

L'affectation de frontières peut se faire de manière synchrone : les robots se dirigent vers leur frontière et une fois celle-ci atteinte, ils attendent que tous les robots aient atteints leurs frontières avant l'affectation des nouvelles frontières. Avec une affectation asynchrone, les robots sont affectés à une nouvelle frontière dès qu'ils atteignent celle qui leur à été précédemment affectée.

3.6 Synthèse des paramètres d'une exploration multi-robot

Lors d'une exploration multirobot, un certain nombre de paramètres sont à prendre en compte :

Pour l'*environnement* :
- la taille
- la topologie

Pour les *robots* :
- leur position initiale,
- leur nombre,
- la portée des capteurs,
- la forme des zones observées.

Pour la *stratégie d'exploration* :
- le choix des cibles à affecter,
- l'algorithme d'affectation des cibles,
- la fréquence d'observation et de ré-affectation,
- la planification et le suivi de trajectoire.

Certains de ces paramètres sont fixes et imposés : l'environnement et les contraintes matérielles ; d'autres sont modifiables comme la stratégie d'exploration. Nous nous intéressons particulièrement à cette dernière.

Dans les chapitres suivants, nous proposons une nouvelle stratégie d'exploration efficace au niveau des performances d'exploration et au niveau des coûts calculatoires. Cette stratégie est fondée sur une heuristique approximant le critère global à optimiser (voir équation 3.1).

Deuxième partie

Algorithmes d'exploration multirobot

Chapitre 4

Nouvel algorithme d'affectation des frontières : *MinPos*

Sommaire

Dans ce chapitre nous nous intéressons à la méthode d'affectation des frontières aux robots. Nous considérons, ici, que les robots observent et recalculent leur affectation périodiquement. De plus, nous nous plaçons dans un cadre où l'on considère que chaque robot communique aux autres sa position et sa carte. Les robots ont donc une connaissance complète de la carte partiellement construite et des coordonnées des autres robots. Les fréquences auxquelles les robots observent, communiquent et recalculent leurs affectations sont étudiées dans le chapitre 5.

Les stratégies de déploiement proposées dans ce chapitre visent à minimiser le temps d'exploration total (temps nécessaire à la construction de la totalité de la carte). Pour cela, les robots utilisent une stratégie qui doit leur permettre de se répartir au mieux dans l'environnement afin d'explorer simultanément des zones distinctes.

Tout au long de ce chapitre, nous examinons les algorithmes de la littérature en comparaison de ceux que nous proposons. Nous étudierons leur performance sur des configurations particulières. Ces configurations sont fréquentes dans les environnements d'intérieur. Il s'agit de situations clé dans la mesure où elles correspondent à des moments où la décision de répartition des robots est lourde de conséquence pour l'efficacité de l'exploration.

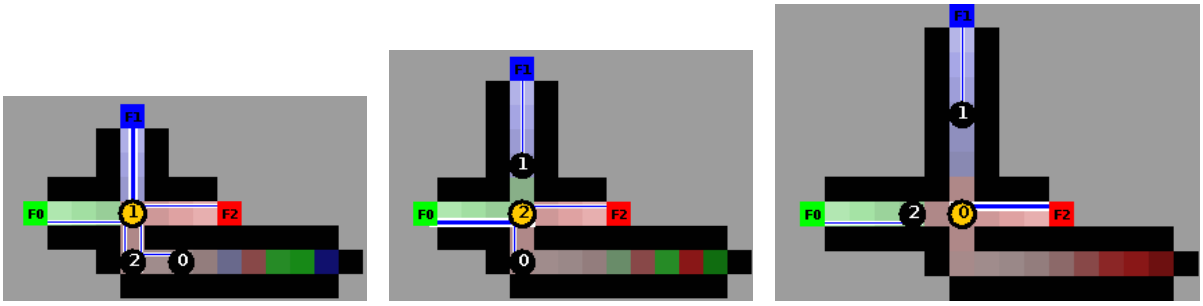


FIGURE 4.1 – Coordination implicite : trois robots sortent d'un couloir exploré et se répartissent de manière équilibrée dans les trois couloirs qui s'offrent à eux. Les frontières sont de couleur rouge, les robots de couleur bleu. Chaque robot se dirige vers la frontière la plus proche de lui.

4.1 Limite des approches existantes

Dans cette section, nous analysons les méthodes d'affectation de frontière existantes vues dans le chapitre 2 qui peuvent être appliquées à notre problématique.

4.1.1 Frontière la plus proche

L'approche la plus simple consiste à affecter chaque robot à sa frontière la plus proche (Yamauchi, 1998), nommée ci-après *MinDist*. Cette approche présente une coordination implicite assez performante. Toutefois, *MinDist* présente une limitation importante pour son efficacité : dans certaines situations les robots se dirigent vers la même frontière alors qu'il existe d'autres frontières non affectées.

L'algorithme 1 définit formellement l'algorithme *MinDist*.

Algorithme 1: *MinDist*

Entrées : $\mathcal{C}_i$ un vecteur de coût de chemin à chaque frontière
Sorties : $\mathcal{A}_i$ l'assignation du robot i
début
  $\alpha_{ij} = 1$ tel que $j = \underset{j|\mathcal{F}j\in\mathcal{F}}{\operatorname{argmin}}\ \mathcal{C}_{ij}$ (choix aléatoire si plusieurs frontières ont des coûts égaux)
fin

La complexité computationnelle de cet algorithme est $O(m)$.

Avec *MinDist* plusieurs robots peuvent donc être affectés à la même frontière F_i. Cette affectation est efficace si le chemin des robots les rapprochent d'autres frontières. Les robots peuvent se répartir ensuite sur ces frontières. Le changement d'affectation s'effectue alors lorsque les chemins menant aux différentes frontières se séparent. Le chemin initialement parcouru vers la frontière F_i n'est donc pas inutile puisqu'il a mené vers les autres frontières. Il est d'usage, alors, de parler de coordination implicite car les robots coopèrent uniquement à travers la construction d'une carte commune. Cette situation est illustrée figure 4.1.

Dans le cas de cet exemple, le déroulement de l'exploration est similaire à ce qu'elle aurait donné en utilisant des algorithmes d'affectation plus complexes qui auraient affecté les robots directement aux frontières qu'ils auront finalement explorées.

En revanche, quand les frontières non affectées sont réparties dans des directions différentes, cette approche est mise en défaut. Il est plus efficace d'affecter directement les robots dans ces directions différentes que de les laisser explorer une même frontière. La figure 4.2 illustre une telle situation où les robots 0 et 5 se dirigent vers la même frontière alors qu'il existe des frontières non affectées. Après un changement d'affectation, le chemin parcouru devra être rebroussé pour rejoindre la frontière situé dans une autre direction.

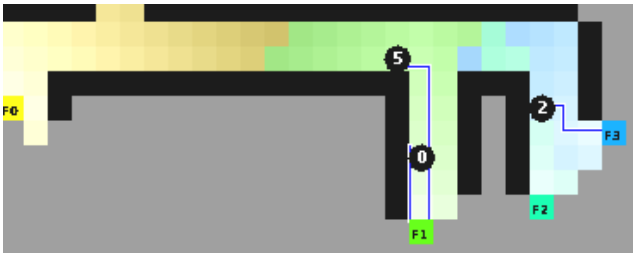


FIGURE 4.2 – Illustration de l'allocation de frontière avec l'algorithme *MinDist*.

MinDist ne fournit pas non plus de bonnes solutions quand le nombre de robots est supérieur au nombre de frontières. Une répartition efficace devrait séparer les robots en groupes de tailles égales vers des directions différentes. Or comme cela est illustré figure 4.3, cet équilibre n'est pas assuré avec l'allocation *MinDist*. Un groupe de robots est proche d'une frontière, alors qu'une autre frontière se trouve plus loin dans une direction opposée. Avec *MinDist*, tous les robots sont affectés à la même frontière tandis que l'affectation idéale devrait répartir les robots en deux groupes de tailles égales.

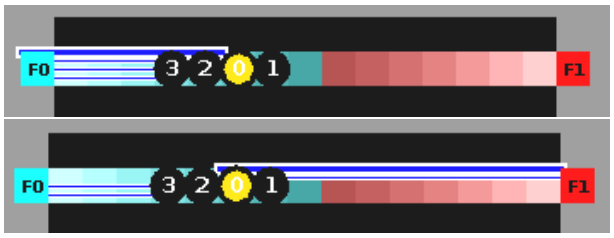


FIGURE 4.3 – En haut : *MinDist* : tous les robots sont assignés à la même frontière ;
en bas : avec l'affectation souhaitée : les robots sont également répartis entre les deux frontières

4.1.2 Affectation gloutonne

Nous analysons, ici, l'approche la plus utilisée de la littérature. L'affectation gloutonne comme l'algorithme hongrois (voir section 2.5.4) minimise la somme des coûts occasionnés par la navigation vers chaque frontière tout en garantissant une répartition équilibrée des robots (voir algorithme 2).

Algorithme 2: Glouton (version décentralisée)

Entrées : $\mathcal{C}$ Matrice de coût
Sorties : α_{aj} assignation du robot $\mathcal{R}_a$ à la frontière $\mathcal{F}_j$

$\mathcal{F}_{init} = \mathcal{F}$

Tant que $\mathcal{R}_a$ *n'est pas assigné* **faire**
  Trouver i, j $= \underset{i,j|\mathcal{R}_i \in \mathcal{R},\ \mathcal{F}j \in \mathcal{F}}{\text{argmin}} \mathcal{C}_{ij}$
  $\alpha_{ij} = 1$
  $\mathcal{R} = \mathcal{R} \setminus \mathcal{R}_i$
  $\mathcal{F} = \mathcal{F} \setminus \mathcal{F}_j$
  si $\mathcal{F} = \varnothing$ **alors** $\mathcal{F} = \mathcal{F}_{init}$
fin

La complexité de l'algorithme 2 (glouton) est $O(n^2 m)$.

Ces critères ne sont cependant pas suffisants pour garantir une exploration efficace. Par exemple, dans la situation illustrée par la figure 4.4, l'affectation gloutonne des frontières attribue les deux frontières les plus proches aux robots : $\mathcal{F}_2$ à $\mathcal{R}_2$ et $\mathcal{F}_3$ à $\mathcal{R}_1$. Les deux robots se dirigent alors dans la même direction plutôt que de se répartir. Une solution plus efficace serait de diriger les robots vers des directions différentes. Ainsi, l'affectation avec laquelle le robot $\mathcal{R}_1$ se dirige vers la frontière $\mathcal{F}_1$, qui est plus éloignée que $\mathcal{F}_2$ et $\mathcal{F}_3$ mais qui est dans une direction différente, sera plus efficace. Dans ce cas, si toutes les frontières sont sans issue, l'exploration complète nécessitera moins de temps car $\mathcal{R}_2$ explorera $\mathcal{F}_2$ puis $\mathcal{F}_3$, alors que pendant ce temps $\mathcal{R}_1$ pourra explorer $\mathcal{F}_1$.

Un exemple similaire, illustré en figure 4.5, montre trois robots avec quatre frontières accessibles. L'algorithme Glouton affectera le robot 5 à la frontière $\mathcal{F}_3$ car elle est plus proche que $\mathcal{F}_0$ et n'a pas encore de robot assigné. Là encore, l'affectation du robot 5 dans une direction différente des frontières $\mathcal{F}_2$ et $\mathcal{F}_3$ est préférable car le robot 2 pourra explorer la frontière 2 après avoir exploré la frontière $\mathcal{F}_2$.

FIGURE 4.4 – Affectation souhaitée (à gauche) et affectation de l'algorithme Glouton (à droite).

Glouton sur l'utilité

Il existe des variantes des affectations gloutonnes pouvant produire des stratégies plus efficaces que l'algorithme Glouton (algorithme 2). Elles utilisent un regroupement des frontières pour affecter plusieurs frontières à un même robot améliorant ainsi le critère de minimisation du maximum du coût d'exploration de chaque frontière (voir équation 3.1).

La notion d'utilité n'est pas, à proprement parler, un regroupement de frontières mais son effet est similaire. Dans ce cadre les frontières à explorer sont évaluées selon l'utilité de les affecter

à un robot connaissant l'affectation des autres robots (Burgard et al., 2005). Ces algorithmes maximisent l'inter-distance entre les frontières affectées tout en prenant en compte la visibilité d'une frontière vue d'une autre. Ainsi, lorsqu'un robot est affecté à une frontière, il est moins utile d'affecter un autre robot à une frontière proche de celle-ci. Cette approche effectue ainsi un couplage entre l'identification des cibles et leurs affectations. Dans l'exemple illustré en figure 4.5 l'affectation du robot 2 à la frontière $\mathcal{F}_2$ réduira l'utilité d'exploration de $\mathcal{F}_3$ car cette dernière frontière est proche et observable depuis $\mathcal{F}_2$. Le robot 5 sera donc affecté à une frontière dont l'utilité est plus grande : $\mathcal{F}_0$. En revanche, sur la situation illustrée en figure 4.4 l'utilité de la frontière $\mathcal{F}_3$ ne sera pas diminuée car elle n'est pas visible depuis $\mathcal{F}_2$. L'affectation utilisant l'utilité sera donc identique à celle donnée par l'algorithme glouton n'utilisant pas l'utilité.

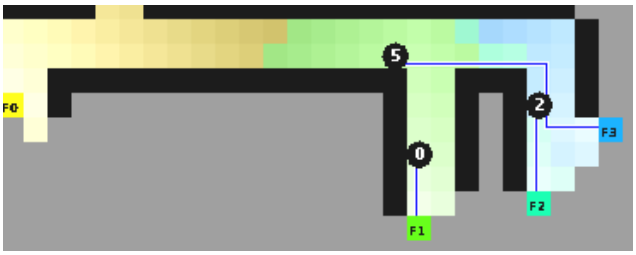


FIGURE 4.5 – Illustration de l'allocation de frontière avec l'algorithme glouton.

Gain d'information

D'autres algorithmes utilisant la notion d'utilité cherchent à maximiser le gain d'information en maximisant une approximation du gain d'information associé à une observation (par exemple en observant les frontières les plus larges) (Burgard et al., 2002). Cette stratégie ne minimise pas le temps d'exploration car elle utilise une approximation qui peut se révéler fausse et contre productive. La figure 4.6 illustre un tel exemple. Dans cet exemple, si l'affectation maximise le gain d'information deux robots seront affectés à la frontière $\mathcal{F}1$ et un seul robot aux frontières $\mathcal{F}2$ et $\mathcal{F}3$. En effet, la frontière $\mathcal{F}1$ est large et le gain d'information associé à son observation est donc grand, tandis que $\mathcal{F}2$ et $\mathcal{F}3$ sont étroites. Une affectation équilibrée des robots sur les frontières répartira les trois robots sur les trois frontières.

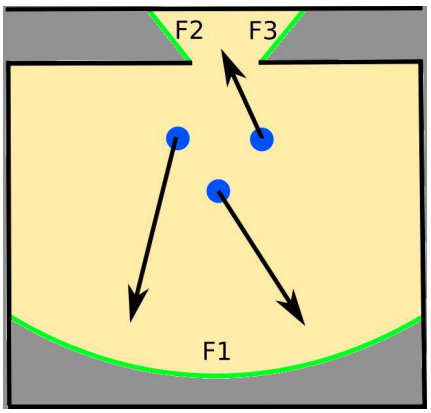


FIGURE 4.6 – Exemple de situation où maximiser le gain d'information est une stratégie moins performante que l'affectation de frontières.

4.1.3 Affectation de plusieurs frontières

Le regroupement des frontières peut prendre en compte le nombre de robots disponibles. Par exemple, (Faigl et al., 2012) identifient un nombre de cibles égal au nombre de robots dans l'environnement. L'idée est de regrouper les cellules frontières en autant de groupe qu'il y a de robot. L'opération de fusion de plusieurs frontières s'effectue avec une variante de l'algorithme K-means. Un groupe de frontière est alors affecté à chaque robot. L'initialisation de l'algorithme K-means est faite en positionnant le centre de chaque classe sur un robot.

Cette approche présente l'avantage de mieux répartir les robots dans l'environnement car elle crée des groupes de frontières en fonction de la distance des robots. Par exemple, sur la figure 4.4 les frontières $\mathcal{F}_2$ et $\mathcal{F}_3$ appartiendront au même groupe et l'affectation souhaitée sera ainsi obtenue. Toutefois cette approche comporte aussi plusieurs désavantages. Le nombre de frontières constituant un groupe peut être très grand ou nul ; il est surtout non homogène. Le calcul de la longueur du chemin permettant de passer par toutes les frontières d'un groupe est un problème NP-complet car il est semblable au problème du voyageur de commerce. En outre, lors de l'observation d'une frontière, celle-ci est en général repoussée et devient la prochaine frontière à explorer car elle est la plus proche du robot, le chemin calculé entre les différentes frontières est donc directement obsolète.

Discussion

L'efficacité des stratégies d'exploration est donc liée à la méthode de regroupement des frontières utilisée et à la méthode d'affectation de ces groupes. La plupart des recherches actuelles améliorent les méthodes classiques en proposant de nouvelles méthodes de regroupement des frontières. Ces approches permettent une meilleure efficacité de l'exploration mais posent problème pour leur décentralisation et leur "désynchronisation". En effet, les frontières sont modifiées quasiment en permanence durant l'exploration et ceci modifie les groupes de frontières et donc l'affectation.

Notons que ces méthodes sont efficaces lorsque le calcul des affectations s'effectue de façon centralisée. La distribution du calcul sur chaque robot est difficile. Si chaque robot effectue le même calcul du regroupement de frontières l'affectation peut être différente d'un robot à l'autre. En effet, si l'exécution n'est pas synchrone les données sont différentes d'un robot à l'autre.

Il existe peu de méthodes d'affectation de frontières. En résumé les méthodes d'affectation existantes sont les suivantes : *MinDist*, Glouton, méthode hongroise et affectation optimale (génération et évaluation de toutes les permutations). *MinDist* est simple, peut facilement être décentralisée mais la coopération entre les robots est limitée. Glouton est la méthode la plus utilisée, même si elle est un peu plus complexe, car elle offre une meilleure coopération et elle peut facilement être décentralisée (système d'enchère). Cependant son efficacité est conditionnée par la méthode de regroupement des frontières. Les affectations avec la méthode hongroise sont très similaires à l'affectation gloutonne mais encore un peu plus complexe. Finalement, l'affectation minimisant la somme des coûts ainsi que le coût maximum d'exploration des frontières (Burgard et al. (2005) algorithme 2) est trop complexe car elle a une complexité exponentielle avec le nombre de robot (évaluation de toutes les permutations des robots sur les frontières).

L'approche proposée est une nouvelle méthode d'affectation qui effectue implicitement un regroupement des frontières.

4.2 Principe général de *MinPos*

Comme nous l'avons vu, la distribution des robots vers des frontières situées dans des directions différentes est une bonne stratégie d'affectation. Nous avons montré précédemment que les approches existantes n'assurent pas toujours cette distribution spatiale, ce qui peut mener à des performances d'exploration médiocres. Notre approche introduit un autre critère plus robuste que la métrique de la distance aux frontières. Il s'agit de l'ordre des robots, selon leur proximité, à chaque frontière. L'idée est alors d'affecter chaque robot à la frontière pour laquelle il est en meilleure position par rapport aux autres robots (ce n'est pas nécessairement la frontière dont il est le plus proche). Cette stratégie favorise implicitement la distribution des robots vers des frontières différentes et non-voisines.

Ainsi, pour obtenir une répartition équilibrée des robots dans les différentes directions menant à des frontières, nous proposons d'affecter les robots en fonction de leurs positions relatives aux frontières. La position est définie comme le rang d'un robot parmi un ensemble de robots ordonnés selon leur distance à une frontière. Pour prendre en compte les positions relatives des robots vers les frontières dans l'environnement, nous calculons pour chaque couple robot-frontière le nombre de robots plus proches de la frontière considérée. Ce nombre est appelé *position du robot sur la frontière*. Chaque robot est alors affecté à la frontière pour laquelle il a la position minimum. Notre proposition est donc une alternative aux approches uniquement fondées sur les distances aux frontières par l'utilisation de la position vers les frontières.

Pour évaluer ces positions nous pouvons utiliser la matrice de coût. Pour la calculer, nous proposons une approche fondée sur la construction de champs de potentiels depuis les frontières (algorithme de front de vague (Barraquand et al., 1991)). Il fournit la distance du chemin à parcourir pour atteindre la frontière en tout point de l'environnement et donc pour chaque robot. Par ailleurs, la descente du gradient de ce champ de potentiel permet de naviguer de façon optimale jusqu'à la frontière. Ce calcul de la matrice de coût par propagation de front de vague sera présenté au chapitre 6.

Le principe général de notre approche détaillée ci-après, sous l'hypothèse que la carte et les positions (coordonnées) des robots sont partagées, peut être résumé par les étapes suivantes :

1. Identification des frontières entre régions explorées/inexplorées (chapitre 7).
2. Calcul des champs de potentiels depuis les frontières (chapitre 6).
3. Attribution des frontières aux robots (ce chapitre).
4. Navigation et retour au point 1.

Chaque robot se dirige alors vers la frontière qui lui est attribuée. La première étape sera détaillée dans le chapitre 7. Dans ce chapitre nous supposons que les cellules frontières sont identifiées et regroupées quand elles sont contiguës.

Ces étapes peuvent être effectuées de façon décentralisée (sur chaque robot) ou sur un serveur central. Dans ce qui suit, nous considérons une exécution décentralisée, c'est-à-dire où chaque robot exécute l'algorithme d'affectation pour lui-même. Pour que les décisions prises par les robots soient cohérentes entre elles, il faut qu'ils partagent les mêmes informations sur l'environnement (positions et carte). Ainsi, les robots mettent à jour leur carte avec les informations communiquées par les autres robots avant de déterminer leur prochaine cible (nous reviendrons sur ces hypothèses aux chapitres 5 et 8).

4.3 Algorithme *MinPos*

Cette section présente la première version, la plus simple, de l'algorithme *MinPos*. Cet algorithme utilise les distances entre robots et frontières.

L'algorithme que nous proposons cherche à distribuer les robots vers les différentes frontières en minimisant le nombre de robots empruntant les mêmes directions (mêmes espaces traversés par des robots). Pour cela nous posons le critère de position d'un robot vers une frontière comme étant le nombre de robots plus proches de la frontière que lui. Formellement, $\mathcal{P}_{ij}$ la position du robot i vers la frontière j est définie par :

$$\mathcal{P}_{ij} = \text{Card } (\tilde{\mathcal{R}}) \; avec \; \tilde{\mathcal{R}} = \{\forall \mathcal{R}_k \in \mathcal{R},\ \mathcal{C}_{kj} < \mathcal{C}_{ij}\}$$

Notre approche consiste à remplacer le critère d'affectation de distance utilisé par *MinDist* par celui de la position. Ainsi, un robot est assigné à la frontière pour laquelle il est en meilleure position (la plus petite). En cas d'égalité entre plusieurs frontières, le robot choisit la plus proche. Le processus d'affectation est présenté par l'algorithme 3.

En raisonnant sur les positions plutôt que sur les distances, deux robots voisins se répartiront sur deux frontières dans des directions distinctes où ils seront premiers quelles que soient les distances. La distance du robot à la frontière à laquelle il est affecté peut être grande. Ce qui s'oppose aux autres approches ; la probabilité d'une meilleure répartition est donc plus élevée. Nous verrons dans la section qui suit que cette répartition dans des directions distinctes favorise une séparation spatiale des robots.

Algorithme 3: *MinPos*

Entrées : $\mathcal{F}$ l'ensemble de frontières, $\mathcal{R}$ l'ensemble de robots, $\mathcal{C}$ la matrice de coût
Sorties : α_{ij} assignation du robot $\mathcal{R}_i$

Pour chaque $\mathcal{F}_j \in \mathcal{F}$ **faire**

$$\mathcal{P}_{ij} = \text{Card } (\tilde{\mathcal{R}}) \; avec \; \tilde{\mathcal{R}} = \{\forall \mathcal{R}_k \in \mathcal{R} \mid \mathcal{C}_{kj} < \mathcal{C}_{ij}\}$$

fin

$\alpha_{ij} = 1$ tel que $j = \underset{j|\mathcal{F}j\in\mathcal{F}}{\text{argmin}}\ \mathcal{P}_{ij}$

En cas d'égalité choisir la frontière avec le coût minimum parmi les minima de $\mathcal{P}_{ij}$

L'algorithme parcourt la liste de frontières $\mathcal{F}$ pour déterminer la position du robot $\mathcal{R}_i$ vis à vis de chacune des frontières. La position est déterminée en parcourant la liste des robots pour compter le nombre de robots ayant un coût inférieur vers la frontière que $\mathcal{R}_i$. La complexité de l'algorithme 3 *MinPos* est en $O(nm)$ n est le nombre de robots et m est le nombre de frontières.

Dans *MinPos*, la notion de rang se distingue des autres approches qui considèrent plutôt des distances. *MinPos* effectue ainsi, du point de vue d'un robot, une partition des frontières en deux catégories. Celles pour lesquelles il existe un robot plus proche que ne l'est le robot considéré et les autres. Pour ce robot, les frontières pour lesquelles il y a un robot plus proche seront considérées comme un tout (regroupement de frontière) qu'il n'a pas à considérer. Il y a ainsi une dispersion plus forte que ne l'aurait permis des algorithmes comme *MinDist* et Glouton. Nous montrons ci-après comment l'algorithme *MinPos* favorise la dispersion des robots sur des situations courantes d'une exploration multirobot.

4.4 Évaluation qualitative

Dans cette section, nous analysons et comparons les affectations des différents algorithmes présentés dans ce chapitre sur des exemples d'intersection, pour lesquelles les choix d'affectation sont déterminants pour une bonne efficacité de l'exploration.

4.4.1 Illustration

La figure 4.7 illustre l'affectation de *MinPos* sur la même situation que celle illustrée en figure 4.2 (*MinDist*) et 4.5 (Glouton). Ici, le robot $R5$ est affecté à la frontière $F0$ vers laquelle il est en première position tandis qu'il est en seconde position vers les frontières $F1$, $F2$ et $F3$. L'affectation de *MinPos* assure ici une meilleure distribution des robots dans l'environnement pour une exploration plus efficace.

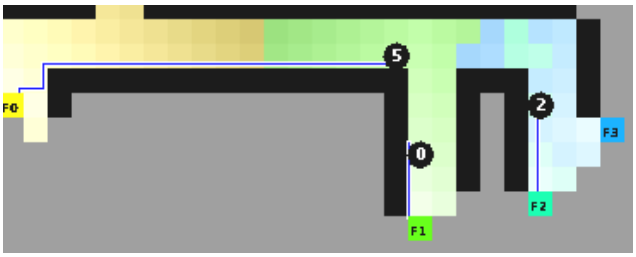


FIGURE 4.7 – Résultat de l'assignation avec l'algorithme 3 *MinPos*.

4.4.2 Affectation sur des cas difficiles

Nous reprenons, ici, l'exemple des deux robots et des trois frontières dont l'une est située à l'opposé des deux autres (voir figure 4.8). Nous évaluons le gain obtenu par l'affectation de *MinPos* sur cet exemple.

Supposons que la distance entre les robots $R1$ et $R2$ soit négligeable par rapport aux distances aux frontières. Notons $d1$, $d2$ et $d3$ les distances des robots aux frontières $F1$, $F2$ et $F3$ respectivement.

Nous posons $d3 > d2$ alors :

Si $d1 >> d2, d3$:

- avec Glouton (figure 4.8 droite), la plus grande distance parcourue parmi les deux robots est $max(2d2 + d1, d3) = 2d2 + d1$. Le robot $R2$ qui va explorer la frontière $F2$ (d2) devra ensuite revenir en arrière (+d2) pour ensuite se rendre à $F1$ (+d1).
- avec *MinPos* (figure 4.8 gauche), la plus grande distance parcourue parmi les deux robots est $max(2d2 + d3, d1) = d1$. Le robot $R2$ qui va explorer la frontière $F2$ (d2) devra ensuite revenir en arrière (+d2) pour ensuite se rendre à $F3$ (+d3).

Le gain opéré par *MinPos* est donc de $2d2$.

Si $d1 > d2, d3$ (si $2d2 + d3 > d1$), les affectations seront les mêmes que précédemment. Avec *MinPos* la plus grande distance parcourue parmi les deux robots est alors $max(2d2 + d3, d1) = 2d2 + d3$. Le gain opéré par *MinPos* est alors égal à la différence des distances entre d1 et d3 $(d1 - d3)$.

Si $d1 < d2, d3$, les affectations des deux algorithmes seront identiques.

FIGURE 4.8 – Affectation de *MinPos* (à gauche) et affectation de l'algorithme Glouton (à droite).

Le temps maximum pour que chaque frontière soit explorée est égal au temps d'exploration de la totalité de ce petit environnement (en considérant que les frontières mènent sur des impasses). Il est minimum avec l'affectation effectuée par *MinPos*. Le gain obtenu par l'affectation de *MinPos* en comparaison de l'affectation gloutonne est variable suivant les distances des robots aux frontières mais il est, dans tous les cas, nul ou positif. Nous pouvons donc conclure que, sur cet exemple, l'affectation de *MinPos* est meilleure que l'affectation gloutonne des frontières aux robots.

Un environnement, nommé ci-après *écurie*, a été construit pour illustrer l'avantage de *MinPos* par rapport à l'affectation gloutonne des frontières. Dans cet environnement, illustré en figure 4.9, plusieurs pièces sont disposés sur une ligne oblique au coté de l'environnement. Trois robots sont placés sur le côté nord en face des trois premières pièces. L'affectation gloutonne des frontières, comme l'assignation de *MinDist*, alloue les frontières des trois premières pièces aux trois robots dans leur ordre respectif. Avec *MinPos*, le robot le plus à gauche est en première position pour les deux premières portes. Le robot du milieu est en deuxième position pour les deux premières portes et en première position pour les deux portes suivantes et sera donc affecté à la troisième pièce. L'assignation de *MinPos* est ici optimale ; les robots se répartissent de façon équilibrée dans l'environnement.

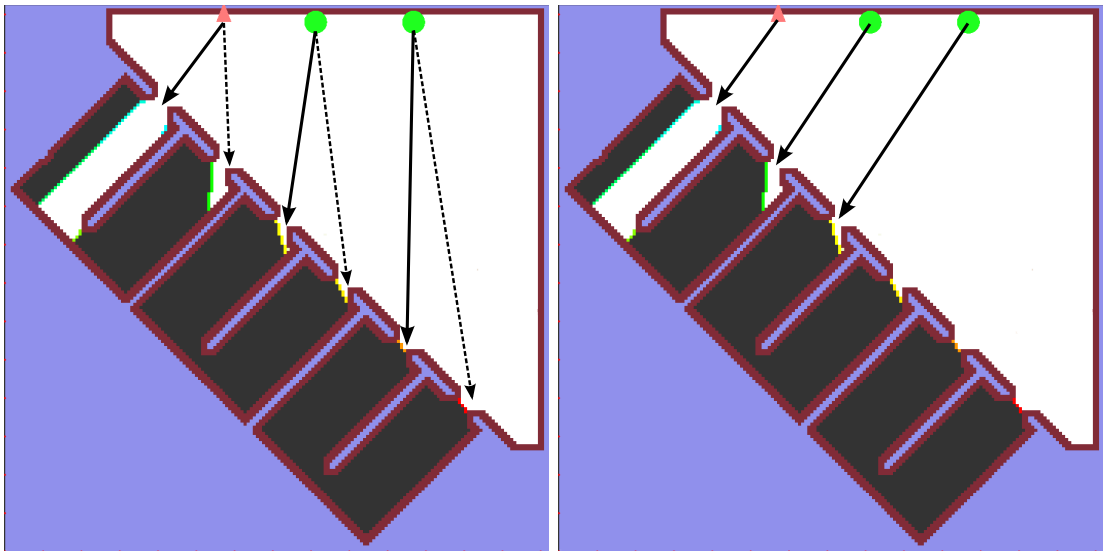


FIGURE 4.9 – Affectation de *MinPos* (à gauche) et l'affectation gloutonne (à droite) sur l'environnement écurie. Les flèches (en ligne pleine) illustrent l'affectation, celles en pointillé montrent les deuxièmes frontières pour lesquelles les robots sont en première position.

4.4.3 Stabilité de l'affectation

MinPos possède une caractéristique intéressante qui est la stabilité de l'affectation ; il est assez peu sensible à l'apparition ou la disparition d'une frontière dans une direction donnée. En effet, *MinPos* ne prend pas en considération le nombre de frontières dans les différentes directions pour calculer l'affectation du robot ; il se base uniquement sur le nombre de robots plus proches.

En revanche, l'apparition et la disparition des frontières peut rendre l'affectation gloutonne très instable. La figure 4.10 illustre une telle situation. Dans l'état initial, quatre robots perçoivent quatre frontières dont trois dans une même direction. L'affectation gloutonne dirigera chaque robot vers une frontière, c'est-à-dire trois robots dans la même direction (voir les flèches sur les robots). Dans la deuxième étape (figure centrale), le robot le plus à gauche découvre des impasses, ce qui élimine deux des trois frontières, tandis que le robot le plus à droite a découvert deux nouvelles frontières. Les deux robots centraux changent donc de direction pour se diriger vers ces nouvelles frontières. Ce changement de direction s'effectue de nouveau lorsque le nombre de frontières dans chaque direction change. Les deux robots centraux restent immobiles ou oscillent entre les deux directions. Cette situation est particulière mais montre un exemple de ce qui peut arriver avec un nombre plus important de robots.

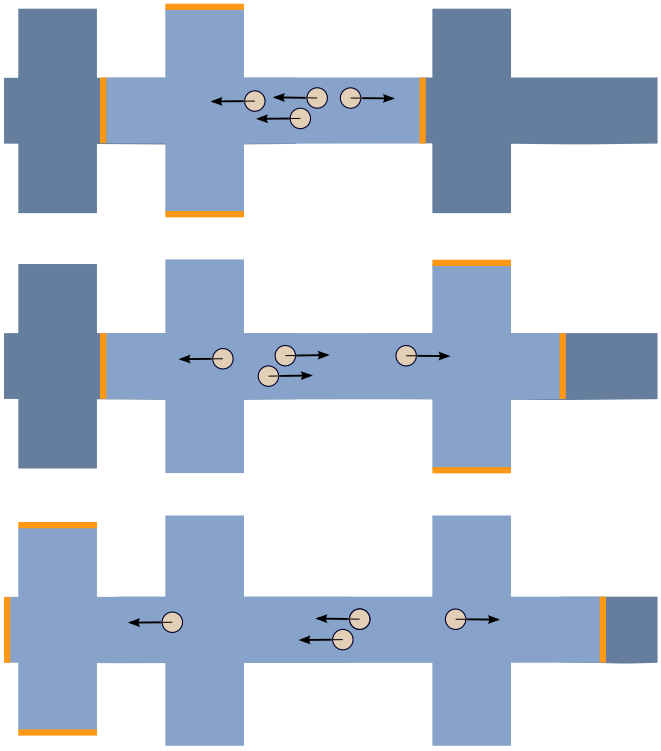


FIGURE 4.10 – Exemple de situation où l'affectation gloutonne est instable. Les deux robots au centre oscillent entre la droite et la gauche selon la disponibilité des frontières

Dans cette situation, l'affectation de *MinPos* sépare les quatre robots en deux groupes l'un se dirigeant vers la gauche, l'autre vers la droite. Les deux robots centraux sont en deuxième et en troisième position dans chaque direction et choisissent la frontière vers laquelle ils sont en deuxième position. Le groupe de robots sera donc séparé en deux groupes égaux vers les deux directions principales du couloir.

4.4.4 Comportement dynamique coopératif

MinPos induit un comportement dynamique très intéressant ; il tend à séparer les robots groupés. Par exemple, dans la situation illustrée en figure 4.11, deux robots se suivent pour atteindre une même frontière. Ils se séparent dès que le premier s'éloigne de la frontière orpheline, celle-ci captant le second robot. En effet, il passe alors en première position vers cette frontière. En comparaison, avec l'algorithme *MinDist*, les deux robots auraient exploré la zone derrière la frontière $\mathcal{F}_1$ tandis qu'avec l'affectation gloutonne, les robots auraient initialement été affectés aux différentes frontières.

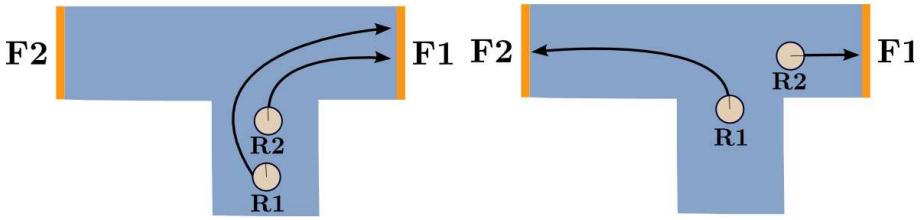


FIGURE 4.11 – Deux robots se séparant sur une intersection en T avec l'algorithme *MinPos*.

En revanche, l'algorithme Glouton est meilleur que *MinPos* lorsque, du point de vue d'un robot, il existe une direction avec plusieurs frontières et plusieurs robots plus proches. *MinPos* prend en compte le nombre de robots dans une direction mais pas le nombres de frontières. Ainsi, si le nombre de frontières est égal à celui des robots, il peut laisser des frontières sans allocation, alors qu'une répartition équilibrée était possible. Ceci est dû au fait que l'algorithme glouton garantit le respect du critère 1 (voir équation 3.3), une répartition équilibrée des robots sur les frontières. La figure 4.12 illustre cette situation où le robot $\mathcal{R}_3$ "voit" la frontière $\mathcal{F}_1$ avec deux robots plus proches. Il est ainsi en troisième position pour les frontières $\mathcal{F}_1$, $\mathcal{F}_2$ et $\mathcal{F}_3$ et va donc se diriger vers la frontière $\mathcal{F}_4$ pour laquelle il est en deuxième position. Dans cette situation, l'allocation gloutonne des frontières est plus efficace, toutefois les résultats montrent que l'impact sur les performances est faible.

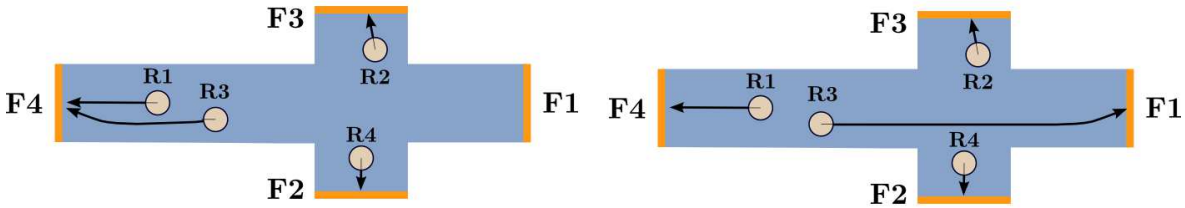


FIGURE 4.12 – Comparaison des assignations avec les algorithmes *MinPos* (à gauche) et Glouton (à droite).

Pour résoudre ce problème dans l'affectation, une solution est une affectation gloutonne des frontières sur les robots. La section suivante présente une nouvelle version d'un algorithme glouton pour l'affectation des cibles utilisant notre critère de position.

4.5 Conjugaison des approches *MinPos* et gloutonne (*MPG*)

Comme nous l'avons vu sur l'exemple 4.12, l'affectation gloutonne garantit l'équilibre de la répartition des robots sur les frontières. En contrepartie, il est moins efficace en répartition spatiale que *MinPos* (voir les exemples 4.4 et 4.9). La conjugaison des deux approches est possible en affectant en priorité les robots aux frontières non encore assignées et en choisissant la frontière pour laquelle le robot est en meilleure position parmi les frontières non assignées.

L'algorithme 4 définit cette synthèse que nous appelons *MPG*. C'est une combinaison de l'algorithme Glouton et de *MinPos*. À partir de la matrice de coût nous calculons une matrice de position en comptant, pour chaque robot, le nombre de robot ayant un coût inférieur au sien. L'algorithme effectue une affectation gloutonne des frontières pour lesquelles les robots sont en première position. Quand une frontière est affectée à un robot, le robot et la frontière sont retirées de leur liste respective. Lorsqu'il n'y a plus de robot en première position vers une frontière non encore affectée, la même opération est effectuée avec les robots en deuxième position, puis en troisième, etc. Cette opération est répétée, jusqu'à ce que le robot effectuant le calcul détermine la frontière à laquelle il est affectée.

L'algorithme procède donc par étapes en faisant varier une valeur de position, ci-après nommée *POS*. Durant chaque étape, les couples robot-frontière, dont les positions sont égales à *POS*, sont assignés de manière gloutonne (en fonction du coût associé au couple).

Algorithme 4: *MPG*

Entrées : $\mathcal{F}$ l'ensemble de frontières, $\mathcal{R}$ l'ensemble de robots, $\mathcal{C}$ la matrice de coût
Sorties : α_{aj} assignation du robot $\mathcal{R}_a$ à la frontière $\mathcal{F}_j$

$POS = 0$, $\mathcal{F}_{init} = \mathcal{F}$

$\forall i \; \forall j$ *Calculer* $\mathcal{P}_{ij} = \text{Card}\,(\tilde{\mathcal{R}})$ *avec* $\tilde{\mathcal{R}} = \{\forall \mathcal{R}_k \in \mathcal{R} \mid \mathcal{C}_{kj} < \mathcal{C}_{ij}\}$

Tant que $\mathcal{R}_a$ *n'est pas assigné* **faire**
  Trouver i, j $= \underset{i,j|\mathcal{R}_i \in \mathcal{R},\ \mathcal{F}j \in \mathcal{F},\ POS = \mathcal{P}_{ij}}{\text{argmin}} \mathcal{C}_{ij}$
  si $(i, j) = \varnothing$ **alors**
    $POS = POS + 1$
  fin
  sinon
    $\alpha_{ij} = 1$
    $\mathcal{R} = \mathcal{R} \setminus \mathcal{R}_i$
    $\mathcal{F} = \mathcal{F} \setminus \mathcal{F}_j$
  fin
  si $\mathcal{F} = \varnothing$ **alors** $\mathcal{F} = \mathcal{F}_{init}$
fin

La complexité de l'algorithme 4 *MPG* est, comme l'algorithme Glouton, en $O(n^2 m)$.

Dans les cas où le nombre de frontières est inférieur ou égal au nombre de robots, l'affectation de *MPG* sera identique à l'affectation gloutonne. Quand le nombre de frontières est supérieur au nombre de robots, l'affectation sera une combinaison de l'affectation de *MinPos* et de Glouton.

La figure 4.13 illustre un exemple montrant la supériorité de l'algorithme *MPG* par rapport aux algorithmes *MinPos* et Glouton. Trois robots sont dans un couloir avec trois frontières vers la gauche et une frontière très éloignée à droite. L'affectation de *MPG* est similaire à celle de *MinPos* mais permet de calculer l'affectation des robots aux frontières est plus durable.

L'affectation peut s'effectuer avec une fréquence moins élevée. En effet, avec *MinPos* les robots $R2$ et $R3$ se sépareront dès que $R2$ sera en première position pour $F3$ mais ceci nécessite de recalculer fréquemment les affectations ; avec *MPG* les robots $R2$ et $R3$ sont directement affectés aux frontières $F3$ et $F2$ respectivement. Le problème du choix de la fréquence de calcul des affectations est abordé dans le chapitre suivant.

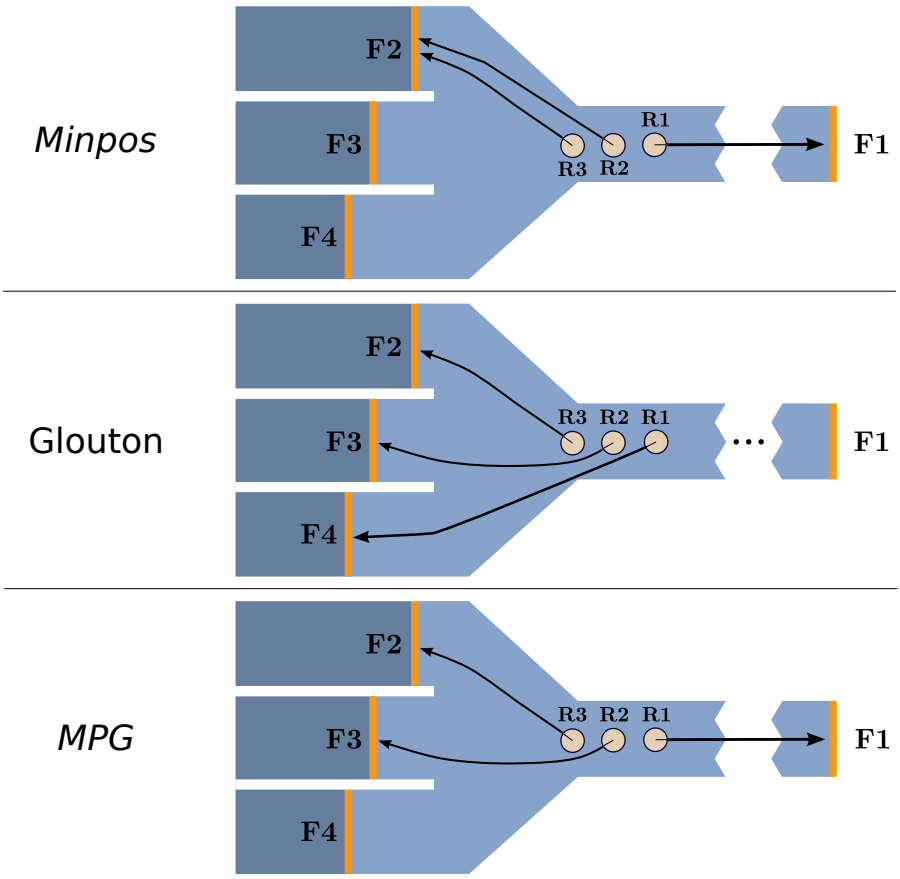


FIGURE 4.13 – Comparaison des assignations *MinPos* (en haut) Glouton (au centre) et *MPG* (en bas).

Le tableau 4.1 fait la synthèse des complexités des principaux algorithmes d'affectation et de leurs avantages. Dans ce tableau, l'équilibre de la répartition indique si l'algorithme respecte le critère 3.3 i.e. le nombre de robots affectés par frontière est équilibré. La prise en compte des directions indique si l'algorithme prend en compte la position du robot vers les frontières. *MinPos* a une complexité intermédiaire entre *MinDist* et Glouton et considère les directions des frontières. *MPG* a une complexité de même ordre que Glouton tout en prenant en compte des directions.

TABLE 4.1 – Comparaison des complexités et caractéristiques des principaux algorithmes d'affectation de frontières

Algorithme	Complexité	Équilibre de la répartition	Prise en compte des directions
MinDist	$O(m)$	-	-
Glouton	$O(n^2m)$	+	-
Algo. Hongrois	$O(n^3)$	+	-
MinPos	$O(nm)$	-	+
MPG	$O(n^2m)$	+	+

Dans le chapitre suivant nous évaluons expérimentalement en simulation les performances des algorithmes proposés ainsi que l'effet de la fréquence de leur exécution.

Chapitre 5

Etude des performances

Sommaire

La première partie de ce chapitre présente les résultats expérimentaux, en simulation, de l'évaluation des algorithmes *MinPos* et *MPG* en les comparant aux approches les plus courantes de la littérature. Dans la seconde partie, nous étudions l'impact des fréquences d'affectation sur les performances de chaque algorithme.

5.1 Comparaisons aux approches standards

L'évaluation des méthodes proposées a été effectuée sur deux simulateurs que nous avons développés pendant la thèse. L'un permet de mesurer les performances des algorithmes comparés en écartant les coûts de calculs liés à la planification de trajectoire non-holonomes, l'autre, plus réaliste, est très proche de la réalité ce qui permet de valider le code qui sera ensuite téléchargé sur les robots réels.

La figure 5.1 résume les fonctions de ces simulateurs. La principale fonction du simulateur est de fournir la matrice de coûts nécessaire aux algorithmes d'affectation. Ceux-ci, fournissent en retour une affectation à un robot. Ceci est effectué, pour chaque robot, régulièrement ou à chaque pas de simulation. Le simulateur planifie les trajectoires des robots, réalise les observations des capteurs des robots et construit la carte. À partir de la carte construite et des déplacements des robots les coordonnées des frontières et des robots sont calculées. Les distances de chaque robot vers chaque frontière sont déterminées permettant de construire la matrice de coûts.

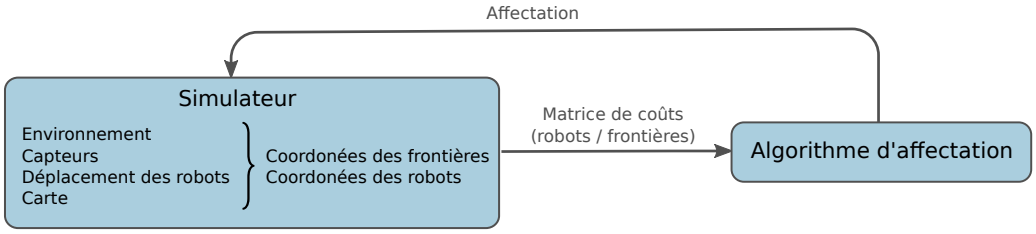


FIGURE 5.1 – Schéma de fonctionnement des simulateurs.

Le premier simulateur est programmé en JAVA, la modélisation du monde est simple : l'environnement et le temps sont discrets, un agent occupe une cellule, connaît sa position et perçoit parfaitement l'environnement à 360° autour de lui avec un rayon paramétrable. Il n'y a pas de différence entre la carte de navigation et la grille de configuration i.e. les robots n'ont pas de volume, n'ont pas d'orientation et sont holonomes. Un agent se déplace d'une cellule à chaque itération si il n'y a pas de robot occupant la cellule où il veut se déplacer. Ainsi, lors d'un pas de simulation chaque agent effectue un mouvement (un changement de cellule). Le système est synchrone ; chaque robot dispose de sa propre carte qui est mise à jour avec les informations des autres robots à chaque début d'itération du simulateur (hypothèse de communication totale et parfaite).

Le second simulateur est en C++. Il utilise une bonne partie des classes fonctionnant réellement sur les robots MiniRex (décrit en section 8.2). Il est donc plus proche de l'implémentation sur robots réels ce qui permet d'utiliser **exactement les mêmes parties de code en simulation et sur les robots**. Sur ce deuxième simulateur, la taille des robots est prise en compte. Les environnements considérés sont aussi de taille réelle. Le temps n'est plus discrétisé en pas de simulation mais il est calculé en utilisant une approximation du temps de déplacement des robots. Le système est asynchrone chaque robot calcule son affectation après avoir parcouru une certaine distance (paramétrable) ou après avoir atteint son objectif. La non-holonomicité et la dynamique des robots sont prises en compte. À chaque pas de simulation, le robot qui effectue son déplacement est le robot qui a utilisé le moins de temps pour ses déplacements. Ce simulateur, plus réaliste, permet d'identifier certains problèmes (par exemple des bugs de programmation) et de les résoudre sur le simulateur ce qui est plus rapide que d'identifier et de résoudre les problèmes sur les robots.

Les expériences ont été réalisées sur différents types d'environnement repris de la littérature ou que nous avons définis. Ceux-ci comprennent :

- des labyrinthes générés aléatoirement,
- des grilles régulières et irrégulières,
- des cartes de bâtiments réels comme une section d'hôpital, l'étage 4 d'un bâtiment du MIT modifiées pour que les portes n'apparaissent pas,
- des cartes construites manuellement comme les écuries.

La figure 5.2 illustre certains de ces environnements.

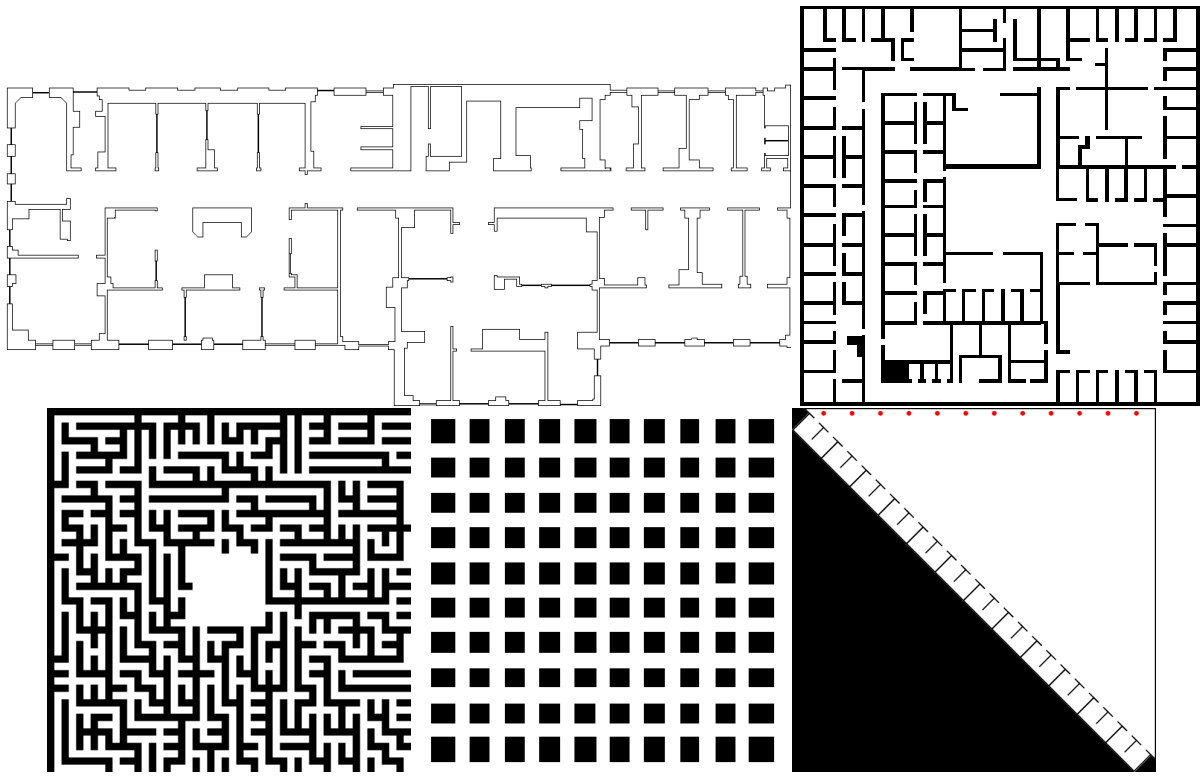


FIGURE 5.2 – Environnements utilisés pour les expériences (dans le sens de lecture) : une section d'hôpital (de Player/Stage (Gerkey et al., 2003)), des bureaux, un labyrinthe, une grille régulière et l'environnement écurie.

Au début d'une simulation, mis à part pour l'environnement écurie, les robots sont placés aléatoirement dans un carré situé au centre de l'environnement d'une taille pouvant contenir l'ensemble des robots. Pour l'environnement écurie, les robots sont alignés nord de l'environnement. Une simulation se termine lorsqu'un robot a construit la totalité de la carte. Le temps est alors mesuré en nombre de pas de simulation pour le simulateur JAVA et en seconde pour le simulateur C++ en utilisant le temps de déplacement calculé par la planification de trajectoire (décrite en section 8.5) qui utilise une approximation du temps de déplacement des robots MiniRex (voir section 8.2).

5.2 Mesures

Pour l'évaluation des algorithmes proposés, nous les comparons avec les algorithmes *MinDist*, Glouton et Glouton sur l'utilité Burgard et al. (2005). Nous comparons les performances en temps d'exploration. L'identification des frontières (voir chapitre 7) est effectuée de manière identique pour tous les algorithmes, excepté pour l'approche fondée sur l'utilité et pour *MinDist* qui utilise la cellule frontière la plus proche.

Les figures 5.3 (environnement section d'hopital) et 5.4 (environnement labyrinthe) présentent deux campagnes de mesure du temps moyen d'exploration en fonction du nombre de robots sur le simulateur en JAVA. Les temps d'exploration sont donnés en pas de simulation. Les algorithmes comparés sont *MinDist* (algorithme 1), une implémentation d'un algorithme

Glouton (algorithme 2) fondée sur l'utilité (Burgard et al., 2005) et *MinPos* (algorithme 3).

Sur ces deux environnements, nous observons que les algorithmes Glouton et *MinPos* sont nettement plus performants que l'algorithme*MinDist*, l'écart est d'autant plus marqué que le nombre d'agents augmente (jusqu'à 20%). Sur l'environnement labyrinthe (taille 60x60 cellules), les performances de Glouton et *MinPos* sont également bien meilleures que *MinDist*.

Dans les deux figures les performances de MinPos et Glouton sont proches. Toutefois, une différence est significative : quand le nombre de robots est faible (entre 2 et 5 ou 10 suivant l'environnement), l'algorithme *MinPos* a de meilleures performances. En effet, l'approche gloutonne optimise seulement les distances et ne distribue pas les robots dans des directions différentes, ce qui a un impact lorsque les robots, peu nombreux, doivent "se partager" l'espace. En revanche, quand le nombre de robots devient plus important, l'approche de Burgard et al. (2005) a des performances équivalentes ou légèrement meilleures.

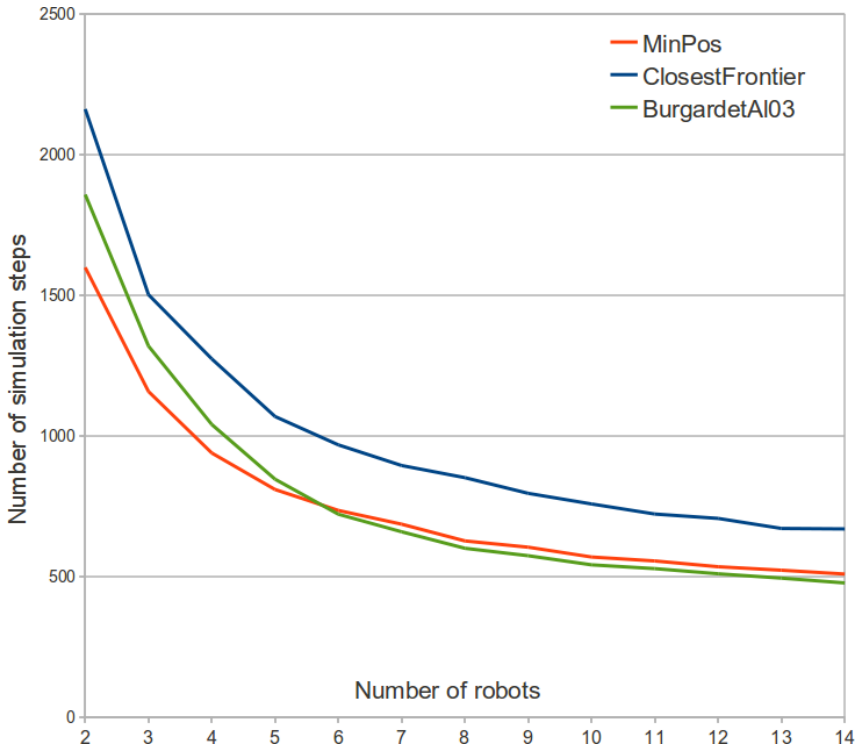


FIGURE 5.3 – Temps d'exploration sur l'environnement section d'hôpital (chaque valeur est une moyenne de 60 simulations).

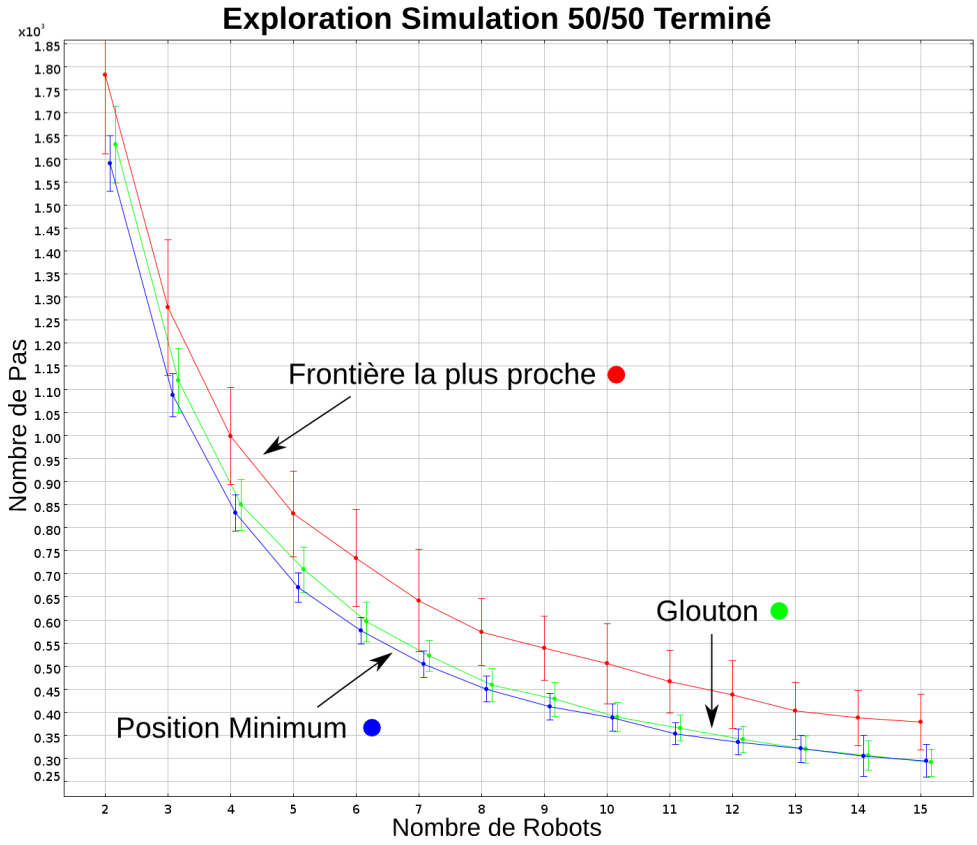


FIGURE 5.4 – Temps d'exploration d'un labyrinthe 60x60 cellules, pour un nombre croissant d'agents (chaque valeur est une moyenne de 50 simulations).

Le simulateur C++ fournit des résultats similaires au simulateur JAVA. La figure 5.5 illustre le résultat sur des labyrinthes générés aléatoirement toutes les dix simulations de chaque algorithme. Pour étudier la qualité de coopération de ces algorithmes nous avons tracé la borne T_{1R}/n où T_{1R} est le temps d'exploration moyen mesuré avec un robot et n le nombre de robots. Elle correspond à une borne inférieure du temps d'exploration.

Nous pouvons observer que les quatre algorithmes suivent la même tendance que la borne. Les trois algorithmes "performants" Glouton, *MinPos* et *MPG* sont très proches en temps d'exploration de la borne T_{1R}/n par rapport à *MinDist*. Ceci est particulièrement vrai lorque le nombre de robots est faible (n<5). Ceci est dû au fait que les robots démarrent de la même zone et doivent se séparer avant de pouvoir observer des zones différentes. Ce problème est moins présent quand les robots sont peu nombreux car ils se séparent plus rapidement.

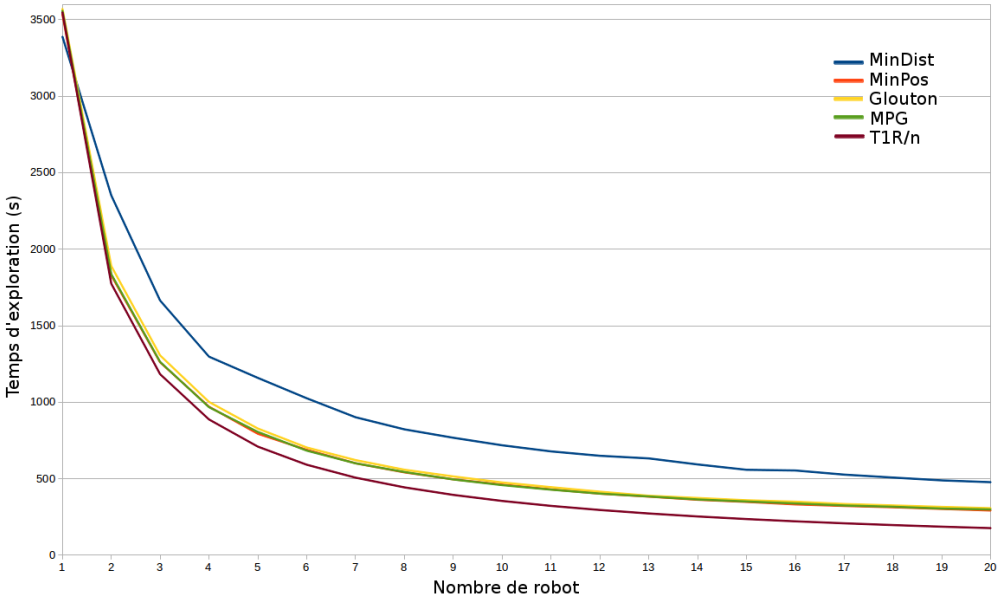


FIGURE 5.5 – Résultat de la simulation de l'exploration de labyrinthes générées aléatoirement (simulateur C++).

Avec l'environnement écurie, *MinPos* reste meilleur, de 18% en moyenne, quel que soit le nombre de robots. Ceci est illustré par le graphique 5.6. Cette différence s'explique par le fait que selon la proximité des frontières entre elles vis-à-vis de la position des robots, l'algorithme *MinPos* va empêcher des assignations à des frontières proches.

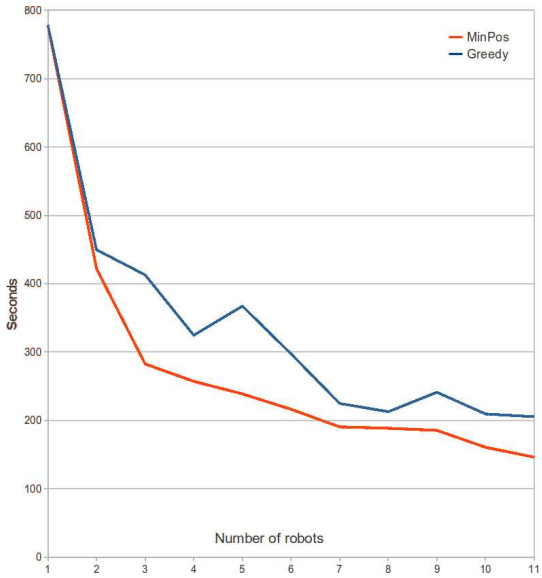


FIGURE 5.6 – Résultat de la simulation de l'exploration de l'environnement écuries (simulateur C++).

5.3 Impact des fréquences d'observation et de décision

Dans cette section, nous étudions l'impact des fréquences d'observation et de l'exécution de l'algorithme d'affectation sur les performances des algorithmes.

Les simulations présentées ont été effectuées en considérant que les acquisitions des capteurs de chaque robot sont immédiatement utilisées pour mettre à jour les cartes locales des robots, et que tous les robots recalculent alors leur affectation.

Ces hypothèses ne sont pas réalistes pour une implémentation réelle (sur des robots). La communication entre robots et la capacité de calcul des robots contraignent fortement la fréquence de mise à jour des cartes et la fréquence de calcul des affectations.

Une autre hypothèse utilisée dans les simulations présentées précédemment est que les acquisitions des capteurs s'effectuent à chaque fois qu'un robot s'est suffisamment déplacé pour modifier la perception. Comme la carte construite est une grille, les observations sont effectuées à chaque fois que le robot change de cellule avec le simulateur JAVA et à une fréquence très elevée avec le simulateur C++. Cette hypothèse n'est pas un cadre réaliste pour un système robotique.

Dans cette section, nous levons ces hypothèses simplificatrices en proposant une méthode de mise à jour régulière des cartes. La fréquence de mise à jour est paramétrable.

Notons

- F_{obs} la fréquence d'observation,
- F_{aff} la fréquence d'affectation,
- F_{envoi} la fréquence à laquelle un robot communique sa carte,
- $F_{reception}$ la fréquence à laquelle le robot lit les cartes reçues des autres robots.

Il est évident qu'il est inutile qu'un robot communique sa carte si elle n'a pas été modifiée et il est inutile que le robot lise les communications reçues si aucun robot n'a envoyé de carte. De même, il est inutile qu'un robot recalcule son affectation s'il n'a pas de nouvelles informations (provenant d'une observation ou de la réception d'une carte) qui pourraient modifier son choix. Nous avons donc les relations suivantes :

$$\begin{cases} F_{obs} \geq F_{envoi} \geq F_{reception} \\ F_{aff} \leq max(F_{obs}, F_{reception}) \end{cases} \tag{5.1}$$

dont nous déduisons :

$$\begin{cases} F_{obs} \geq F_{envoi} \geq F_{reception} \\ F_{obs} \geq F_{aff} \end{cases} \tag{5.2}$$

La fréquence d'observation est donc la limite supérieure des autres fréquences.

Pour ce chapitre, nous considérons que la fréquence d'affectation est égale aux fréquences de communication. Ceci n'est pas irréaliste si la fréquence d'affectation est peu élevée ne saturant ainsi pas la bande passante. Les robots communiquent afin d'échanger leur carte et leur position (coordonnées) avant de calculer leur affectation. Nous posons donc l'égalité suivante :

$$F_{aff} = F_{envoi} = F_{reception} \tag{5.3}$$

Nous verrons plus en détail la stratégie de communication en section 8.4.3 et comment elle est réalisée sur les robots.

5.3.1 Fréquence d'observation

Pendant l'exploration, d'un point de vue global, les frontières apparaissent, disparaissent et sont repoussées grâce aux observations des robots. Le processus d'observation comprend l'acquisition des informations délivrées par un capteur et leurs traitements en particulier pour la mise à jour de la carte. La fréquence d'observation est liée aux caractéristiques techniques du capteur : elle ne peut être supérieure à la fréquence d'acquisition du capteur. Il est préférable d'effectuer les observations à la même fréquence que les acquisitions du capteur (le plus fréquemment possible) car ceci permet d'accroître la précision de la carte et de découvrir une partie de l'environnement le long du chemin parcouru. La figure 5.7 illustre la différence des zones perçues avec des fréquences d'observation différentes en environnement ouvert. Cette différence peut être encore plus marquée avec des environnements d'intérieurs. Par exemple sur la situation illustrée en figure 5.8, le robot passe devant un couloir. En observant le long du chemin, le robot découvre une partie importante du couloir tandis qu'en observant peu fréquemment, le couloir n'est presque pas découvert. Toutefois, avec le même exemple sans le couloir, il n'y aurait pas de différence de gain d'information entre les deux fréquences d'observation.

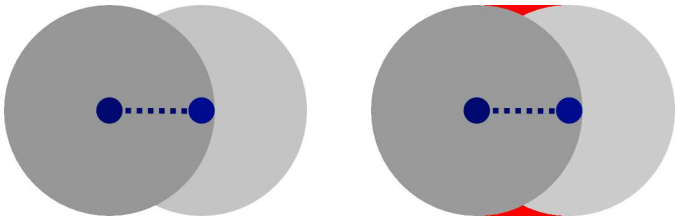


FIGURE 5.7 – Zone explorée quand le robot effectue une observation lorsqu'il arrive sur sa cible (à gauche) et quand sa fréquence d'observation est élevée (à droite). Les disques gris illustrent la zone perçue par le robot bleu. Le rouge illustre la différence entre les deux fréquences d'observation.

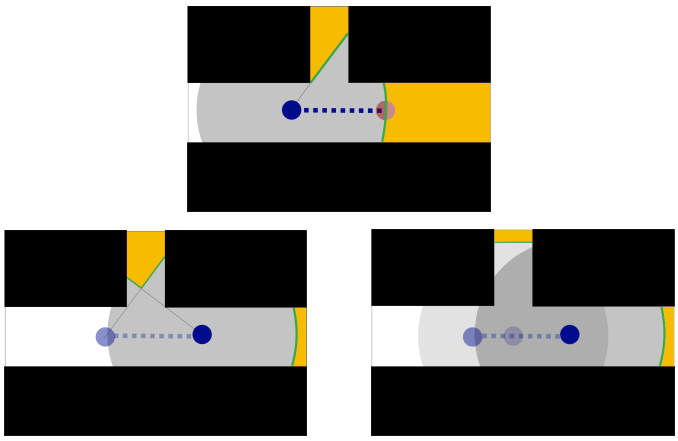


FIGURE 5.8 – En haut, état initial de l'exemple : le disque gris illustre la zone perçue par le robot bleu celui en rouge la position planifiée. La frontière est en vert, le blanc montre la zone explorée, le orange celle inexplorée. En bas, zone explorée avec une fréquence d'acquisition faible, quand le robot arrive sur sa cible (à gauche), et élevée (à droite).

Le gain d'information est en général faible avec une fréquence d'observation élevée ; une grande fréquence est surtout utile pour la localisation du robot (voir la section 8.4). De plus, le traitement des observations peut être coûteux en temps de calcul, particulièrement avec des capteurs trois dimensions. Suivant les capacités de calcul des robots et le capteur utilisé, les robots ne pourront pas forcément traiter la totalité des informations provenant d'un capteur. Par exemple, le télémètre laser Hokuyo UTM-30LX génère 43 000 points 2D par seconde ce qui est raisonnable pour être traité en temps réel avec une puissance de calcul limitée. En revanche, la caméra 3D Kinect génère $9,2 * 10^6$ points 3D par seconde, ce qui est difficilement traitable même avec une puissance de calcul importante. Si le robot est capable de gérer toutes les acquisitions du capteur alors il est intéressant de toutes les ajouter dans la carte, sinon il faudra choisir lesquelles sont importantes.

Certains capteurs nécessitent des acquisitions à l'arrêt pour éviter les perturbations dues au mouvement. Par ailleurs, un robot n'a pas toujours les capacités de calcul nécessaires pour traiter l'information des acquisitions tout en naviguant. Dans ce cas, le robot observera uniquement lorsqu'il sera arrêté sur la cible planifiée. La fréquence d'affectation est alors la plus élevée possible car elle est égale à la fréquence d'observation..

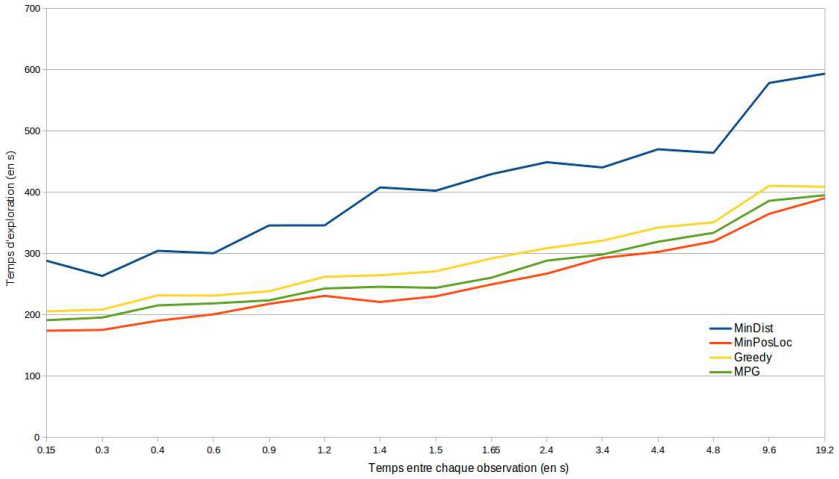


FIGURE 5.9 – Temps d'exploration de l'environnement MIT avec 10 robots en faisant varier la fréquence d'observation.

La figure 5.9 illustre les résultats de l'exploration de l'environnement MIT avec des fréquences d'observation différentes. La fréquence d'affectation est la plus basse possible c'est-à-dire que les robots calculent leur affectation quand ils ont atteint leur frontière. Nous pouvons observer que, comme attendu, les temps d'exploration augmentent avec la période d'observation. Les différences en performance entre les différents algorithmes d'affectation comparés restent similaires.

5.3.2 Fréquence d'affectation des cibles

La fréquence d'affectation des cibles aux robots a un impact important sur les performances du système. En général, plus la fréquence d'affectation est élevée, moins le temps d'exploration total est élevé. En effet, la coopération est plus efficace car les robots calculent leurs affectations avec des informations plus récentes (prenant en compte les frontières créées et détruites). En revanche, si les robots changent de cibles entre deux affectations les performances peuvent aussi se

dégrader. Les robots peuvent osciller entre plusieurs cibles et ne plus avancer (voir section 4.4.3). Cette dernière situation peut avoir lieu avec les affectations qui garantissent un équilibre de la répartition des robots sur les frontières (respect du critère 3.3). En effet, il est fréquent que des frontières apparaissent puis disparaissent rapidement par exemple lorsqu'un robot découvre un couloir donnant sur une petite impasse. La stabilité des directions des cibles affectées aux robots est donc utile pour obtenir de bonnes performances. Les affectations réalisées par l'algorithme *MinPos* sont plus stables qu'avec les approches gloutonnes car l'algorithme n'est pas sensible à l'apparition et la disparition de frontière derrière un robot. C'est-à-dire qu'un robot aura la même position/rang pour toutes les frontières ayant des robots plus proches.

Impact sur l'observation d'une frontière

Plus la fréquence d'affectation est élevée meilleures seront les performances d'exploration. Cette affirmation est compréhensible par un exemple simple illustré en figure 5.10. Dans cet exemple, un robot est dans un couloir et sa frontière affectée est à la limite du bout de l'impasse. Il est clair que le plus tôt le robot effectuera une observation suivie d'un calcul d'affectation, plus tôt il fera demi-tour pour explorer une autre partie de l'environnement. Cet exemple est une situation courante pendant l'exploration, elle apparaît pour toutes les frontières qui disparaissent après une observation.

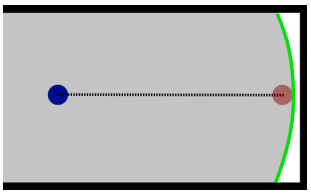


FIGURE 5.10 – Robot explorant une impasse. Les murs sont de couleur noire. Le robot est illustré par le disque bleu et sa cible est en rouge. Le vert montre la frontière.

Coopération implicite

La fréquence d'affectation joue un rôle important pour la coopération, particulièrement avec les algorithmes d'affectation reposant sur la coordination implicite. Sur l'exemple des deux robots dans l'intersection en T à nouveau illustré en figure 5.11, le résultat de l'execution de *MinPos* sur le robot $R1$ dépend de la mise à jour des coordonnées du robot $R2$ pour que les robots se répartissent sur les deux frontières existantes. En comparaison, le résultat de l'execution de *MinDist* dépend de la mise à jour de la carte avec les acquisitions du robot $R2$ afin que la frontière $F1$ soit repoussée et par conséquent soit plus éloignée de $R1$ que $F2$. *MinPos* nécessite donc des mises à jour plus fréquentes des positions que des cartes.

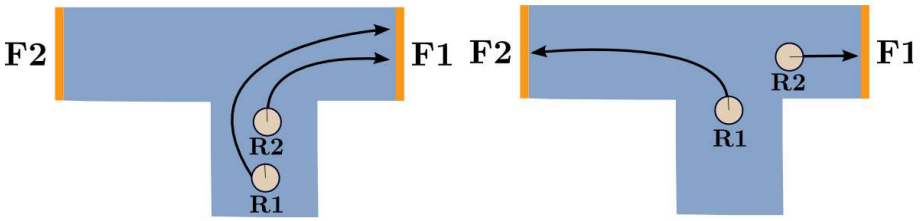


FIGURE 5.11 – Deux robots se séparant sur une intersection en T avec l'algorithme *MinPos*.

La figure 5.12 illustre les résultats de l'exploration de l'environnement MIT avec des fréquences d'affectation décroissantes. La fréquence d'observation est fixée à la même valeur que la fréquence d'affectation. Nous pouvons observer que, comme attendu, les temps d'exploration augmentent avec la période d'affectation. La stabilisation avec des fréquences faibles est attendue car les robots recalculent leur affectation quand ils atteignent leur frontière. L'écart entre *MinPos* et Glouton se resserre tandis qu'il reste stable entre *MPG* et Glouton. Ceci s'explique par le fait que *MinPos* repose sur la coopération implicite nécessitant des affectations le plus souvent possible. Pour la même raison, *MPG* devient plus performant que *MinPos* lorsque la fréquence d'affectation devient faible.

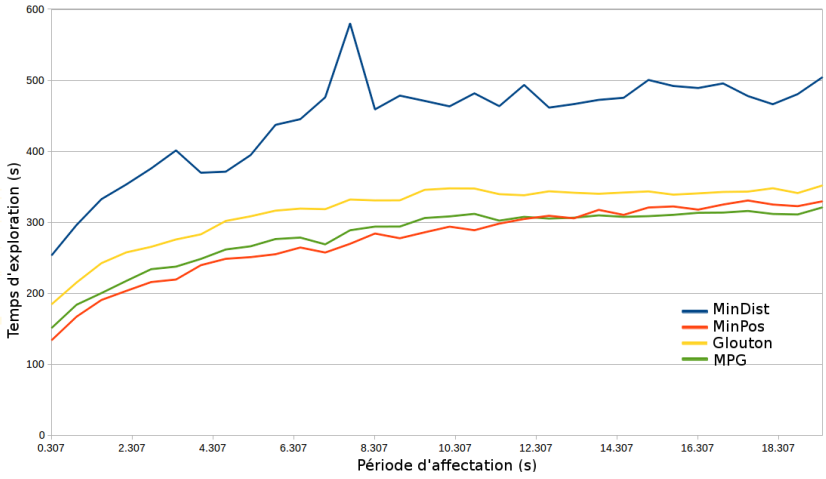


FIGURE 5.12 – Temps d'exploration de l'environnement MIT avec 15 robots en faisant varier la période d'affectation.

5.3.3 Conclusion

Sans surprise, les performances des algorithmes d'affectation sont meilleures lorsque les fréquences d'observation et d'affectation sont élevées. Toutefois, la fréquence d'affectation doit être limitée car l'affectation est coûteuse en temps de calcul et nécessite de communiquer avec les autres robots. La fréquence d'observation peut aussi être limitée mais une fréquence élevée est souvent nécessaire pour la localisation du robot.

Nous distinguons les deux cas suivants distingués par le coût en calcul des observations. En 2D, les observations sont peu coûteuses en calcul, tandis qu'en 3D, les observations sont relativement très coûteuses. et la localisation en 3D du robot peut-être fondée sur la localisation

2D et, par conséquent, ne nécessite pas une fréquence d'observation élevée. Nous avons donc défini deux stratégies en fonction du type d'exploration (2D ou 3D).

Exploration 2D

Pour l'exploration 2D, la fréquence d'observation est égale à la fréquence d'acquisition du capteur principalement pour la localisation des robots. Les robots envoient leur carte après avoir atteint leur cible ou à chaque fois qu'une certaine distance a été parcourue, et réceptionnent les cartes envoyées par les autres robots avant de recalculer leur affectation :

$$F_{obs} \gg F_{aff} = F_{envoi} = F_{reception} \tag{5.4}$$

Exploration 3D

Pour l'exploration 3D, les fréquences choisies sont identiques aux fréquences de l'exploration 2D à l'exception de la fréquence d'observation qui doit être limitée car le traitement des informations est très coûteux. Les robots effectuent donc des observations uniquement quand ils ont atteint leurs frontières.

$$F_{obs} = F_{aff} = F_{envoi} = F_{reception} \tag{5.5}$$

5.4 Conclusion

Dans ce chapitre nous avons étudié les performances des algorithmes proposés (*MinPos* et *MPG*). Les performances de *MinPos* sont similaires à celles de Glouton et dans certains cas meilleures. Nous avons aussi étudié l'impact de la fréquence d'observation et d'affectation. Il apparaît que *MPG* est plus performant que *MinPos* lorsque la fréquence d'affectation est faible. Nous tiendrons compte de ces observations pour les expériences robotiques décrites au chapitre 9. Dans le chapitre suivant nous décrivons des implémentations efficaces des algorithmes *MinPos*, Glouton et *MPG* utilisant une propagation de fronts de vagues à partir des frontières.

Chapitre 6

MinPos avec propagation de vagues

Sommaire

Les algorithmes *MinPos* et *MPG*, présentés dans le chapitre 4, utilisent la matrice de coût (distance des robots aux frontières) pour déterminer l'affectation d'un robot. Le calcul de cette matrice est relativement coûteux car il nécessite de déterminer le chemin le plus court de chaque robot à chaque frontière. Dans ce chapitre, nous proposons une méthode de calcul des chemins fondée sur la propagation de fronts de vagues depuis les frontières. Comme nous le verrons cette approche donne une implémentation très efficace des algorithmes *MinPos*, Glouton et *MPG*.

6.1 Évaluation de la position

Notre démarche est de considérer le calcul des distances en partant des frontières plutôt que des robots. Nous optons pour un algorithme de construction d'un champ de potentiel centré sur la frontière car il permet de connaître les plus courts chemins en s'éloignant progressivement de la frontière. Nous verrons dans les sections suivantes comment cette propriété est exploitée pour limiter les calculs nécessaires à l'affectation des frontières.

6.1.1 Propagation de front de vague

Pour le calcul de la matrice de coût, nous utilisons l'algorithme de propagation de front de vague (Barraquand et al., 1991) qui construit un champ de potentiel artificiel avec un seul minimum sur la source.

Un champ de potentiel définit une valeur (un potentiel) en tout point de l'espace (discret ou continu). Notons $\mathcal{X}$ l'ensemble des cellules de l'environnement. La fonction de potentiel ϕ associe une valeur (un réel ou un entier) à chaque cellule de l'environnement :

$$\phi : \mathcal{X} \rightarrow \mathbb{R} \tag{6.1}$$

Les champs de potentiels permettent de guider la navigation d'un ou plusieurs robots vers un but en évitant les obstacles (Khatib, 1986). En effet, placer un champ attracteur sur le but et des champs répulsifs sur les obstacles, permet de calculer un gradient descendant vers l'objectif à atteindre ($\overrightarrow{v} = -\nabla\phi$). La figure 6.1 illustre un champ avec un but et deux obstacles.

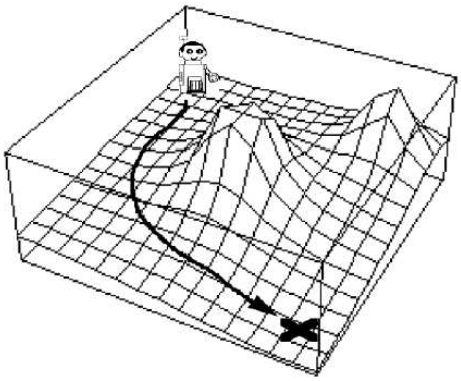


FIGURE 6.1 – Illustration d'un champ de potentiel en trois dimensions (Ferber, 1995)

Le principal défaut du contrôle par champs de potentiels est le risque d'évoluer vers un minimum local, par exemple, lorsque l'environnement présente des obstacles concaves. La méthode de Barraquand et al. (1991) résout ce problème en calculant un seul champ descendant vers une source, sans minima locaux. Il donne, pour tout point de l'espace (discret), la distance du plus court chemin à la source. Cet algorithme est réputé coûteux en calcul mais dans notre cas il s'avèrera efficace car nous utilisons une propagation de vague pour connaître les distances de n robots à une source (frontière). Nous verrons plus loin que plusieurs optimisations sont possibles, le rendant ainsi très efficace en comparaison aux approches de type A*.

Nous précisons maintenant la définition de l'algorithme de la vague. Il consiste à calculer incrémentalement, sur l'espace de configuration ($\mathcal{C}_{free}$), l'ensemble des cellules situées à la même distance de la source [2] – le front de vague –, à partir de la source initialisée à 0. Notons $\mathcal{W}_i$, l'ensemble des cellules du front de vague à la distance i et $\mathcal{X}_G$ la ou les cellules but (les cellules d'une frontière) de potentiel 0. L'algorithme 5 précise le calcul de la vague. Cet algorithme a

2. distance du chemin parcouru depuis la source

une complexité en $O(n)$ où n est le nombre de cellules atteignables.

Algorithme 5: Algorithme de propagation de front vague (Barraquand et al., 1991)

1 Initialiser $\mathcal{W}_0 = \mathcal{X}_G; i = 0$
2 Initialiser $\mathcal{W}_{i+1} = \varnothing$
3 **Pour chaque** $x \in \mathcal{W}_i$ **faire**
4 $\quad \phi(x) = i$
5 $\quad$ insérer toutes les cellules voisines $y \in \mathcal{C}_{free}$ de x dans $\mathcal{W}_{i+1}$
6 **fin**
7 **si** $\mathcal{W}_{i+1} = \varnothing$ **alors** terminer **sinon** i=i+1 et aller à l'étape 2

6.1.2 Adaptation à *MinPos*

Notre approche consiste à calculer des vagues depuis les cellules frontières, afin de connaître, pour tout point de l'environnement, la distance du plus court chemin à ces frontières. Le champ calculé d'une frontière donne le coût en distance pour tous les robots situés dans le champ.

L'environnement est discrétisé en cellules contenant une information sur son statut (*occupé*[3], *vide* ou *inexploré*[4]). Sur cet espace discret, l'algorithme de Barraquand et al. (1991), ci-après nommé "calcul de la vague", propage un front de vague à partir des frontières. La propagation s'effectue en utilisant une fonction de voisinage qui renvoie les cellules voisines *vides* d'une cellule. Cette méthode est du même type qu'un parcours en largeur d'abord dans un graphe.

La figure 6.2 illustre une vague calculée depuis un point sur un environnement constitué de plusieurs pièces.

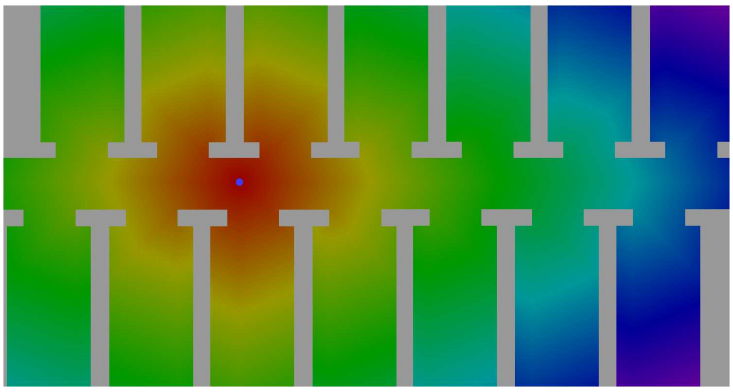


FIGURE 6.2 – Résultat du champ de potentiels calculé à partir d'une frontière avec l'algorithme 5 en utilisant un voisinage de 8 cellules. La couleur d'une cellule est fonction de la distance à la frontière.

La fonction de voisinage peut être en 4-connexité ou 8-connexité (voir figure 6.3). Nous utilisons la fonction en 8-connexité car malgré la nécessité d'effectuer une approximation pour les cases en diagonales, elle estime mieux la distance réelle. Un voisinage différent pourrait être utilisé pour améliorer l'estimation mais nécessite d'agrandir le voisinage ce qui rend cette méthode plus complexe (Strand and Normand, 2012). La propagation avec fonction de voisinage

3. par occupé nous entendons occupé par un obstacle et non par un robot
4. non observé par un robot

en 8-connexité utilise une approximation de $\sqrt{2}$ et ajoute les cellules voisines dans une file de priorité (*priority queue*) de façon à traiter la cellule la plus proche de l'origine en premier pour ne propager qu'une seule fois chaque cellule. La propagation s'effectue en premier sur les cellules horizontales et verticales puis sur les diagonales.

	1	
1	0	1
	1	

$\sqrt{2}$	1	$\sqrt{2}$
1	0	1
$\sqrt{2}$	1	$\sqrt{2}$

7	5	7
5	0	5
7	5	7

FIGURE 6.3 – Voisinage et distance en 4, en 8-connexité et en 8-connexité avec des entiers.

La figure 6.4 illustre l'exécution de l'algorithme 5 avec un voisinage de 4 et 8 cellules.

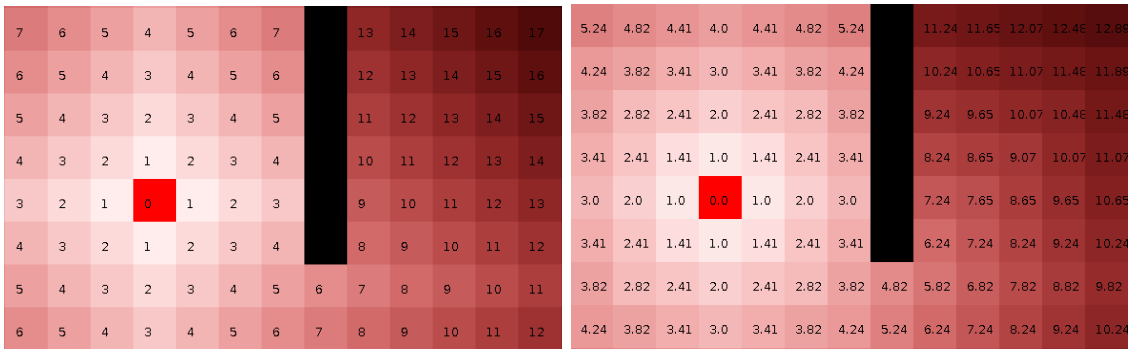


FIGURE 6.4 – Résultat des champs de potentiels calculé avec l'algorithme 5, à partir d'une frontière d'une cellule initialisée à 0 en utilisant un voisinage 4-connexité (à gauche) et 8-connexité (à droite). Les valeurs affichées sont les distances à la frontière.

6.1.3 Arrêt du calcul de la propagation de vague

Pour le calcul de la matrice de coût, une vague est propagée par frontière. Chaque vague peut être arrêtée quand tous les robots ont été atteints par la vague. À ce moment, la distance de l'ensemble des robots vers la frontière est connue.

Dans le cas du calcul de la position d'un robot, noté R_i, vers une frontière, il n'est pas nécessaire de propager la vague dans tout l'environnement. Afin de limiter le coût calculatoire pour le calcul des positions des robots vers une frontière, l'exécution de l'algorithme de propagation de front de vague peut être arrêté dès qu'il atteint la cellule occupée par le robot qui calcule son assignation. En effet, lorsque le robot R_i est atteint, tous les robots plus proches l'ont déjà été. Dans *MinPos*, la position d'un robot est déterminée par le nombre de robot plus proches (de la frontière que lui). Pour les robots plus éloignés, il n'est donc pas nécessaire d'évaluer leur distance, il suffit de mettre leur valeur à $+\infty$ (grande valeur).

La figure 6.5 illustre la propagation de deux vagues, depuis deux frontières de couleur bleu et rouge, sur un environnement constitué de plusieurs pièces. Ici, les vagues sont arrêtées lorsqu'elles rencontrent le seul robot (en jaune).

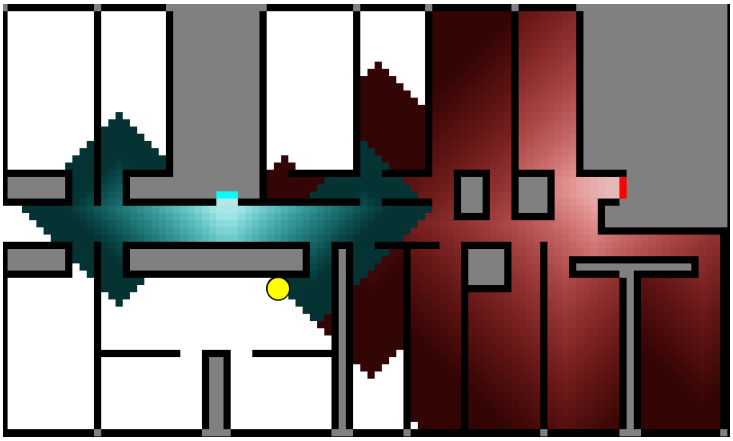
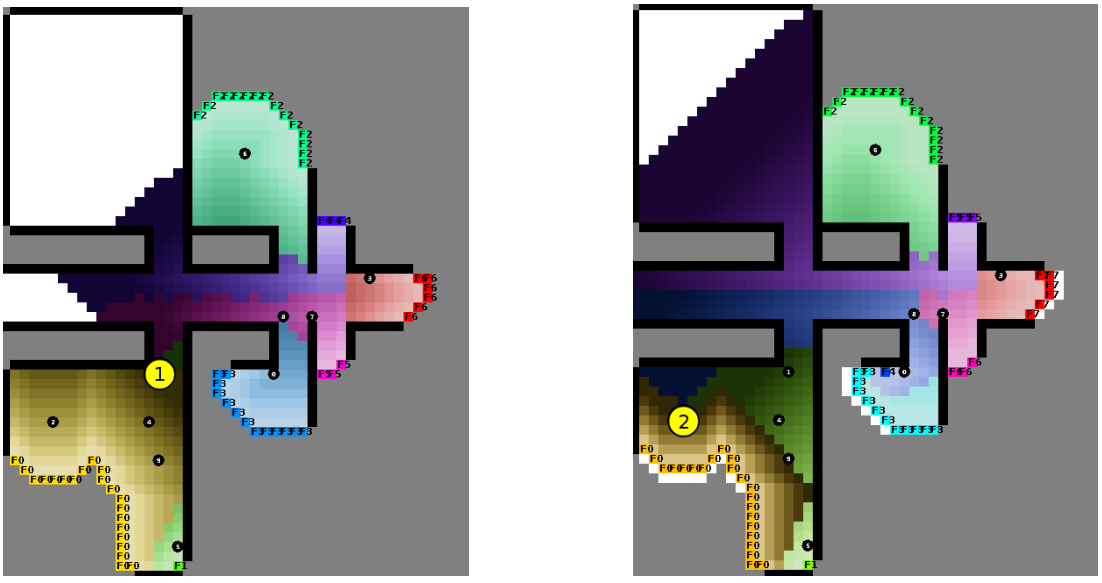


FIGURE 6.5 – Résultat du champ de potentiels calculé à partir de deux frontières avec l'algorithme 5 en utilisant un voisinage de 4 cellules. La couleur d'une cellule est la couleur associée à la frontière la plus proche et sa luminosité décroit en fonction de la distance à la frontière.

Durant l'exploration menée par plusieurs robots, chaque robot calcule son affectation en propageant des vagues à partir des frontières jusqu'à ce qu'elles atteignent sa position courante. Les vagues sont donc propagées jusqu'à des distances différentes par chaque robot. La figure 6.6 illustre les vagues calculées depuis les frontières pour le robot $R2$ à gauche et pour le robot $R1$ à droite. Pour visualiser les vagues qui se recouvrent, nous affichons, en chaque cellule, uniquement celle de valeur minimale en distance. Nous pouvons observer qu'une même vague est plus ou moins propagée sur une figure que sur l'autre en fonction de la distance entre le robot et la frontière associée à la vague. Par exemple la vague jaune est moins propagée à droite car le robot $R1$ est plus proche de frontière jaune (F0) que le robot $R2$.



(a) Vagues calculées pour le robot jaune $\mathcal{R}_2$.

(b) Vagues calculées pour le robot jaune $\mathcal{R}_1$

FIGURE 6.6 – Illustration des vagues calculées avec l'implémentation de *MinPos* avec l'arrêt des vagues.

Nous terminons par l'illustration de l'exploration d'un environnement de pièces par sept robots (voir la série de figures 6.7). Nous affichons les vagues propagées pour l'affectation du robot jaune. Les zones inexplorées sont en gris, les murs en noir. Les autres cellules sont blanches lorsqu'aucune vague ne les a traversé. Il est clair que les vagues sont propagées seulement localement autour du robot calculant son affectation.

6.1.4 Comparaison à A*

Nous évaluons, ici, l'efficacité calculatoire de l'approche par propagation de front de vague pour le problème du calcul des distances robots-frontières. L'algorithme de la vague fournit la distance d'une frontière à toutes les cellules atteignables. L'alternative à cette approche est de calculer un plus court chemin de la frontière vers chaque robot. Un algorithme A* est classique et efficace car il utilise une heuristique (h) pour diriger la recherche vers la cible. Sa complexité dépend de l'heuristique utilisée. L'utilisation de la distance euclidienne comme heuristique est simple et généralement efficace.

Le nombre de nœuds calculés dépend alors de la topologie de l'environnement. Si de nombreux obstacles sont présents et en particulier si ils sont concaves[5] le nombre de nœuds calculés se rapprochera de celui de la propagation de vague. Si le nombre de robots est faible, calculer un chemin de la frontière vers chaque robot peut être plus efficace. En revanche, si le nombre de robots est important, calculer une propagation de front de vague sera bien plus efficace. L'efficacité de chaque technique peut être observée par le nombre de cellules où la distance à la frontière à été calculée. La figure 6.8 montre une comparaison entre A* et la propagation de front de vague en colorant les cellules où la distance a été recalculée plusieurs fois.

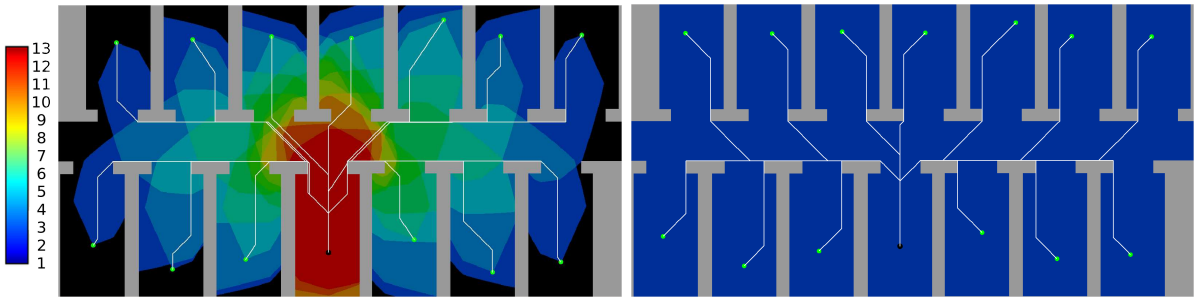


FIGURE 6.8 – Comparaison du nombre de fois où les cellules ont été calculées avec l'algorithme A* (à gauche) et de la vague (à droite). La couleur rouge représente les cellules où la distance à été recalculée autant de fois que de chemin calculé.

Pour évaluer l'efficacité de notre approche du calcul des distances, nous la comparons, en temps de calcul, aux approches existantes en faisant varier le nombre de robots. Pour cela, nous avons développé :

- un A* en 2D utilisant la distance euclidienne comme heuristique. Le A* calcule un chemin entre la frontière et un robot et recommence le calcul pour chaque robot.
- un A* multiobjectif dont l'heuristique est égale à la distance euclidienne minimum d'un robot parmi l'ensemble des robots non encore atteints par l'exploration du A*. Cette technique permet de diriger l'exploration vers les robots en évitant, comme avec la propagation

5. l'heuristique euclidienne guidant la recherche en fonction de la distance au but, la recherche persistera dans la concavité jusqu'à l'explorer complètement

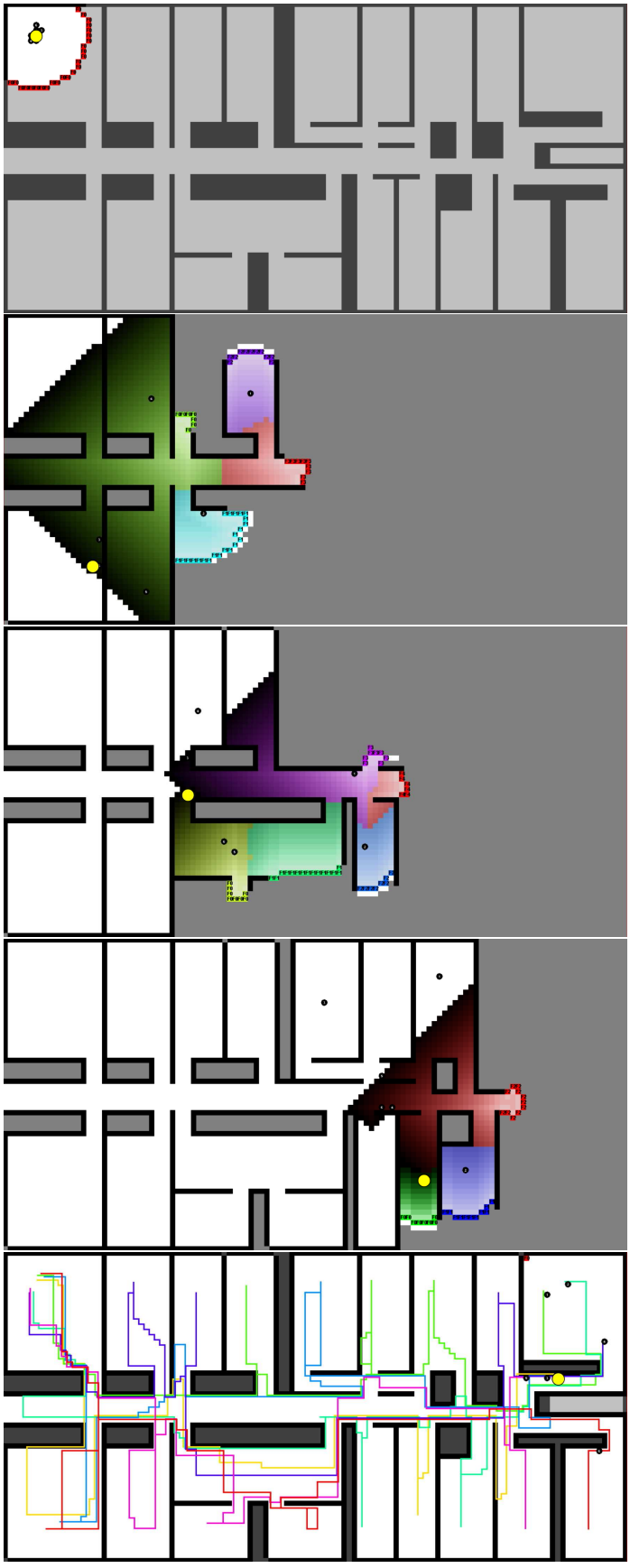


FIGURE 6.7 – Vagues propagées pour l'affectation du robot jaune.

de vague, de recommencer le calcul à partir de la frontière à chaque fois qu'un robot est atteint.

La figure 6.9 illustre les temps de calcul et le nombre de cellules où la distance a été calculée. Nous tirons aléatoirement un nombre de points inoccupés dans l'environnement (qui n'est pas occupé par un obstacle). Un de ces points est choisi comme frontière, les autres servant de robots. Nous calculons ensuite la distance de la frontière vers chaque robot avec les différents algorithmes comparés. Nous faisons varier le nombre de robots et les environnements car certains environnements favorisent l'utilisation de l'heuristique euclidienne. Par exemple, avec l'environnement "free" (environnement sans obstacle), la distance euclidienne donnera toujours une bonne approximation de la distance à parcourir tandis qu'avec les labyrinthes l'estimation sera beaucoup plus loin de la distance réelle.

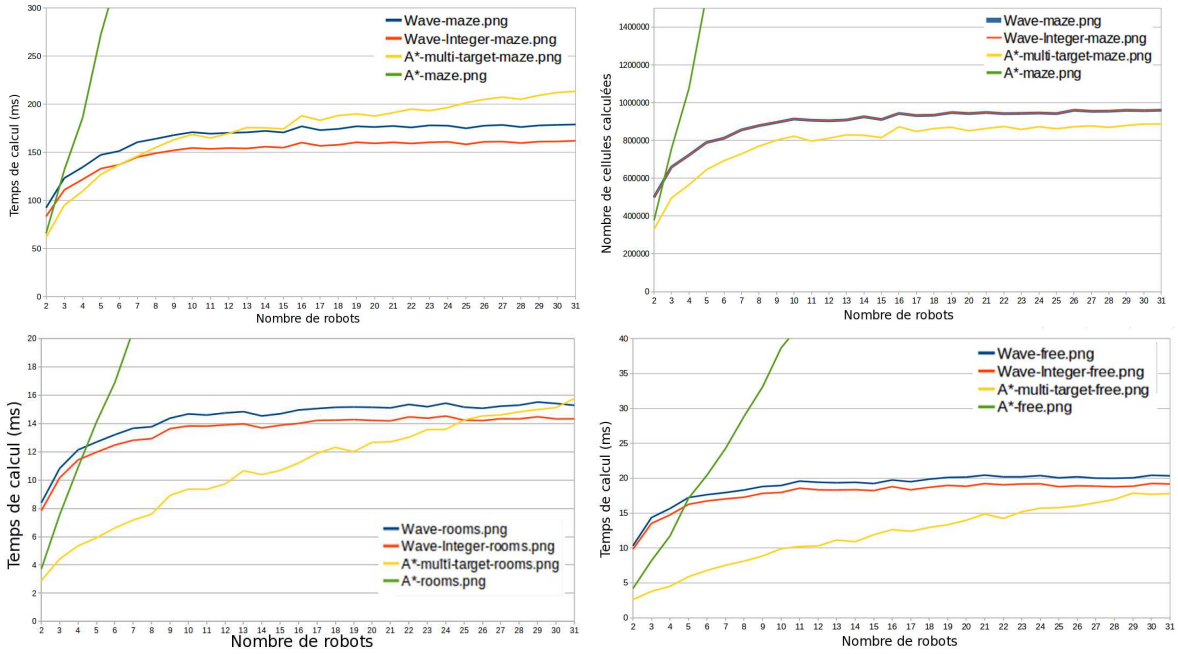


FIGURE 6.9 – Temps de calcul A*, A* multiobjectif et la propagation de vague (en nombre entier et à virgule flottante) pour un nombre croissant de robots sur différents environnements. En haut avec l'environnement labyrinthe : à gauche le temps de calcul, à droite le nombre de cellules où la distance à la frontière a été calculée.
En bas : à gauche l'environnement chambre, à droite un environnement sans obstacle.

Nous pouvons observer que l'exécution d'un A* en direction de chaque frontière est plus performant quand le nombre de robots est faible (entre 3 et 5 suivant les environnements). En revanche, dès que le nombre de robots est plus important, le calcul de la propagation de vague est plus performant et devient de plus en plus performant. Le A* multiobjectif calcule toujours moins de cellules que la vague, mais le calcul de l'heuristique fait qu'à partir d'un certain nombre de robots (dépendant de l'environnement) la vague est moins coûteuse en temps de calcul.

6.2 *MinPos* : version parallélisée

Le calcul des vagues permet de calculer la matrice de coûts et d'utiliser les algorithmes *MinPos* et *MPG* présentés au chapitre 4. La propagation des vagues est l'étape la plus coûteuse en temps de calcul de l'affectation d'une frontière au robot. Nous avons vu en section 6.1.3 qu'il n'était pas nécessaire de propager toutes les vagues sur la totalité de l'environnement mais uniquement jusqu'à ce qu'elles atteignent le robot calculant son affectation. Nous proposons ici d'intégrer le calcul d'affectation à la propagation de vague afin de réduire davantage les distances sur lesquelles les vagues sont propagées. Les vagues sont propagées sur les distances minimum nécessaires à l'affectation du robot calculant son affectation. Pour cela, seules les vagues répondant aux critères fixés par l'algorithme d'affectation sont propagées.

Avant de donner le détail de l'algorithme pour *MinPos*, nous prenons l'exemple de *MinDist* pour donner une intuition du fonctionnement des algorithmes présentés dans la suite de ce chapitre. Pour rappel, *MinDist* affecte le robot à la frontière la plus proche. Pour trouver cette frontière, les vagues lancées depuis chaque frontière peuvent être propagées jusqu'au robot pour déterminer sa distance à chaque frontière. Ensuite en comparant les distances, nous pouvons déterminer la plus proche. Une approche plus efficace consiste à propager les vagues simultanément (en parallèle), si elles se propagent à la même vitesse (en les synchronisant sur la distance de propagation). La première vague a atteindre le robot calculant son affectation est celle provenant de la frontière la plus proche. Les propagations peuvent alors toutes être arrêtées car le robot a déterminé son affectation. Toutes les vagues n'ont alors pas été propagées jusqu'au robot, économisant ainsi du temps de calcul.

Nous appliquons le même principe à l'algorithme *MinPos* en synchronisant les vagues, non pas uniquement sur la distance de propagation, mais sur le nombre de robot rencontré par la vague (calculant ainsi la position du robot).

En parenthèse, notons que notre implémentation de *MinDist* propage une seule vague depuis le robot qui est arrêtée dès qu'elle atteint une frontière (la frontière la plus proche du robot).

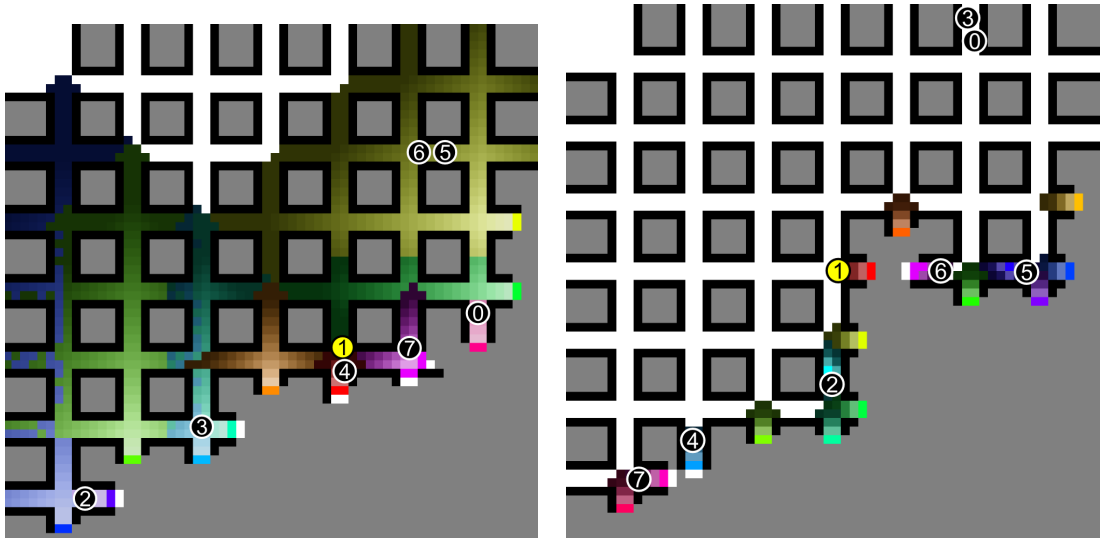
6.2.1 Algorithme

Pour l'affectation de *MinPos*, il n'est pas nécessaire de propager toutes les vagues jusqu'au robot calculant son affectation. Nous avons développé une approche de calcul parallèle de la propagation des différents fronts de vagues. Pour cela, les fronts de vagues sont démarrés simultanément depuis chaque frontière et se propagent à la même vitesse. Lorsqu'un front atteint une cellule occupée par un robot, deux cas sont considérés :

- si le robot rencontré est le robot calculant son affectation alors le calcul est terminé. Le robot est assigné à la frontière correspondant à la vague.
- si c'est un autre robot (que le robot calculant son affectation), la propagation de ce front est mise en pause. Si tous les fronts de vagues sont en pause, il se peut pour autant que le robot cherchant son affectation n'ait pas été atteint par un front. Tous les fronts sont alors redémarrés séquentiellement en commençant par le front ayant la valeur de potentiel la moins élevée (le front de vague s'étant le moins propagé). Ceci, afin que les distances parcourues par les fronts soient égales et que le premier front touchant le robot calculant son affectation soit bien le front associé à la frontière la plus proche pour laquelle le robot à un rang minimum.

Ce processus permet d'avoir une synchronisation des fronts de vagues d'abord sur le nombre de robots rencontrés, puis sur la distance aux frontières.

La figure 6.10 illustre le calcul des fronts de vagues en parallèle (6.10b) par rapport au calcul séquentiel de tous les fronts de vagues jusqu'au robot calculant son assignation (6.10a). Les vagues sont représentées par des couleurs différentes liées à la couleur de la frontière associée en faisant varier sa luminosité en fonction de la valeur de son potentiel. Il apparaît clairement que les distances sur lesquelles les vagues ont été propagées sont inférieures avec la propagation des vagues en parallèle.



(a) Vagues calculées avec l'arrêt des vagues sur le robot calculant son affectation

(b) Vagues calculées avec la version parallélisée

FIGURE 6.10 – Comparaison de l'implémentation de *MinPos* avec l'arrêt des vagues sur le robot qui calcule son assignation et la version parallélisée.

Par rapport à l'algorithme classique de la propagation de vague, nous avons ajouté une information dans la vague correspondant au nombre de robots qu'elle rencontre lors de sa propagation. Formellement, l'algorithme 6 consiste à créer l'ensemble des fronts de vagues $\mathcal{MRW}$ (*Min Rank Wavefronts*) ayant rencontrés le minimum de robots. Ensuite, les vagues ayant le potentiel le moins élevé $\mathcal{MRMDW}$ (*Min Rank Min Distance Wavefronts*) parmi les vagues de $\mathcal{MRW}$ sont propagées. Cette opération est répétée jusqu'à ce que le robot calculant son affectation soit rencontré par un front de vague ou que tous les fronts de vagues ne puissent plus être propagés (la totalité des cellules atteignables depuis la source a été parcourue). Dans ce dernier cas, le robot n'a pas d'affectation possible, toutes les frontières étant inaccessibles pour lui, son exploration est terminée.

Afin de synchroniser les vagues sur le potentiel atteint, chaque vague (répondant au critère de rang) est, séquentiellement, propagée d'un pas. La taille de ce pas est choisie en fonction de la précision d'affectation voulue (voir figure 6.11). La fonction *Propagate* utilisée dans les algorithmes proposés propage la vague d'un pas.

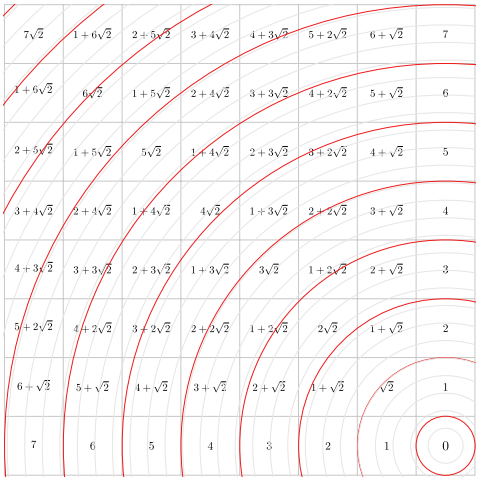


FIGURE 6.11 – Étapes de propagation nécessaires pour la propagation des vagues en parallèle (fonction *Propagate*)

Notations :

$\mathcal{W}$ l'ensemble des fronts de vagues qui sont propagés,
$\mathcal{W}_0^j$ l'ensemble des cellules frontière appartenant à la frontière $\mathcal{F}_j$,
$\mathcal{W}_d^j$ l'ensemble des cellules du front de vague associé à la frontière $\mathcal{F}_j$ à la distance d,
Rank$(\mathcal{W}^j)$ le nombre de robots rencontrés par la vague liée à la frontière j après la dernière étape de propagation de ce front,
Pot$(\mathcal{W}^j)$ le potentiel atteint après la dernière étape de propagation de ce front par la vague associée à la frontière $\mathcal{F}_j$,
Cell$(\mathcal{R}_i)$ la cellule occupée par le robot i.
Assign$(\mathcal{R}_i)$ l'affectation du robot i, **Assign**$(\mathcal{R}_i) \in \{\varnothing \cup \mathcal{F}\}$,

Algorithme 6: Algorithme *MinPos* avec propagation des fronts de vagues en parallèle

Données : $\mathcal{F}$, $\mathcal{R}$, $\mathcal{CG}_{rid}$
Résultat : Affectation du robot $\mathcal{R}_a$

$\mathcal{W} = \mathcal{F}$
Tant que Assign$(\mathcal{R}_a) = \varnothing$ **et** $\mathcal{W} \neq \varnothing$ **faire**
  $\mathcal{MRW}$=$\{\mathcal{W}^j \mid \mathcal{W}^j \in \mathcal{W}$ et **Rank**$(\mathcal{W}^j) = \underset{\mathcal{W}^k \in \mathcal{W}}{\text{argmin}}$ **Rank**$(\mathcal{W}^k)\}$
  $\mathcal{MRMDW}$=$\{\mathcal{W}^j \mid \mathcal{W}^j \in \mathcal{MRW}$ et **Pot**$(\mathcal{W}^j) = \underset{\mathcal{W}^k \in \mathcal{MRW}}{\text{argmin}}$ **Pot**$(\mathcal{W}^k)\}$
  Pour tous les $\mathcal{W}^j \in \mathcal{MRMDW}$ **faire**
    $d =$ **Pot**$(\mathcal{W}^j)$
    $\mathcal{W}_{d+1}^j = Propagate(\mathcal{W}_d^j)$
    Si Cell$(\mathcal{R}_a) \in \mathcal{W}_{d+1}^j$ **Alors Assign**$(\mathcal{R}_a) = \mathcal{F}_j$ et Terminer
    Si $\exists\ \mathcal{R}_i \in \mathcal{R} \mid$ **Cell**$(\mathcal{R}_i) \in \mathcal{W}_{d+1}^j$ **Alors Rank**$(\mathcal{W}^j) =$ **Rank**$(\mathcal{W}^j) + 1$
    Si $\mathcal{W}_{d+1}^j = \varnothing$ **Alors** retirer $\mathcal{W}^j$ de $\mathcal{W}$
  fin
fin

6.2.2 *MinPos* localisé

L'efficacité de l'algorithme de propagation des fronts de vagues en parallèle est lié au nombre de pas de propagation de l'ensemble des fronts (nombre d'exécutions de la fonction *propagate*) et à la largeur de ce front (nombre de cellules appartenant au front). La largeur du front n'est pas modifiable mais la propagation des vagues en parallèle permet de limiter le nombre de pas de propagation en ne propageant que les vagues répondant au critère d'affectation. Pour le réduire davantage, nous pouvons limiter le nombre de vagues qui sont propagées.

Tous les fronts de l'environnement sont propagés en parallèle mais la plupart étant proches d'un robot, ils sont rapidement mis en pause, économisant ainsi le calcul de la propagation de ces fronts. Le robot calculant son affectation est aussi, en général, proche d'une frontière. La frontière à laquelle il sera affecté est, en général, proche en termes de chemin mais aussi en termes de distance euclidienne. L'algorithme avec propagation des vagues en parallèle garantit que le premier front de vague à toucher le robot est le front de vague associé à la frontière à laquelle le robot sera affecté. En effet, les vagues propagées sont les seules à répondre aux critères des minima de position et de distance. Or la distance euclidienne est inférieure ou égale à la distance de propagation (dû à l'utilisation d'une grille et à la prise en compte d'obstacles). Une vague ayant été propagée à une distance inférieure à la distance euclidienne séparant la frontière du robot ne pourra donc jamais rencontrer un robot. Nous pouvons donc facilement borner l'ensemble des vagues qui peuvent être propagées en utilisant la distance euclidienne de la cible au robot.

Nous posons une variante de l'algorithme 6 *MinPos* parallélisé, nommé *MinPos localisé*, qui limite le nombre de vagues propagées en s'appuyant sur le critère de distance euclidienne à la frontière. L'algorithme de propagation parallélisée des vagues reste identique mais les vagues sont lancées séquentiellement selon leur distance euclidienne au robot et propagées jusqu'à ce que la vague suivante (en distance euclidienne) ait une possibilité d'atteindre le robot c'est-à-dire que la longueur de propagation des vagues soit supérieure ou égale à la distance euclidienne entre le robot et la cible associée à la vague suivante.

La figure 6.12 illustre la propagation des vagues avec l'algorithme *MinPos localisé* implémenté avec la propagation des vagues en parallèle. L'environnement contient trois frontières et un robot au centre (en rouge). La vague verte (en bas à gauche de chaque image), la plus proche en distance euclidienne, se propage jusqu'à ce qu'elle se soit propagée d'une distance supérieure à la distance euclidienne séparant le robot des frontières brune et rouge (en haut à droite des images). À ce moment, les vagues brune et rouge sont propagées jusqu'à ce qu'elles aient parcourues la même distance que la vague verte (pour "rattraper leur retard de propagation" afin qu'elles soient synchronisées). Les trois vagues sont ensuite propagées de manière synchronisée jusqu'à ce que la vague la plus proche du robot (en distance de chemin) atteigne le robot.

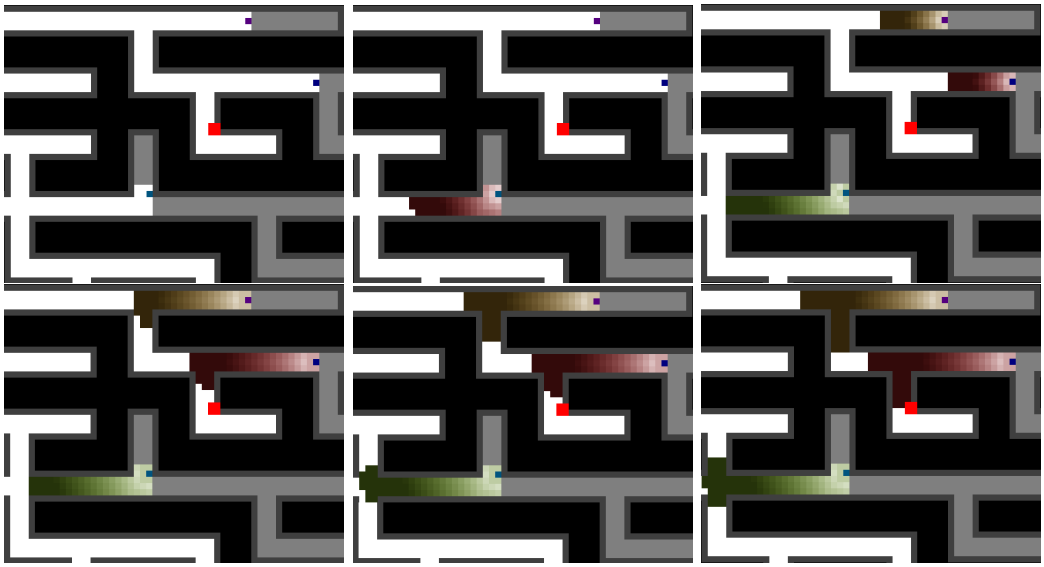
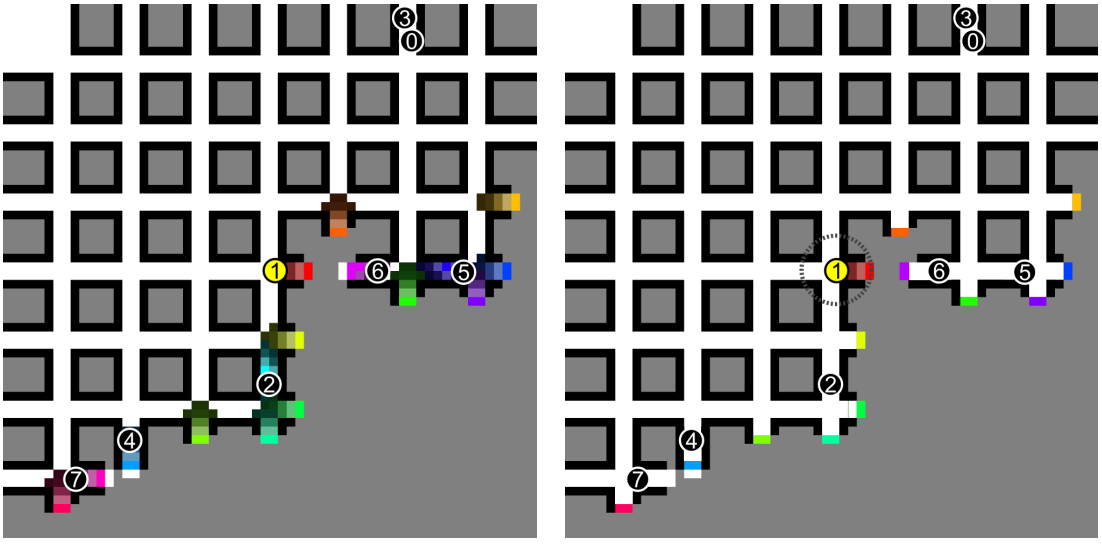


FIGURE 6.12 – Propagation des vagues en parallèle, trois frontières et un robot (en rouge)

La figure 6.13 ci-dessous illustre le calcul *MinPos* parallélisé (6.13a) par rapport au calcul de *MinPos* localisé (6.13b). Les vagues sont représentées par des couleurs différentes liées à la couleur de la frontière associée en faisant varier sa luminosité en fonction de la valeur de son potentiel. Sur cet exemple, une seule vague a été propagée avec *MinPos* localisé tandis qu'avec *MinPos* parallèle toutes les vagues sont propagées.



(a) Vagues calculées avec la version parallélisée

(b) Vagues calculées avec la version parallélisée et localisée

FIGURE 6.13 – Comparaison de l'implémentation de *MinPos* version parallélisée et version localisée.

6.3 Algorithme Glouton : version parallélisée

Nous avons vu, au chapitre 4, que l'affectation gloutonne des frontières garantit une répartition équilibrée des robots sur les cibles en affectant le couple robot-frontière ayant le coût le plus bas à chaque itération. Lorsque le nombre de robots est supérieur au nombre de cibles, plusieurs robots sont affectés à la même cible. Nous proposons ici une implémentation de l'algorithme Glouton utilisant la propagation des vagues en parallèle. Cette implémentation est efficace car elle ne nécessite pas de calculer la matrice de coût.

L'algorithme 7 donne l'algorithme Glouton utilisant la propagation des vagues en parallèle. La différence avec l'algorithme *MinPos* est que, au lieu de construire un ensemble $\mathcal{MRW}$ contenant les vagues ayant rencontré le minimum de robots, un ensemble $\mathcal{MAW}$ (MinAssigned Wavefront) est construit avec les vagues ayant le minimum de robots affectés (affectés à la frontière associée à la vague). Les vagues sont mises en pause quand elles rencontrent un robot n'ayant pas d'affectation. Lors de cette rencontre, le statut du robot change de non-affecté à affecté. Les vagues ont donc un compteur du nombre de robots non-affectés rencontrés qui est incrémenté lorsqu'elles rencontrent un robot *non-affecté*. Elles sont synchronisées sur ce compteur et sur la distance de propagation. Dans l'algorithme *MinPos* parallélisé le statut des autres robots ne change pas quand un robot est atteint par une vague car leur critère de propagation est nombre de robots rencontrés. Ici, une vague doit pouvoir connaître le statut du robot rencontré afin d'incrémenter son compteur quand elle rencontre un robot *non-affecté*.

En plus des notations précédemment définies nous notons :
$\mathbf{A}(\mathcal{W}^j)$ le nombre de robots non assignés rencontré par la vague liée à la frontière j après la dernière étape de propagation de ce front.

Algorithme 7: Algorithme Glouton avec propagation des fronts de vagues en parallèle

Données : $\mathcal{F}$, $\mathcal{R}$, $\mathcal{CG}_{rid}$
Résultat : Affectation du robot $\mathcal{R}_a$

$\mathcal{W} = \mathcal{F}$
Tant que $\mathbf{Assign}(\mathcal{R}_a) = \varnothing$ **et** $\mathcal{W} \neq \varnothing$ **faire**
  $\mathcal{MAW}$=$\{\mathcal{W}^j \mid \mathcal{W}^j \in \mathcal{W}$ et $\mathbf{Assigned}(\mathcal{W}^j) = \underset{\mathcal{W}^k \in \mathcal{W}}{\text{argmin}}\ \mathbf{Assigned}(\mathcal{W}^k)\}$
  $\mathcal{MAMDW}$=$\{\mathcal{W}^j \mid \mathcal{W}^j \in \mathcal{MRW}$ et $\mathbf{Pot}(\mathcal{W}^j) = \underset{\mathcal{W}^k \in \mathcal{MRA}}{\text{argmin}}\ \mathbf{Pot}(\mathcal{W}^k)\}$
  Pour tous les $\mathcal{W}^j \in \mathcal{MAMDW}$ **faire**
    $d = \mathbf{Pot}(\mathcal{W}^j)$
    $\mathcal{W}^j_{d+1} = Propagate(\mathcal{W}^j_d)$
    Si $\mathbf{Cell}(\mathcal{R}_a) \in \mathcal{W}^j_{d+1}$ **Alors** $\mathbf{Assign}(\mathcal{R}_a) = \mathcal{F}_j$ et Terminer
    Si $\exists\ \mathcal{R}_i \in \mathcal{R} \mid \mathbf{Cell}(\mathcal{R}_i) \in \mathcal{W}^j_{d+1}$ **Et** $\mathbf{Assign}(\mathcal{R}_i) = \varnothing$
     Alors $\mathbf{Assigned}(\mathcal{W}^j) = \mathbf{Assigned}(\mathcal{W}^j) + 1$ **Et** $\mathbf{Assign}(\mathcal{R}_i) = \mathcal{F}_j$
    Si $\mathcal{W}^j_{d+1} = \varnothing$ **Alors** retirer $\mathcal{W}^j$ de $\mathcal{W}$
  fin
fin

Cet algorithme Glouton, grâce à l'utilisation de la propagation des vagues en parallèle, ne nécessite pas de calculer la matrice de coûts et est ainsi plus efficace.

6.4 Algorithme *MPG* (MinPosGlouton) : version parallélisée

Nous présentons maintenant l'algorithme de la conjugaison des approches *MinPos* et gloutonne (*MPG*) avec la propagation des vagues en parallèle. Cette approche intègre la construction de l'ensemble intermédiaire $\mathcal{MAMRW}$ qui contient les vagues ayant le moins de robots assignés et ayant rencontré le minimum de robots. Seules les vagues de cet ensemble sont propagées. Les avantages de cet algorithme ont été présentés en section 4.5. L'algorithme 8 présente l'algorithme *MPG* utilisant la propagation des vagues en parallèle. Cette implémentation de l'algorithme *MPG*, est plus performant car tout comme les implémentations avec la propagation des vagues en parallèle, elle ne calcule pas la totalité de la matrice de coût. De plus, contrairement à l'algorithme donné en section 4, il ne nécessite pas de calculer la position de chaque robot vers chaque frontière.

Algorithme 8: Algorithme *MPG* avec propagation des fronts de vagues en parallèle

Données : $\mathcal{F}$, $\mathcal{R}$, $\mathcal{CG}_{rid}$
Résultat : Affectation du robot $\mathcal{R}_a$

$\mathcal{W} = \mathcal{F}$
Tant que **Assign**$(\mathcal{R}_a) = \varnothing$ **et** $\mathcal{W} \neq \varnothing$ **faire**
  $\mathcal{MAW}$=$\{\mathcal{W}^j \mid \mathcal{W}^j \in \mathcal{W}$ et **Assigned**$(\mathcal{W}^j) = \underset{\mathcal{W}^k \in \mathcal{W}}{\text{argmin}}$ **Assigned**$(\mathcal{W}^k)\}$
  $\mathcal{MAMRW}$=$\{\mathcal{W}^j \mid \mathcal{W}^j \in \mathcal{MAW}$ et **Rank**$(\mathcal{W}^j) = \underset{\mathcal{W}^k \in \mathcal{MAW}}{\text{argmin}}$ **Rank**$(\mathcal{W}^k)\}$
  $\mathcal{MAMRMDW}$=$\{\mathcal{W}^j \mid \mathcal{W}^j \in \mathcal{MAMRW}$ et **Pot**$(\mathcal{W}^j) = \underset{\mathcal{W}^k \in \mathcal{MAMRW}}{\text{argmin}}$ **Pot**$(\mathcal{W}^k)\}$
  Pour tous les $\mathcal{W}^j \in \mathcal{MAMRMDW}$ **faire**
    $d =$ **Pot**$(\mathcal{W}^j)$
    $\mathcal{W}^j_{d+1} = Propagate(\mathcal{W}^j_d)$
    Si Cell$(\mathcal{R}_a) \in \mathcal{W}^j_{d+1}$ **Alors Assign**$(\mathcal{R}_a) = \mathcal{F}_j$ et Terminer
    Si $\exists\ \mathcal{R}_i \in \mathcal{R} \mid$ **Cell**$(\mathcal{R}_i) \in \mathcal{W}^j_{d+1}$ **Et Assign**$(\mathcal{R}_i) = \varnothing$
     Alors Assigned$(\mathcal{W}^j) =$ **Assigned**$(\mathcal{W}^j) + 1$ **Et Assign**$(\mathcal{R}_i) = \mathcal{F}_j$
    Si $\exists\ \mathcal{R}_i \in \mathcal{R} \mid$ **Cell**$(\mathcal{R}_i) \in \mathcal{W}^j_{d+1}$ **Alors Rank**$(\mathcal{W}^j) =$ **Rank**$(\mathcal{W}^j) + 1$
    Si $\mathcal{W}^j_{d+1} = \varnothing$ **Alors** retirer $\mathcal{W}^j$ de $\mathcal{W}$
  fin
fin

La complexité de l'algorithme *MPG* utilisant la matrice de coût (algorithme 4) est du même ordre que Glouton ($O(n^2m)$). Il est toutefois plus complexe car il nécessite de pré-calculer les positions des robots vers chaque frontière. L'implémentation de *MPG*, utilisant la propagation des vagues en parallèle, rend *MPG* aussi efficace que les algorithmes *MinPos* et Glouton. En effet, son implémentation naïve consiste uniquement à ajouter à l'algorithme 6 de *MinPos*, la recherche des vagues ayant le minimum de robots assignés pour les propager. Néanmoins, nous utilisons une file de priorité fondée sur un tas binaire pour stocker les vagues. Cette file est ordonnée selon les critères[6] utilisés par l'algorithme pour décider quelle vague doit être propagée. Les implémentations des algorithmes données ici sont alors très similaire ; seules les fonctions de comparaison des vagues sont différentes. La vague ayant une priorité maximum est retirée de la file, propagée puis insérée dans la file.

6. nombre de robots assignés, nombre de robots rencontrés, distance de propagation

Ces implémentations permettent de calculer l'affectation d'un robot sans calculer la matrice de coût. Ainsi, elles sont très efficaces en comparaison des approches nécessitant de la calculer (impliquant le calcul de la distance de chaque robot vers chaque frontière). Notons que la synchronisation des différentes propagations est suffisante pour déterminer la frontière à laquelle un robot sera affecté. Ces implémentations ne nécessitent donc pas réellement de calculer la distance.

Le principe utilisé pour la version localisée de *MinPos* (voir section 6.2.2) peut être aussi utilisé pour les algorithmes Glouton et *MPG* ce qui les rendrait encore plus efficace. Avec cette version, lorsque le robot calculant son affectation est proche d'une frontière, son affectation est très rapide. Lorsque des robots sont plus proches que lui, le temps de calcul de son affectation est plus long car elle nécessite de propager plus de vagues sur des plus grandes distances.

Les implémentations des algorithmes *MinPos*, Glouton et *MPG* proposées dans ce chapitre sont efficaces pour un système décentralisé (chaque robot calcule son affectation). Avec un système centralisé, il est plus efficace de propager les vagues jusqu'à ce que tous les robots soient atteints.

La dernière partie de la thèse présente l'intégration robotique des algorithmes proposés et les expérimentations avec des robots réels. Nous commençons dans le chapitre suivant par étudier l'identification des frontières d'où sont propagées les vagues afin de pouvoir identifier les frontières avec des capteurs réels.

Troisième partie

Intégration robotique et expérimentations

Chapitre 7

Identification et stratégies d'observation des frontières

Sommaire

Les chapitres précédents se sont concentrés sur l'affectation des cibles - i.e. des frontières - aux robots en supposant que celles-ci étaient bien identifiées. Dans ce chapitre, nous décrivons dans un premier temps les méthodes d'identification des cellules constituant les frontières et nous montrons comment les regrouper pour réduire les coûts de calcul de l'affectation et aussi rendre l'exploration plus efficace. Dans un second temps, nous adressons le problème de l'observation des frontières en considérant différents types de capteurs.

Notre proposition de regroupement des cellules frontières est inspiré du travail de Dornhege and Kleiner (2011). Leur travail s'applique à l'exploration monorobot en trois dimensions. Nous l'avons appliqué au multirobot en 2D. Le résultat de son application est similaire à l'approche Utilité (Burgard et al., 2005) car elle pénalise l'affectation de plusieurs robots à des cellules frontières proches. Nous étudions à la fin de cette première partie leurs différences.

7.1 Identification des frontières

Dans notre approche multirobot de l'exploration d'un environnement inconnu, chaque robot se dirige vers une cible donnée. Ces cibles sont des lieux à atteindre permettant d'accroître leur connaissance de l'environnement. Comme cela est communément fait, en monorobot ainsi qu'en multirobot, nous guidons le ou les robots vers les limites entre les zones explorées et inexplorées de l'environnement. Ainsi, nous utilisons comme cibles les parties sans obstacle de ces limites qui sont les **frontières**. Nous rappelons que nous utilisons une carte discrète, sous forme de grille constituée de cellules, pour représenter l'environnement.

Afin d'identifier les frontières, nous utilisons plusieurs représentations discrètes stockées sous forme de grilles dont les cellules contiennent des informations complémentaires.

- La **grille d'occupation** est une grille dont chaque cellule contient la probabilité que cette cellule soit occupée par un obstacle. Elle est maintenue à jour par l'algorithme de localisation et cartographie du robot.
- La **grille d'exploration** maintient le statut d'exploration des cellules. Ainsi, chaque cellule contient une valeur booléenne : vrai, si elle a été observée et faux sinon. Elle est notée $\mathcal{CE}xplo$ et est en deux dimensions, (x,y) les coordonnées dans la grille.
- La **grille de configuration** stocke les configurations du robot qui ne sont pas en collision avec un obstacle. Elle est notée $\mathcal{CG}rid$ en deux ou trois dimensions avec (x, y, θ) les coordonnées dans la grille et l'orientation du robot.

Le principe général de l'identification des frontières est le suivant : à partir de la grille d'occupation, nous calculons une grille de configuration et une grille d'exploration avec une résolution inférieure. Ces dernières sont utilisées pour identifier les cellules frontières qui sont ensuite regroupées pour déterminer les frontières guidant l'exploration. Une **frontière** est définie comme un ensemble de cellules frontières contiguës.

Dans la suite, nous décrivons comment nous avons abordé le problème du calcul de la grille de configuration, permettant d'identifier les cellules frontières, en utilisant la grille d'occupation.

7.1.1 Calcul des grilles de configuration et d'exploration

À partir de la grille d'occupation, fournie en entrée de l'algorithme d'identification des cibles, nous voulons calculer l'espace de configuration du robot pour évaluer rapidement si une configuration est navigable (sans collision). L'espace de configuration fait référence à l'ensemble des configurations du robot qui ne sont pas en collision avec un obstacle. En effet, une cellule non occupée par un obstacle n'est pas forcement navigable, les robots occupant une surface supérieure à une cellule. L'espace de configuration assimile le robot à un point dans un espace à deux dimensions. Il est calculé en deux étapes : le marquage des cellules occupées puis leur dilatation.

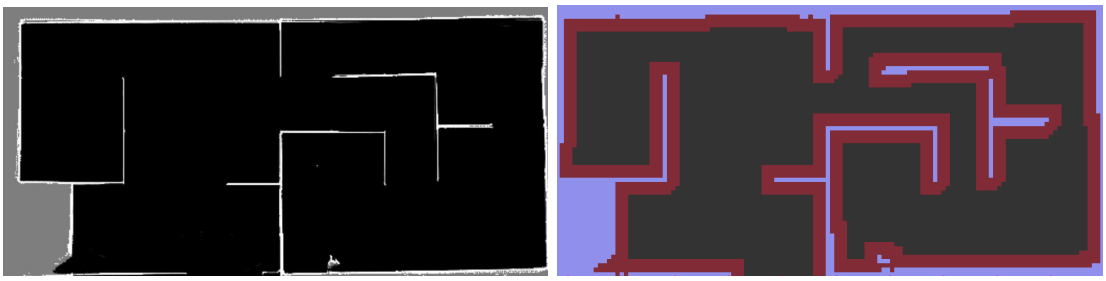
La **première étape** de ce calcul consiste à marquer les cellules occupées par un obstacle. Cette opération est nécessaire car la résolution de la grille de configuration est inférieure à celle de la grille d'occupation (passage de cellule de $2cm^2$ à $8cm^2$). Une cellule des grilles de configuration et d'exploration ($\mathcal{CG}rid$ et $\mathcal{CE}xplo$) comprend donc 16 cellules de la grille d'occupation.

Nous distinguons alors trois cas, suivant les valeurs des cellules de la grille d'occupation :

- Les cellules de $\mathcal{CG}rid$ sont considérées comme des **obstacles** si au moins trois cellules de la grille d'occupation (parmi les 16 correspondantes) sont occupées c'est-à-dire qu'elles ont une probabilité d'occupation supérieure à 0.5. Ce seuil de trois cellules, choisi expérimentalement, offre un bon compromis entre sécurité et filtrage du bruit dans la grille d'occupation.
- De la même manière, les cellules de $\mathcal{CG}rid$ sont considérées comme **vide** si elles ne sont pas marquées comme des obstacles et si au moins trois cellules de la grille d'occupation sont considérées comme vides (probabilité d'occupation inférieur à 0.5).
- Les cellules, qui ne sont pas marquées comme obstacle ou vide, sont marquées comme inexplorées dans la grille d'exploration $\mathcal{CE}xplo$.

Ce passage d'une grille d'occupation probabiliste à une grille d'occupation déterministe avec une réduction de résolution permet de filtrer le bruit présent dans la grille d'occupation probabiliste mais ne permet pas de prendre en considération les obstacles ayant une surface inférieure à $6cm^2$.

Afin de limiter les calculs, nous faisons l'hypothèse que les robots sont circulaires et holonomes. La **seconde étape** du calcul de l'espace de configuration consiste alors simplement à effectuer une dilatation des obstacles du rayon d'un robot. Toutefois, comme nous utilisons un espace discret, une approximation de la taille du robot en nombre de cellules est nécessaire. Nous avons choisi d'arrondir vers la valeur haute afin de garantir que les configurations évaluées soient sûres. La figure 7.1 illustre la grille d'occupation et la grille de configuration correspondante.



(a) Grille d'occupation (b) Grille de configuration

FIGURE 7.1 – Calcul d'une grille de configuration. Les probabilités d'occupation de la carte de SLAM, grille d'occupation avec des cellules de $2cm^2$ (7.1a), sont seuillées et les obstacles dilatés (de trois cellules de $8cm^2$) dans la grille de configuration (7.1b). Dans la grille d'occupation, le niveau de gris illustre la probabilité d'occupation : plus il est clair plus la probabilité est importante. Dans la grille de configuration, le gris montre l'espace de configuration, le bleu et le rouge les cellules interdites le rouge correspondant à la dilatation des obstacles de trois cellules.

Notons que pour lever l'hypothèse de circularité du robot, il faudrait calculer l'espace de configuration en trois dimensions (x,y,orientation) et effectuer une somme de Minkowski [7] entre les obstacles et le robot pour chaque orientation (Lozano-Pérez and Wesley, 1979). Le calcul des vagues devra alors être effectué en trois dimensions (coordonnées 2D et orientation) ce qui est plus coûteux. De plus, pour lever l'hypothèse de robots holonomes, il faudrait modifier la fonction de voisinage utilisée par la propagation de vague qui devra aussi s'effectuer en trois dimensions. Dans ce qui suit, nous poursuivons avec l'hypothèse d'un robot de forme cylindrique.

7.1.2 Identification des cellules frontières

Les frontières sont des points d'intérêt pour découvrir des nouvelles parties de l'environnement. Étant donné que nous utilisons une représentation discrète, les frontières sont constituées de cellules appelées **cellules frontières**. Les cellules frontières sont des cellules de la grille d'observation de type vide [8] et exploré [9] et adjacentes à une cellule inexplorée.
La grille de configuration ($\mathcal{CG}rid$), décrite dans la section précédente, est utilisée pour identifier ces cellules frontières.

En monorobot, il est possible de maintenir une liste de ces cellules frontières et d'en faire une mise à jour incrémentale uniquement à partir des nouvelles zones perçues. En multirobot, cette méthode n'est pas applicable car des cellules frontières peuvent apparaître ou disparaître dans les cartes reçues des autres robots en plus des zones nouvellement perçues par le robot.

7. Équivalent d'une dilatation des obstacles en utilisant la forme du robot comme objet structurant.
8. dans la grille de configuration
9. dans la grille d'exploration

Les cellules frontières sont donc identifiées en parcourant la totalité de la carte résultant de la fusion de la carte du robot et des celles reçues des autres robots. La figure 7.2 illustre les cellules frontières identifiées sur la grille de configuration, précédemment calculée, fusionnée avec la grille d'exploration.

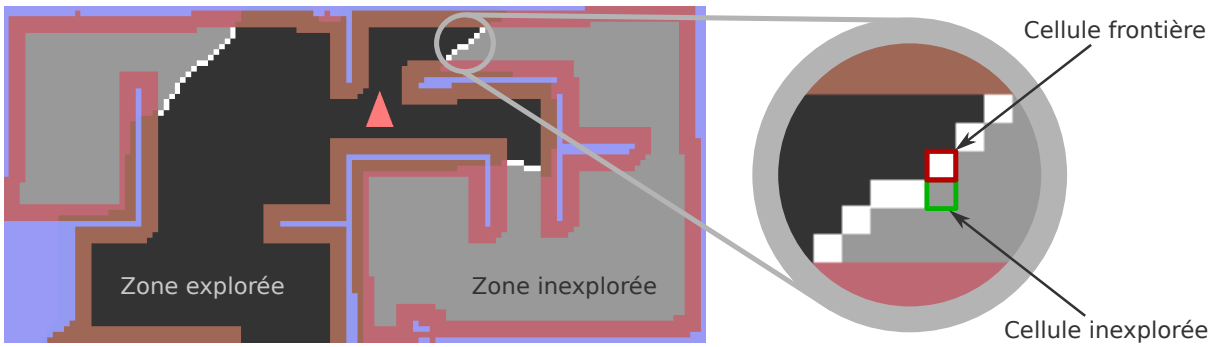


FIGURE 7.2 – Identification des cellules frontières

7.1.3 Identification des frontières : regroupement des cellules

Toutes les cellules frontières pourraient être utilisées pour l'affectation. Toutefois, il faudrait calculer les distances robot-frontière vers chaque cellule frontière. Aussi, avec l'algorithme *MinPos* parallélisé, une vague devrait être propagée depuis chaque cellule. Cette approche est évidemment couteuse en temps de calcul car la propagation de vague est l'étape de la méthode d'affectation la plus coûteuse. Pour réduire ce coût, nous visons à calculer une seule vague par frontière, c'est-à-dire à partir d'une cellule représentant la frontière. Celle-ci doit permettre d'observer, avec une acquisition, toute la frontière.

Les cellules frontières contiguës sont regroupées car elles appartiennent à la même frontière. Un seuil est fixé pour limiter la taille des groupes de cellules formant les frontières. Ce seuil est lié à la taille de la zone perçue par les robots afin de créer des frontières observables avec une seule acquisition.

Cette méthode est d'autant plus intéressante car au lieu d'affecter le robot à la cellule frontière la plus proche pour laquelle le robot est en meilleure position, il sera affecté à la cellule la plus centrale du groupe. Ainsi, la cible du robot est cette cellule représentante du groupe de cellules.

Algorithme de regroupement des cellules frontières

Les cellules frontières sont regroupées en utilisant un **algorithme de remplissage par diffusion**. Les groupes sont composés de cellules frontières formant une composante connexe ayant une taille maximum.

Dans un premier temps, nous marquons toutes les cellules frontières comme non-visitées. Pour créer un groupe, une cellule non-visitée est sélectionnée aléatoirement comme point de départ, elle est ajoutée à ce nouveau groupe et marquée comme visitée.

Dans un second temps, les cellules frontières non-visitées voisines de la dernière cellule sont ajoutées au groupe.

Ce processus est répété récursivement jusqu'à ce que toutes les cellules frontières connexes soient marquées ou que le groupe ait atteint le nombre maximum d'éléments.

Dans un environnement intérieur, les frontières sont le plus souvent dans des couloirs ou dans des pièces dont la largeur est inférieure au rayon de perception. Dans ces cas, il n'est pas nécessaire de limiter la taille des groupes de cellules frontière ; toutes les cellules frontière contiguës peuvent être regroupées. Toutefois, dans des grandes pièces ou en environnement ouvert, il est nécessaire de limiter la taille des groupes pour une meilleure répartition des robots qui doivent être affectés à des frontières différentes s'ils ne sont pas capables de percevoir toute la frontière. Nous limitons donc la taille des groupes de cellules.

La **taille maximum des groupes de cellules frontières** est calculée en fonction du rayon de perception du robot noté R_p. Pour un capteur 360°, le périmètre du disque de perception est le nombre maximum de cellules frontières pouvant être observées. Considérons deux observations consécutives, nous calculons la taille maximum de la frontière observable. Celle-ci correspond à l'arc de cercle de la première observation couvert par le disque de la deuxième observation.

Ce calcul est illustré figure 7.3. Si la fréquence d'observation est élevée, la taille maximale du groupe perçu est égale à la moitié du périmètre du disque de perception. Dans ce cas, nous utilisons $2\pi R_p/2$. Si la fréquence d'observation est faible, i.e. le robot n'observe qu'une fois arrivé sur la frontière, la taille des groupes devra être égale à la longueur de l'arc couvert par le disque de perception centré sur la frontière (voir l'arc rouge sur la figure 7.3). Dans ce cas, nous utilisons $2\pi R_p/3$.

Pour chaque groupe, une seule vague est propagée à partir de la cellule permettant d'observer les cellules du groupe. Pour un groupe cette cellule est identifiée en choisissant la cellule la plus proche du point moyen du groupe[10].

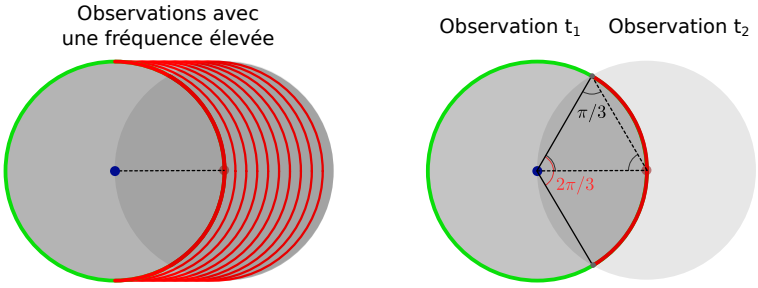


FIGURE 7.3 – En rouge et vert la frontière, le rouge montre le calcul de la taille des groupes des cellules frontières

La taille maximale des groupes de cellules frontières n'affecte que peu les résultats des algorithmes, particulièrement pour les environnements d'intérieur dans lesquels la taille des groupes est limitée par les obstacles (limitant la contiguïté des cellules frontières) : une seule frontière est créée par couloir ou par pièce. La figure 7.4 illustre les groupes de cellules frontières ainsi que les vagues propagées depuis l'ensemble des cellules de chaque groupe.

10. le point dont les coordonnées sont égales à la moyenne des coordonnées x et y des cellules du groupe

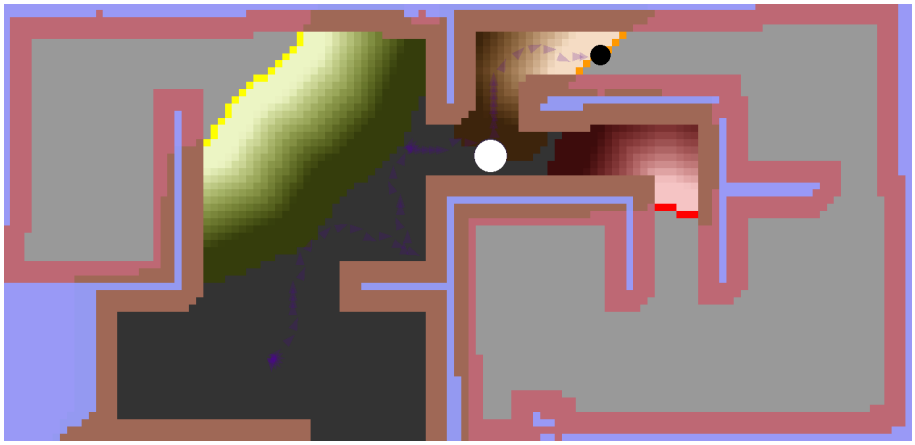


FIGURE 7.4 – Groupement des cellules frontières et calcul de l'affectation : en blanc, le robot, en noir, la cellule cible (représentante du groupe de cellules frontière)

La figure 7.5 présente les performances de l'exploration d'un environnement de type bureau avec cinq robots. Les robots perçoivent à une distance 3,2 mètres, la taille des cellules est $0.08m^2$. Nous pouvons observer que l'algorithme Glouton (*Greedy*) est bien moins performant lorsque la taille des frontière est petite, ceci jusqu'à ce qu'il atteigne le rayon de perception (groupe de 40 cellules). En revanche les algorithmes *MinPos* et *MPG* ne sont pas affectés par la taille des groupes de cellules frontières. Ceci est dû au fait qu'ils considèrent prioritairement le nombre de robots dans chaque direction par rapport au nombre de frontières.

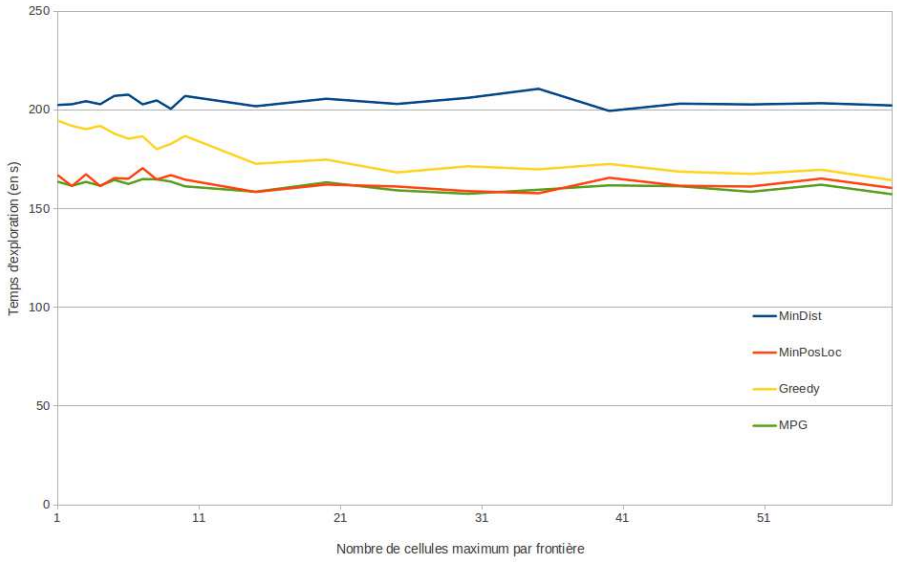


FIGURE 7.5 – Impact de la taille des groupes sur les performances des algorithmes *MinDist*, *MinPos*(MinPosLoc), Glouton (Greedy) et *MPG*sur l'environnement bureau avec 5 robots

Différence avec l'approche Utilité

L'approche Utilité est proche de notre méthode de regroupement de frontière, nous détaillons dans cette section les principales différences.

Quand le nombre de frontières est supérieur ou égal au nombre de robots, la principale différence de notre approche avec l'approche Utilité Burgard et al. (2005) (cf. section 2.5.3) vient du fait que nous prenons en compte la visibilité entre les cellules frontières au lieu de ne considérer que leur contiguïté. En effet, l'approche Utilité diminue l'intérêt d'observer les cellules frontières visibles depuis la cellule frontière affectée. Dans la plupart des situations, le groupement des cellules contiguës est équivalent car il interdit l'affectation de plusieurs robots aux cellules frontières d'un même groupe. Toutefois, pour être complètement équivalent, il faudrait étendre la méthode pour grouper les cellules frontières, non plus contiguës, mais par groupes de cellules frontières visibles entre elles, c'est-à-dire pour lesquelles il existe un segment sans obstacle de distance inférieure au rayon de perception. Les groupes seraient alors composés de cellules qui peuvent être explorées en une seule observation (par le même capteur en une acquisition).

Quand le nombre de robots est supérieur au nombre de frontières, l'approche utilité permet d'affecter les robots à des cellules frontières différentes de la même frontière. Ceci permet une répartition équilibrée des robots sur les grandes frontières, car cette approche maximise l'inter-distance entre les cellules frontières affectées. En revanche l'intérêt est limité dans le cas des environnements structurés car il n'existe pas de frontière suffisamment grande pour que les robots se répartissent le long d'une frontière.

L'affectation des groupes de frontières aux robots fonctionne très bien pour l'exploration avec des capteurs ayant un grand angle de perception et une fréquence d'acquisition élevée (de type télémètre laser, LIDAR). Toutefois pour apporter de la généricité, nous avons étendu notre approche afin de considérer des capteurs différents. La section suivante décrit cette extension.

7.2 Considération des caractéristiques des capteurs

Dans les parties précédentes de cette thèse, nous avons supposé que se diriger vers une frontière permettrait de l'observer. Ceci n'est pas toujours vrai, par exemple, si le robot est équipé d'un capteur avec un angle de vue limité et que le capteur est dirigé vers l'avant du robot alors se diriger vers la frontière en marche arrière n'apportera pas de nouvelles informations. La plupart du temps, les robots se dirigeant vers les frontières pourront observer la frontière mais cela dépend de la forme observée par le capteur, de sa direction et de la fréquence d'observation. Dans cette section nous levons l'hypothèse que toutes les frontières sont accessibles et observables.

Pour réaliser l'observation des frontières avec des capteurs ayant des zones de perception de formes quelconques, il faut prendre en compte ces formes, c'est-à-dire l'intervalle de distance où le capteur perçoit et l'angle du champ de vision.

Dans cette section, nous réexaminons notre approche en considérant ces contraintes de perception.

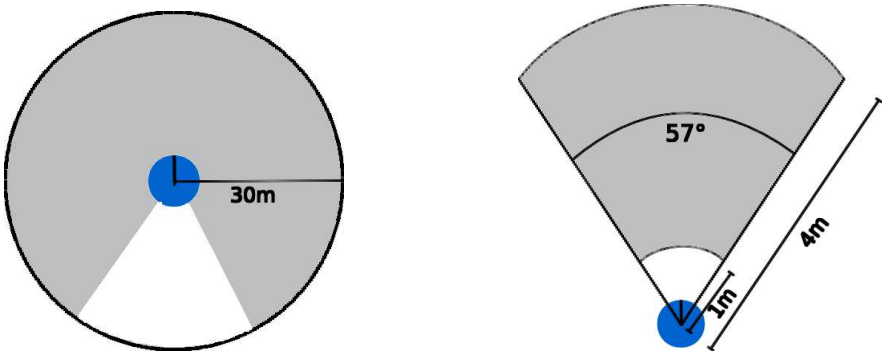
7.2.1 Stratégie d'observation avec un capteur 3D (Kinect)

Pour la construction d'une carte en trois dimensions (3D) et l'évitement des obstacles qui ne sont pas à la hauteur du LIDAR, il est nécessaire d'utiliser un capteur 3D. Nous utilisons l'exemple du capteur 3D Kinect qui a un champ de perception particulier.

En général, l'installation et la nature des capteurs LIDAR 2D leur permettent d'observer sur un plan horizontal avec un angle important (supérieur à 180°). Leurs portées sont très variables

mais la distance minimale de perception est en général très faible (inférieur à 10cm). Comme le montre la figure 7.6b, le capteur 3D Kinect a un champ de vision bien inférieur aux capteurs LIDAR et une distance minimale de perception de 1 mètre.

Pour utiliser la stratégie des frontières définie pour un LIDAR avec un Kinect, nous effectuons huit acquisitions Kinect sur place, couvrant ainsi 360° avec une certaine redondance. L'image de la Kinect étant déformée, particulièrement sur les côtés, nous n'utilisons pas toutes la largeur de l'image i.e. le champ de vision, réellement utilisé est inférieur à 57° horizontalement, il est plus proche de 45°. La redondance est donc limitée. De cette manière, le capteur Kinect est assimilable à un capteur 360°.



(a) Zone perçue par un capteur LIDAR d'une portée de 30m à 270°

(b) Zone perçue par un capteur 3D Kinect (projection verticale en 2D)

FIGURE 7.6 – Capteurs 2D LIDAR (a) et 3D Kinect (b))

Les robots s'arrêtent régulièrement (tous les 1,5m le long de la trajectoire planifiée) pour calculer leur affectation et ainsi mieux coopérer (voir le chapitre 5). Durant ces arrêts les robots effectuent des acquisitions 360°. Rappelons que, lors de ces arrêts, les robots communiquent avec l'ensemble de la flottille pour diffuser leurs cartes. Pour éviter que plusieurs robots effectuent des acquisitions Kinect très proches les unes des autres, nous avons ajouté dans cette communication les positions où les acquisitions Kinect ont été effectuées. Ainsi à chaque arrêt, les robots n'effectuent des observations Kinect que si la distance minimale aux positions des autres acquisitions Kinect (effectuées par l'ensemble de la flottille de robots) est supérieure à 1,5m.

La figure 7.7 illustre les positions des acquisitions Kinect lors d'une exploration avec deux robots d'un environnement de test de $35m^2$ construit à Nancy. Nous observons que cette méthode couvre bien l'espace avec les acquisitions Kinect.

Toutefois de nombreuses acquisitions sont inutiles car elles sont dirigées vers un mur dont le robot est trop proche pour laisser le recul nécessaire à l'observation avec le capteur Kinect. Nous proposons dans la suite une approche différente pour l'exploration avec des capteurs ayant des champs de perception quelconques.

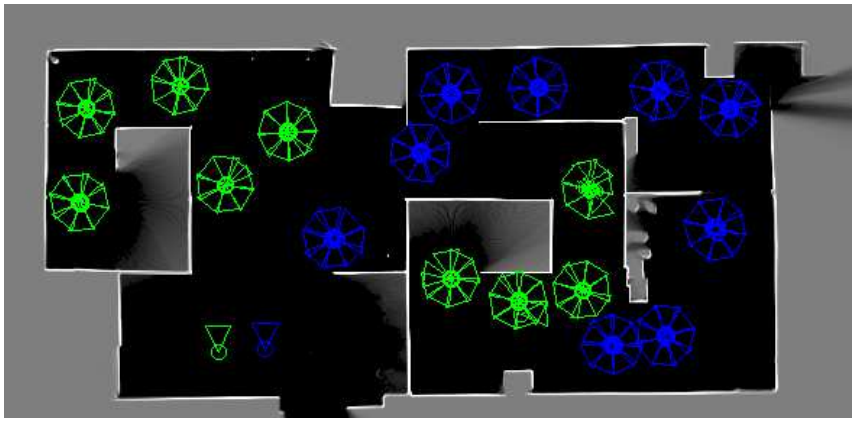


FIGURE 7.7 – Exploration Kinect avec la méthode 360 degrés avec deux robots.

7.2.2 Approche par configuration d'observation

Sur une carte sous forme de grille, l'identification des frontières consiste à identifier les cellules explorées et inoccupées qui sont voisines d'au moins une cellule inexplorée (non obstacle). Une alternative à cette méthode est de maximiser le gain d'information (par exemple (Butzke and Likhachev, 2011)). Ce gain est fondé sur une estimation de ce qui sera visible dans la configuration cible mais il est difficile à estimer car, si les observations sont plus fréquentes que les affectations de cibles, il nécessite de calculer le chemin qui optimise le gain d'information le long de la trajectoire. Or, le gain d'information sur le chemin est égal à la somme des gains de chaque état. L'ajout d'un état modifie le gain d'information de l'état suivant. Cette propriété non-markovienne (le gain d'information d'un état ne dépend pas uniquement de son état mais aussi des états précédents) rend le calcul du gain d'information d'un chemin coûteux. Une approximation est de maximiser le gain d'information de la configuration cible sans prendre en compte le gain d'information sur le chemin y menant.

Nous utilisons une approche frontière car elle est simple et efficace en comparaison des approches fondées sur la maximisation du gain d'information.

Limitations de l'approche actuelle

L'approche par affectation de frontières aux robots fonctionne correctement et permet d'explorer tout l'environnement lorsque toutes les frontières découvertes sont accessibles et observables.

Toutefois, selon les caractéristiques des robots et leur capacité de perception, l'approche actuelle présente quelques limitations. Nous présentons, ci-après, ces différentes limitations et nous proposons une approche fondée sur les frontières pour les surmonter.

Frontière observable mais inaccessible

Lors de l'affectation d'une frontière à un robot, les cellules frontières peuvent se trouver dans une zone non atteignable par le robot. Dans ces situations, l'approche actuelle ne considère pas ces frontières et l'exploration ne sera pas complète. Ceci est le cas pour toutes les cellules frontières identifiées se trouvant en dehors de l'espace de configuration du robot. L'approche actuelle ne considérera pas ces frontières car l'identification des cellules frontière s'effectue sur

l'espace de configuration et la propagation des vagues s'effectue également sur cet espace. La figure 7.8 illustre une telle situation : le robot a identifié une frontière mais celle-ci n'est pas accessible car elle se trouve dans un couloir trop étroit pour le robot. Si le robot était affecté à cette frontière, il ne pourrait calculer de chemin y menant.

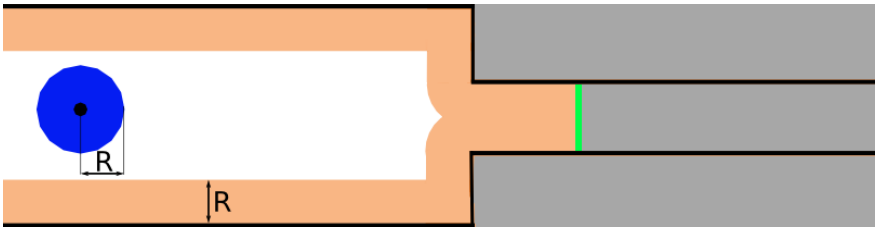


FIGURE 7.8 – Frontière observable mais non accessible

Frontière inobservable et inaccessible

Toutes les frontières en dehors de l'espace de configuration du robot ne sont toutefois pas observables ; une frontière peut ne pas être observable. C'est le cas pour toutes les frontières pour lesquelles il n'existe pas de configuration permettant de l'observer. Par exemple, dans la situation illustrée figure 7.9 le robot identifie une frontière mais ne peut l'observer. En effet, dans ce cas, les configurations permettant de l'observer sont en dehors de son espace de configuration.

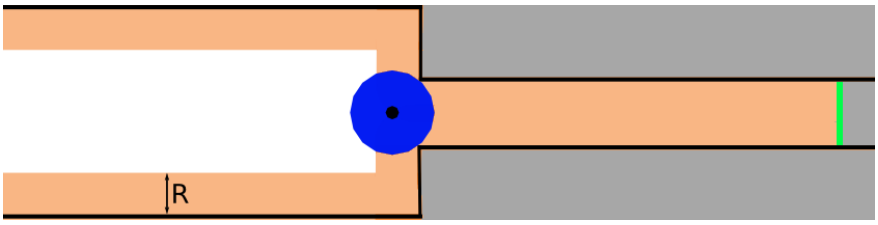


FIGURE 7.9 – Frontière non observable

Frontière accessible mais inobservable

Les frontières accessibles (dans l'espace de configuration du robot) ne sont pas systématiquement observables. Avec un capteur 2D de type LIDAR les frontières accessibles sont observables car il n'y a pas de distance minimale pour observer une frontière. En revanche, avec un capteur 3D de type Kinect le capteur n'observe qu'à partir d'un mètre et la frontière peut se trouver dans un espace ne permettant pas le recul nécessaire à son observation. Les frontières accessibles peuvent donc être accessibles mais non observables. La figure 7.10 montre un exemple d'une telle situation.

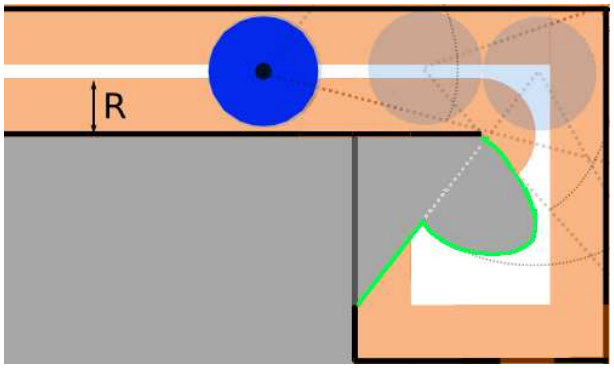


FIGURE 7.10 – Frontière accessible mais inobservable : le robot peut atteindre la frontière en vert mais il ne pourra pas l'observer car il n'existe pas de configuration permettant son observation.

Ces limitations démontrent la nécessité de déterminer l'observabilité des cellules frontières c'est-à-dire de déterminer une configuration atteignable par le robot permettant d'observer ces cellules.

7.2.3 Calcul des configurations d'observation

Cette section décrit la méthode permettant d'identifier la configuration d'observation d'une frontière si c'est possible. Une cellule frontière est observable si il existe un segment reliant cette cellule à une cellule de l'espace de configuration qui n'intersecte pas d'obstacle et dont la longueur est comprise dans l'intervalle de distance d'observation du capteur.

La figure 7.11 illustre le calcul de cellule d'observation Kinect. Nous traçons plusieurs cercles de rayons différents (compris dans l'intervalle de distance de perception du capteur) centrés sur chaque cellule frontière. Nous utilisons un pas de rayon de 0.25 mètre compris entre 1 mètre et 1,5 mètre mais ceci est paramétrable suivant la précision requise. Plus le nombre de cercles est important plus la configuration permettant l'observation d'un maximum de cellules frontières du groupe sera précise.

Un accumulateur (un tableau associant une valeur quantifiée à chaque cellule) est construit pour les cellules où passent au moins un cercle. Les cercles sont tracés en utilisant l'algorithme de Bresenham qui est efficace (Bresenham, 1977). Pour chaque cellule du cercle ayant un segment sans obstacle vers le centre du cercle (la cellule frontière) la cellule correspondante dans l'accumulateur est incrémentée. Le maximum de l'accumulateur est choisi comme la cellule d'observation de cette frontière car elle correspond à la cellule d'où un maximum de cellules frontières sera observable. Nous déterminons, ensuite, l'orientation de la configuration, en calculant la moyenne des angles en direction des cellules frontières ayant contribué à l'accumulateur.

Si aucune cellule de l'accumulateur n'est incrémentée alors aucune cellule frontière de ce groupe de frontière n'est observable pour les distances d'observation choisies (rayon des cercles). Il est alors possible d'agrandir les rayons ou la taille de pas entre chaque rayon pour déterminer plus précisément l'observabilité de la frontière. Les frontières n'étant pas observables ne sont pas prises en compte pour l'affectation car elle n'ont pas de cellules d'observation.

Les cellules d'observation ainsi calculées sont affectées de la même manière que les frontières avec l'algorithme *MinPos* ou *MPG* en utilisant la cellule d'observation comme point de départ de la propagation des vagues.

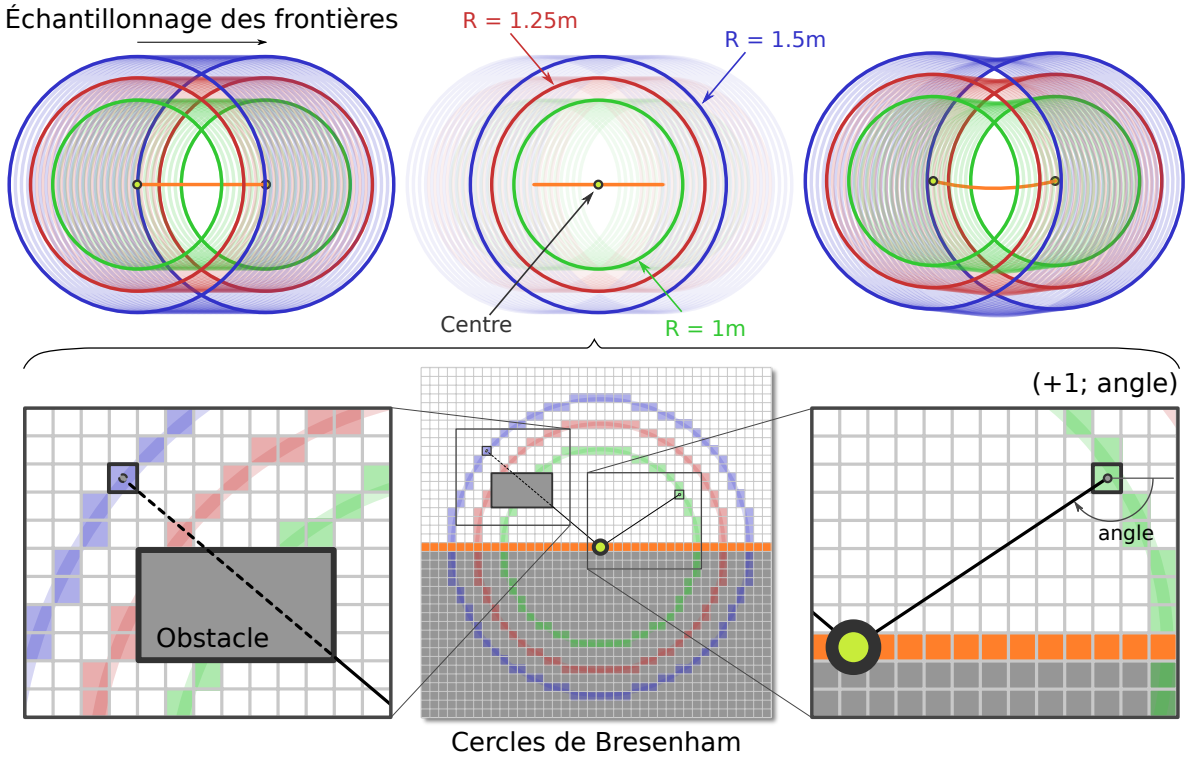


FIGURE 7.11 – Calcul de la configuration d'observation. Les cellules frontières sont de couleur orange, l'espace libre est blanc, l'espace inexploré en gris. En haut : illustration des cercles de différents rayons centrés sur les cellules frontières. En bas : pour chaque cellule des cercles discrétisés un accumulateur est incrémenté si le segment entre la cellule du cercle et la cellule frontière considérée n'intersecte pas d'obstacle.

Cellule multiangle

Après le calcul des différentes configurations d'observation, il est possible que plusieurs d'entre elles se trouvent proches. Ceci peut obliger un robot à observer une frontière, à se déplacer un peu puis à observer une seconde frontière. Pour cela nous regroupons les configurations d'observation proches dans des cellules qualifiées de multiangle. Pour chaque frontière la cellule d'observation est calculée, si plusieurs cellules d'observation sont proches les unes des autres, elles sont fusionnées pour former une cellule d'observation multiangle. La cellule d'observation correspondra à l'observation de plusieurs frontières.

En conclusion de ce chapitre, rappelons que l'approche consistant à affecter les frontières fonctionne bien avec les capteurs ayant une grande portée (relativement à l'environnement), un angle de champ de vision important ($>180°$) et une fréquence d'acquisition élevée (relativement à la vitesse de déplacement du robot). Avec les capteurs n'ayant pas ces caractéristiques, il est nécessaire d'identifier les configurations des robots permettant l'observation des frontières.

Le chapitre suivant décrit l'implémentation robotique des approches proposées.

Chapitre 8

L'architecture Cart-O-Matic

Sommaire

Ce chapitre décrit la mise en œuvre de notre approche avec des robots réels définis spécifiquement pour le défi robotique Carotte (résultats présentés dans le chapitre suivant). Nous commençons par présenter le robot MiniRex, développé par le laboratoire LISA en 2012. Nous présentons ensuite nos travaux communs pour implémenter l'approche d'exploration multirobot présentée dans les chapitres précédents. Il s'agit des couches logicielles concernant la cartographie multirobot, la planification de trajectoire et la navigation.

8.1 Défi CAROTTE

Le développement des robots et de l'architecture Cart-O-Matic a été guidé par les contraintes fixées par le défi robotique CAROTTE (CArtographie par RObot d'un TErritoire). C'est un défi robotique national consistant à cartographier un environnement structuré *a priori* inconnu et à y localiser des objets de manière complètement autonome (sans intervention humaine) et dans un temps contraint.

L'environnement à explorer était de type appartement, construit à l'intérieur d'une arène d'environ $120m^2$. Contrairement à un appartement classique, le sol n'était pas constitué d'un matériel unique et globalement lisse. On pouvait y trouver du gazon et des grilles métalliques (caillebotis pressé) et des bacs de terre ou de graviers. En effet, afin d'accroître la difficulté de l'exploration et de la navigation, ces zones pièges étaient mises en place par les organisateurs. De

plus, on pouvait trouver des parois réfléchissantes (miroirs) ou transparentes (plexiglas) rendant la localisation et la cartographie par capteurs infra-rouges (Kinect, LIDAR) difficiles.

Pour assurer une égalité des moyens embarqués dans l'arène entre les équipes, la valeur globale du système robotisé participant aux épreuves devait être inférieure à 50 000 euros. La communication devait utiliser la technologie WiFi, qui respecte la norme française en termes de puissance d'émission, et qui était suffisante pour couvrir toute l'arène mais avec peu de stabilité. L'entrée (et la sortie) du système robotisé dans l'arène se faisait par une zone dont la géométrie était connue, afin que les robots puissent recaler leur cartes par rapport à l'orientation de l'arène et que, ainsi, toutes les équipes fournissent leur carte dans le même repère. Le système robotisé était positionné dans la zone d'entrée par l'équipe et la position de départ dans la zone était laissée au libre choix de l'équipe.

Les cartes métriques 2D et 3D à fournir en fin de mission pour l'évaluation devaient avoir une résolution de 2 cm. La carte sémantique à produire devait montrer le découpage en pièces de l'environnement. L'exploration de l'arène devait aussi permettre de produire un fichier contenant la description des objets présents dans l'arène avec pour chaque objet sa position, l'estampille temporelle de son observation, une photo de l'objet et la classe à laquelle il appartient. Le règlement faisait part en effet des objets éventuellement présents dans l'arène selon plusieurs catégories.

8.2 Robot MiniRex

Le consortium Cart-O-Matic a développé la plateforme baptisée MiniRex[11](MINI Robot d'EXploration) illustrée en figure 8.1. Cette plateforme a été conçue et fabriquée à l'université d'Angers, conjointement entre le LISA et l'IUT d'Angers, sous la direction de Philippe Lucidarme.

Le MiniRex est un robot équipé de plusieurs capteurs :

- Un télémètre laser ou LIDAR (*LIght Detection and RAnging*) infra-rouge de marque Hokuyo. C'est un télémètre mono-couche d'une portée de 30 mètres et d'un angle de vue horizontal de 270 degrés avec une résolution de 0.25 degré et une précision de 50 millimètres maximum dans des conditions normales (sans éclairage intense comme la lumière directe du soleil). Sa fréquence d'acquisition est de $40Hz$. C'est le seul capteur utilisé pour la localisation et le principal capteur utilisé pour l'exploration. Il mesure la distance aux objets en mesurant le temps de vol entre l'émission et la détection d'une lumière laser émise.
- Une Kinect qui est une caméra 3D de couleur et de profondeur donnant ainsi une image couleur dont chaque pixel peut être associé à une profondeur. Le champ de vision sur le plan horizontal et sur le plan vertical sont respectivement de 57 degrés et de 43 degrés fournissant une image de 640×480 en couleur 32 bits. Dans cette image, la profondeur est disponible pour les objets situés entre 1 et 4 mètres. La fréquence d'acquisitions de la Kinect est de 30 images par seconde. Comme le LIDAR ce capteur est actif. Il utilise une lumière pour déterminer la distance aux objets. La principale différence est qu'il ne mesure pas le temps de vol mais analyse la déformation d'une grille de point projetée en laser infra-rouge. C'est ce capteur qui sert à la reconstruction 3D. La taille des images de profondeur acquises par ce capteur est importante et leur traitement est coûteux (nous n'utilisons que quelques images par minute).

11. depuis 2011 pour la finale du défi Carotte 2012

– Trois capteurs ultra-sons installés à l'avant du robot. Ils ont une portée assez petite relativement aux autres capteurs utilisés. Ils sont utilisés pour détecter des obstacles non détectés par les autres capteurs devant le robot, principalement les obstacles transparents ou réfléchissants.
– Un capteur d'inclinaison, ou inclinomètre, permet de détecter si le robot monte sur un obstacle ou une barre de seuil.
– Un capteur mesure la tension utilisée par les moteurs. Il permet de détecter quand le robot force contre un obstacle qui n'aurait pas été détecté par les capteurs cités ci-dessus.

Le processeur du MiniRex est un Atom à 1,2GHz. La puissance de calcul du MiniRex est donc assez faible en comparaison des processeurs actuels. La mobilité du MiniRex est assurée par deux moteurs actionnant des chenilles.

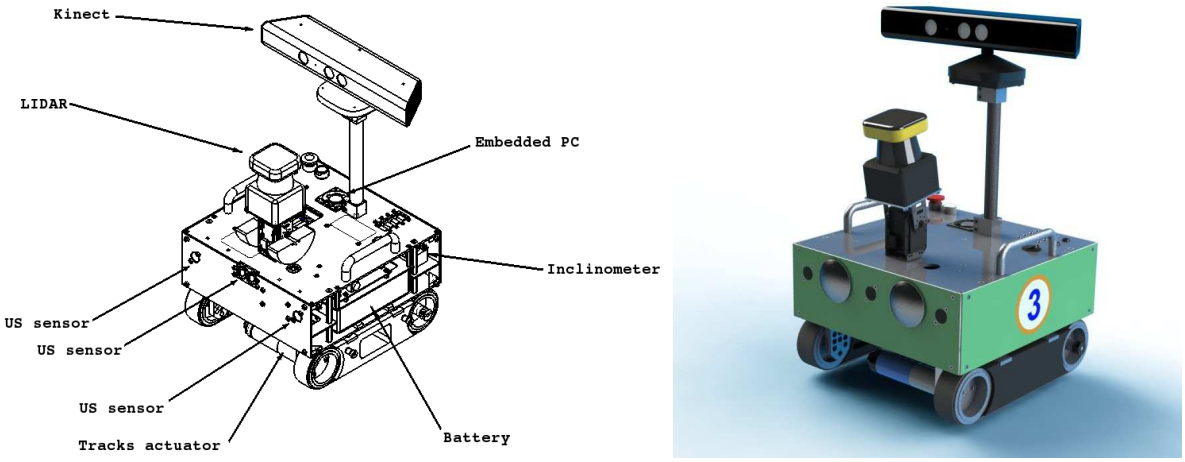


FIGURE 8.1 – Robot MiniRex.

8.3 Architecture logicielle

Nous avons fait le choix, pour des raisons de robustesse, de développer un seul processus de contrôle fondé sur une machine à états chargée de coordonner les différentes phases de l'exploration (démarrage, navigation, gestion multirobot, acquisitions ...). La machine à états repose sur des fonctions organisées en cinq modules (voir figure 8.2). Nous décrivons, ci-après, succinctement ces modules :

– Le module CheckUp permet de vérifier intégralement le bon fonctionnement de toutes les parties matérielles et logicielles du robot (chargement du fichier de configuration, fonctionnement des capteurs et actionneurs, connexion à tous les éléments du robot, allocation mémoire, espace disque, communication avec la machine déportée, position initiale, etc...). Cette vérification est effectuée avant le lancement de la mission et les problèmes éventuels sont remontés vers la machine déportée.
– Le module de bas niveau MiniRex assure l'interface avec la partie matérielle (actionneurs et capteurs).
– Le module Slam-O-matic assure la localisation et la cartographie simultanée (SLAM) pendant les phases de déplacement du robot (phases de navigation).

– Le module Expl-O-matic est un module de haut niveau qui calcule la stratégie d'exploration et la répartition des robots dans l'environnement. Son cœur est l'algorithme *MinPos* proposé au chapitre 4.
– Le module Plan-O-matic permet la planification de trajectoires une fois les frontières assignées.

La figure 8.2 illustre les interactions entre les principaux modules du robot (machine à états, modules et micrologiciel (*firmware*)).

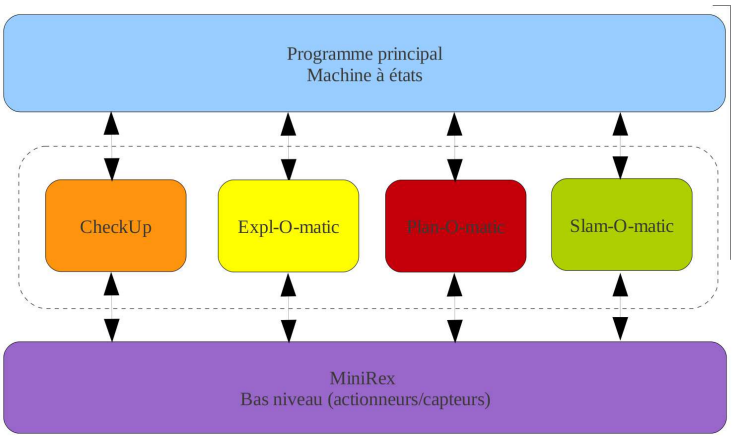


FIGURE 8.2 – Modules principaux composant l'architecture logicielle du MiniRex

La figure 8.3 illustre la machine à états principale du robot MiniRex permettant de définir son comportement. Nous en décrivons, ici, les principaux états dans lesquels le robot va se trouver, correspondant aux différentes phases de la mission, et leur fonctionnement.

Les noms d'états ont un préfixe :
– INIT pour les états appartenant à la phase d'initialisation,
– EXPL pour les états appartenant à la phase d'exploration,
– ERR pour les états de résolution d'une erreur de planification de trajectoire vers une cible donnée,
– HOME pour les états appartenant à la phase de retour vers la position initiale du robot,
– END pour les états appartenant à la phase de fin de mission.

Dans la suite, nous décrivons plus en détail les phases principales de la mission : l'initialisation et l'exploration.

Phase d'initialisation

Au démarrage du programme, le robot est dans l'état `INIT_WAIT_FOR_START` où il effectue une vérification du système et attend un démarrage de la mission. La mission démarre soit par la pression d'un bouton du robot soit par un ordre de démarrage par le réseau (interface de contrôle de la mission). Au démarrage de la mission, le robot effectue une première acquisition Kinect (`INIT_SINGLE_CAPTURE`) et commence à naviguer après un décompte de temps (`INIT_TIMER`), différent sur chaque robot, pour éviter que les robots démarrent simultanément et se gênent à la sortie de l'aire de départ qui est étroite.

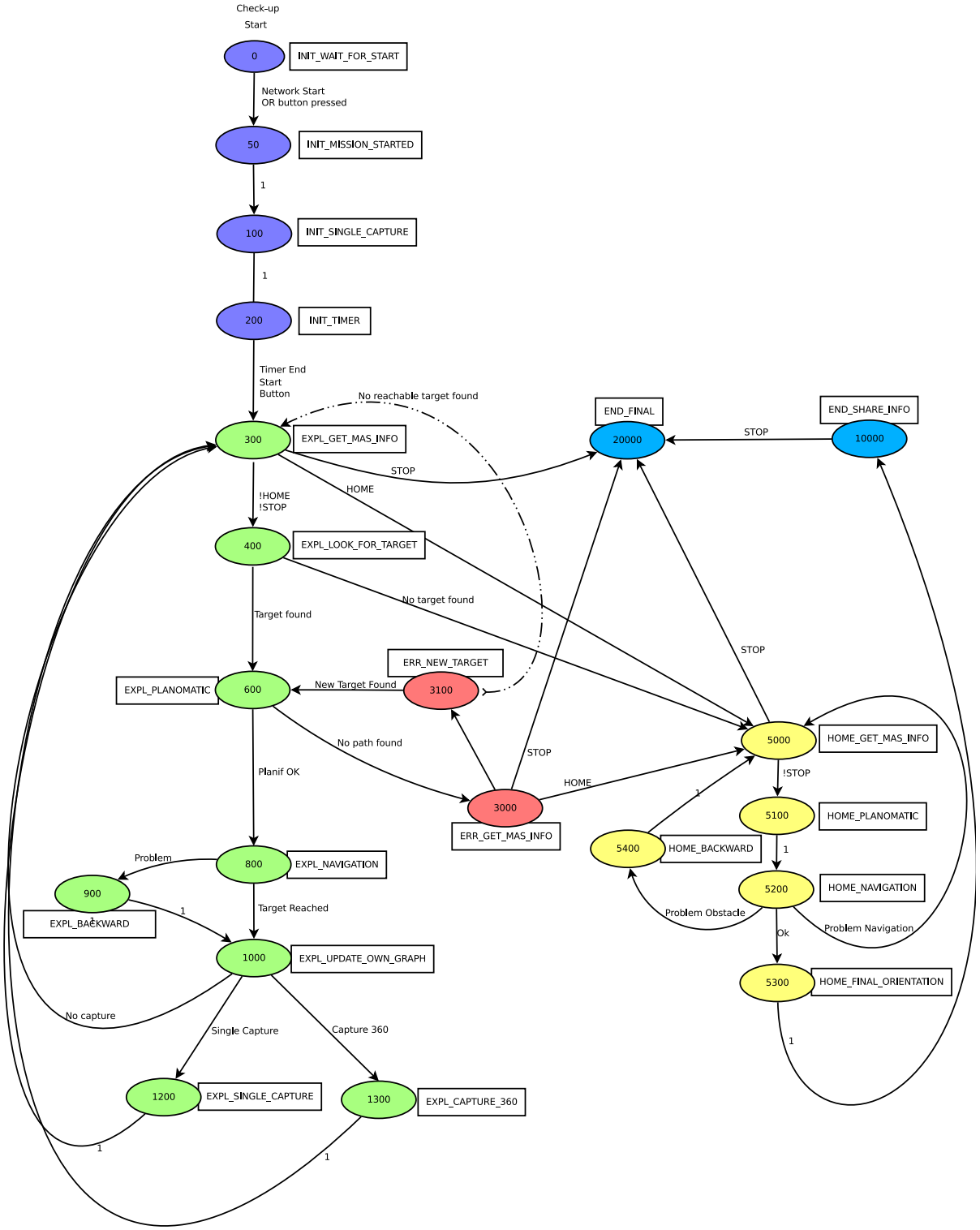


FIGURE 8.3 – Machine à état du MiniRex

Phase d'exploration

Durant la phase d'exploration, le robot itère sur les états suivants :
- `EXPL_GET_MAS_INFO` où le robot construit la carte de navigation avec sa carte locale et les cartes et les positions reçus des autres robots (voir section section 8.4.3),
- `EXPL_LOOK_FOR_TARGET` où le robot exécute les algorithmes d'affectation pour déterminer sa prochaine cible,
- `EXPL_PLANOMATIC` où le robot planifie la trajectoire pour atteindre la cible (voir section 8.5),
- `EXPL_NAVIGATION` où le robot navigue vers la cible (voir section 8.6 Navigation) et effectue simultanément le SLAM (voir section 8.4.2,
- `EXPL_UPDATE_OWN_GRAPH` où le robot met à jour sa carte de navigation à partir des données acquises lors de la navigation, par exemple, la détection d'un miroir avec le capteur ultrason (voir section 8.6 Navigation).

Ces états sont fortement liés à la grille de navigation. En effet, soit ils l'alimentent (`EXPL_GET_MAS_INFO`, `EXPL_NAVIGATION`, `EXPL_UPDATE_OWN_GRAPH`) soit ils l'exploitent (`EXPL_LOOK_FOR_TARGET`, `EXPL_PLANOMATIC`). Nous détaillons sa structure dans les paragraphes suivants. Ensuite les principaux modules de l'architecture sont détaillés. Notons que l'état `EXPL_LOOK_FOR_TARGET` correspond à l'exécution de l'algorithme d'affectation et n'est pas détaillé ici car il est explicité dans les chapitres 4 et 7.

8.3.1 Carte de navigation

La carte ou grille de navigation regroupe les informations provenant des différents capteurs, et des autres robots afin de calculer l'espace de configuration du robot comme nous l'avons présenté en section 7.1.1. Les algorithmes de calcul de cible, d'affectation de cible et de planification de trajectoire sont exécutés dessus.

La carte de navigation est multicouche. Certaines couches de cette carte sont associées à une source (Kinect, limiteur de couple, lidar, ultrasons ...), ce qui permet de dissocier l'origine des obstacles. Les autres couches servent à stocker le résultat des calculs de la grille de configuration et de la couche servant à calculer la proximité d'obstacle (utilisée par le module de planification de trajectoire). On note que la grille de navigation a une résolution de $8cm^2$; cette taille est inférieure à la résolution de la carte à construire pour le défi mais elle est uniquement utilisée pour la navigation et la coopération entre robots (échange entre robots). Chaque cellule de la grille de navigation stocke les informations des différentes couches. Les paragraphes suivants décrivent la nature de ces différentes couches composant la carte de navigation, la manière dont elles sont calculées ainsi que leur utilisation.

Couche LIDAR

Une première couche stocke les obstacles provenant du LIDAR placé à 30cm de hauteur. Celle-ci est directement calculée à partir de la grille d'occupation du SLAM (présenté section 8.4.2)ayant une résolution de $2cm^2$, on note qu'une réduction de résolution a lieu lors de son calcul. Un seuil obstacle est utilisé pour déterminer, à partir de la probabilité d'occupation, si la cellule de la grille d'occupation est considérée comme un obstacle. Nous considérons que les cellules de la grille de navigation sont occupées par un obstacle LIDAR si au moins trois cellules des 16 correspondantes dans la grille d'occupation sont occupées. Cette opération permet de

filtrer du bruit de la grille d'occupation mais ne permet pas d'éviter les obstacles de surface au sol de $6cm^2$ et moins. Ce processus a été présenté en section 7.1.1.

Couche Kinect

Une deuxième couche stocke les obstacles détectés par la Kinect. En effet, tous les obstacles dont la hauteur est comprise entre le sol et la hauteur du robot (50cm) ne sont pas perçus par le LIDAR. Pour les éviter, ils sont ajoutés dans la couche Kinect en projetant leur position sur le sol. Ceci permet d'éviter les obstacles qui ne sont pas sur le plan du LIDAR mais à hauteur du robot.

Couche SONAR

Une troisième couche stocke les obstacles provenant du télémètre ultra-sons (sonar). En effet, les matériaux transparents, réfléchissants et dans une moindre mesure les matériaux de couleurs sombres ne sont pas perçus par le LIDAR et la Kinect car ces capteurs utilisent tous les deux des rayons infra-rouges qui traversent, sont réfléchis ou absorbés par les différents matériaux précités. Cette couche permet donc de prendre en compte les vitres et les miroirs de l'environnement car ceux-ci sont détectés par le capteur ultra-sons.

Couche Limiteur, inclinomètre et divergence du SLAM

Une quatrième couche stocke les obstacles ou problèmes provenant du limiteur de couple, de l'inclinomètre ou d'une divergence détectée par le SLAM (quand le robot détecte qu'il a été kidnappé). Lors de leurs détections, la navigation est interrompue et un obstacle est ajouté dans cette couche.

Couche position des autres robots

Une cinquième couche stocke la position des autres robots. À la réception des communications, les positions reçues d'autres robots sont conservées dans cette couche. La position précédente de chaque robot (i.e. stockée dans la couche avant la nouvelle transmission) est effacée et remplacée par la nouvelle position reçue. De plus, lorsque la position d'un robot n'est pas mise à jour pendant un temps paramétrable celle-ci n'est plus prise en compte car, étant trop ancienne, elle est surement devenue obsolète.

Couche cartes des autres robots

Une sixième couche stocke les cartes reçues des autres robots. Une carte est stockée par autre robot de la flottille. Comme pour les positions des autres robots, la réception d'une nouvelle carte d'un robot écrase la version précédente de la carte de ce robot. Mais ici, les cartes anciennes sont utilisées indéfiniment (elles ne sont jamais considérées comme obsolètes) car elles peuvent contenir des informations utiles pour la navigation ou l'exploration. En effet, l'environnement est statique les informations récoltées par un autre robot ne sont donc jamais obsolètes.

Couche TomThumb

Une septième couche est activée lorsque le robot ne trouve pas de chemin, elle garantit que le robot possède toujours un chemin vers son point de départ. Pour cela, à chaque déplacement

le robot pose "un petit caillou" sur la cellule de cette couche ce qui signifie, pour lui, que s'il est passé par là une fois il doit pouvoir y repasser.

Couches exploration

Les huitièmes et neuvièmes couches sont des couches d'exploration, elles permettent de connaître les zones explorées ou inexplorées en maintenant une grille de cellule dont le statut est explorée ou inexplorée. L'une d'entre elles contient les cellules observées par le capteur LIDAR et l'autre les cellules 2D théoriquement observées par le capteur Kinect i.e. les cellules qui sont dans le champ de vision théorique de la Kinect en effectuant un lancé de rayon et en marquant les cellules comme explorées. On note que les rayons s'arrêtent sur les obstacles LIDAR.

Couche de dilatation des obstacles

La dixième couche contient les informations concernant tous les obstacles détectés avec tous les types de capteurs. Pour cela, toutes les couches représentant des obstacles sont fusionnées sur cette couche appelée couche de dilatation (figure 8.4). Elle porte ce nom car, à partir de cette couche, l'espace de configuration est calculé en dilatant les obstacles du rayon du robot. En effet, la carte de navigation est utilisée pour calculer la cible à affecter au robot (voir chapitre 7) et la trajectoire pour s'y rendre (voir section 8.5).

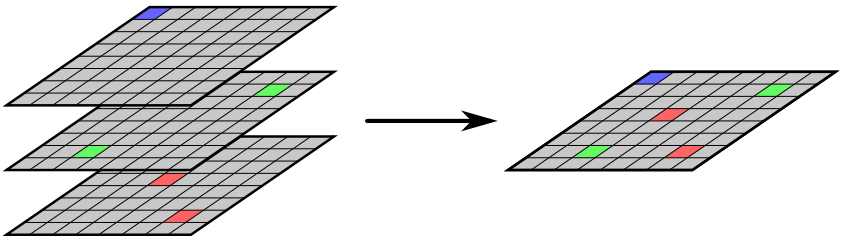


FIGURE 8.4 – Fusion des différentes couches de la carte de navigation

Couche de proximité des obstacles

La onzième et dernière couche maintient pour les cellules inoccupées de la couche fusionnée (couche de dilatation), la distance à l'obstacle le plus proche. Dans le calcul de trajectoire, cette couche sert à pénaliser les trajectoires passant près des obstacles.

La figure 8.5 illustre un exemple de carte de navigation obtenue avec un robot après une exploration des couloirs du LISA à Angers. La couche TomThumb (en gris) montre, dans la partie centrale de l'image, la trajectoire où le robot a essayé, à plusieurs reprises, de traverser une baie vitrée.

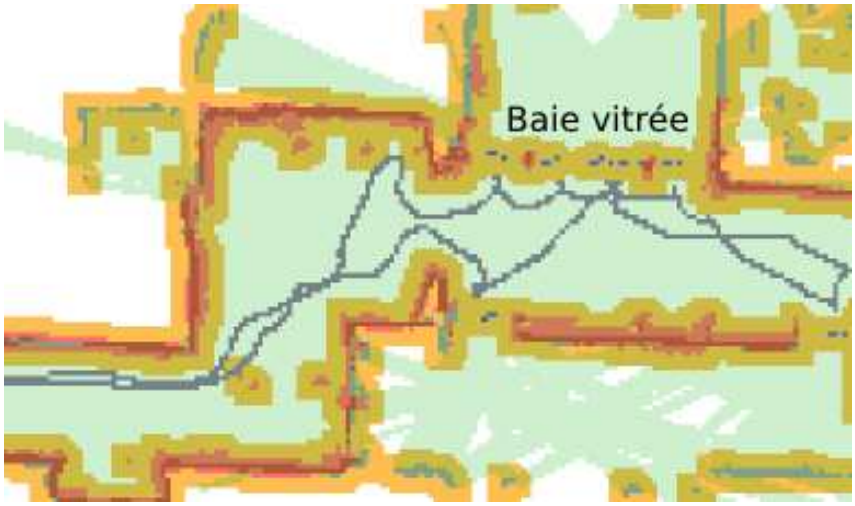


FIGURE 8.5 – Exemple de carte de navigation incluant par transparence les différentes couches. Code couleur : en vert la couche LIDAR le vert clair étant vide le vert foncé occupé, en rouge, les obstacles de la couche Kinect, en bleu les obstacles de la couche SONAR, en gris la couche TomThumb, en jaune la couche de dilatation et en blanc l'inexploré.

Maintenant que nous avons montré comment est construite la carte permettant la navigation des robots à l'aide d'informations stockées sur différentes couches, nous allons voir comment est construite la grille d'occupation qui alimente la couche LIDAR avec un rappel du fonctionnement d'un algorithme de SLAM puis une brève description du SLAM que nous utilisons.

Nous montrons, ensuite, comment une carte similaire est construite sur chaque robot (cartographie multirobot) avant de décrire les modules de planification de trajectoire et de navigation.

8.4 Localisation et cartographie multirobot

La construction d'une carte complète de l'environnement implique le déploiement de robots ainsi que leur localisation dans une carte partiellement construite. Le problème de 'la cartographie et de la localisation simultanée' (en anglais SLAM, *Simultaneous Localization and Mapping*) est un champ de recherche important en robotique depuis qu'il a été ouvert par l'article fondateur de Smith et Cheeseman Smith and Cheeseman (1986).

Dans cette section, nous commençons par décrire les différentes représentations possibles d'une carte dont celle que nous avons choisie. Nous expliquons le principe général du SLAM, monorobot dans un premier temps et multirobot dans un second temps, pour enfin décrire notre méthode de cartographie multirobot.

8.4.1 Représentation de la carte

Les trois principaux types de représentation sont les cartes métriques comprenant les grilles d'occupation, les cartes topologiques et les représentations hybrides décrites ci-après :

– une carte métrique peut être une une grille d'occupation Elfes (1989). Celle-ci est une discrétisation de l'environnement en cellules à la résolution souhaitée où chaque cellule stocke la probabilité qu'elle soit occupée par un obstacle. L'estimation de cette probabilité est

calculée en considérant que la probabilité d'occupation de chaque cellule est indépendante des probabilités d'occupation des cellules voisines, ce qui constitue une approximation,
- une carte topologique est un graphe dont les nœuds sont des amers de l'environnement,
- les représentations hybrides peuvent être fondées sur une carte topologique où des informations métriques sont présentes dans la relation entre les nœuds. Elles peuvent aussi être fondées sur un graphe où les nœuds sont des cartes métriques locales.

Nous avons choisi d'utiliser des cartes métriques et, plus particulièrement, nous utilisons une grille d'occupation pour la construction de carte à partir des informations du LIDAR. Ces grilles sont largement utilisées pour la cartographie car elles sont simples et efficaces (Yguel et al., 2006). La cartographie avec des grilles d'occupation permet de créer des cartes sous forme de grille à partir d'observations bruitées quand la pose (x,y,θ) du robot est connue.

La grille d'occupation maintient la probabilité *a posteriori* de l'occupation d'une cellule étant donné des observations. Ainsi une cellule non-observée a une probabilité d'occupation de $0,5$. L'efficacité des grilles d'occupation réside dans le fait que la probabilité d'occupation d'une cellule est considérée comme indépendante de toutes les autres cellules notamment de ses cellules voisines. Il est fréquent d'utiliser une représentation des probabilités en *Logit*[12] pour simplifier la mise à jour qui n'est alors plus qu'une addition. Nous avons choisi d'en utiliser une version simplifiée : la probabilité d'occupation est un entier qui est incrémenté quand la cellule est observée comme occupée et décrementé quand elle est observée comme vide.

Maintenant que nous avons choisi un type de représentation, nous allons voir comment construire la carte tout en localisant les robots.

8.4.2 SLAM monorobot

Le SLAM est un problème difficile car pour se localiser un robot a besoin d'une carte et pour construire une carte il lui faut connaître sa position. Un algorithme de SLAM produit une carte et fournit la position du robot à partir de ses perceptions :
- Entrées : perception proprioceptive et extéroceptive.
- Sorties : carte de l'environnement, trajectoire suivie par le robot.

Une étape importante du SLAM est d'associer les informations courantes avec celles perçues précédemment.

Les perceptions du robot sont, en général, composées de :
- l'odomètrie (perception proprioceptive) permettant de connaître le mouvement effectué mais celle-ci dérive avec la distance parcourue,
- d'informations provenant d'un télémètre laser (perception extéroceptive) fournissant en général des informations de distance sur un plan horizontal.

Les caméras donnent des informations en deux dimensions mais pas d'information de profondeur. De plus, les images sont complexes à exploiter pour la construction d'une carte et le traitement d'image est coûteux en calculs. Ainsi les approches reposant uniquement sur une caméra construisent en général une carte des amers choisis dans l'environnement mais pas de cartes des murs et des objets de l'environnement.

En utilisant un modèle de mouvement, u_t, les données proprioceptives au temps t, servent à la prédiction de l'état du robot x_t (en général (x, y, θ)) à partir de l'estimation de l'état précédent. Cette prédiction est corrigée avec l'observation z_t reçues. La figure 8.6 montre le sens

12. ou *log-odds*, $logit(p) = log(\frac{p}{1-p})$. Ceci transforme la probabilité d'occupation $p \in [0, 1]$ en évidence $]-\infty, +\infty[$

des relations entre les variables cachées (x et m, état et carte) et les variables observées (u et z, odométrie et observation).

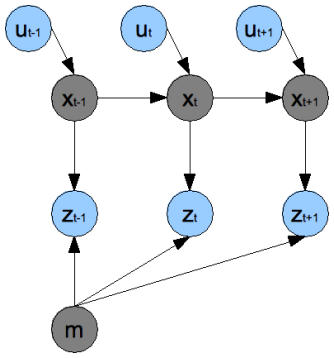


FIGURE 8.6 – SLAM, en bleu les variables observées et en gris les variables cachées

Nous pouvons distinguer deux types de SLAM différenciés par la façon de traiter ces informations :

– Full SLAM où l'on cherche à estimer toute la trajectoire du robot à chaque ajout de données odométriques et d'observation :

$$p(x^{0..t}, m|u^{0..t}, z^{0..t}) \tag{8.1}$$

– Online SLAM où l'on estime uniquement la dernière position du robot :

$$p(x^{t}, m|u^{0..t}, z^{0..t}) \tag{8.2}$$

L'algorithme de SLAM que nous utilisons est de type Online SLAM, il est nommé Slam-O-Matic. Il a été développé par notre partenaire à Angers. Il consiste à minimiser la distance entre le scan LIDAR et la carte partiellement construite. L'odométrie n'est pas utilisée pour estimer le déplacement effectué entre deux observations. La position initiale de recherche de minimum est démarrée à partir de la dernière position calculée du robot. Nous ne donnons pas les détails de son implémentation car ils sont protégés par un brevet (Lucidarme and Lagrange, 2011). La figure 8.7 illustre une carte construite de plus de $100m^2$ de longueur avec l'algorithme Slam-O-Matic.

FIGURE 8.7 – Carte construite à l'université d'Angers avec l'algorithme Slam-O-Matic (Lucidarme and Lagrange, 2011))

8.4.3 Cartographie multirobot

Pour effectuer une cartographie multirobot, il est nécessaire d'utiliser les perceptions de l'ensemble des robots. La mise en commun d'informations peut être effectuée à plusieurs niveaux : en communiquant les données extéroceptives (scans LIDAR) ou une carte construite sur chaque robot. Ces deux niveaux ne sont pas équivalents :

- la communication des données extéroceptives permet d'exécuter un SLAM multirobot (localisation de l'ensemble des robots et cartographie avec l'ensemble des perceptions des robots au fur et à mesure des acquisitions ou en utilisant leur estampille temporelle),
- la communication de cartes permet de construire une carte commune résultat de la fusion des cartes construites sur chaque robot et d'y localiser les autres robots avec leurs coordonnées.

Notons que l'utilisation de plusieurs robots peut générer des interférences entre les capteurs utilisés (notamment les ultra-sons). Les robots peuvent se gêner mutuellement dans leur perception, il peut donc être nécessaire de différencier les robots des obstacles pour ne pas les intégrer dans la carte.

Partage au niveau des données de perception extéroceptive

La technique de SLAM multirobot que nous utilisons consiste à exécuter l'algorithme de SLAM avec une recherche globale de la meilleure position en utilisant successivement une perception de chaque robot. Ceci permet d'avoir une construction de carte précise sur chaque robot à partir des perceptions de l'ensemble des robots de la flottille. Cependant, cette technique ne peut pas être utilisée durant l'exploration car elle nécessite d'effectuer le SLAM multirobot sur chaque robot ce qui est trop coûteux pour la capacité de calcul des robots MiniRex et nécessite de nombreuses communications. Afin de construire une carte précise de l'environnement, elle est donc exécutée, sur un ordinateur puissant, à la fin de l'exploration avec les données acquises par la flottille.

Cette technique est utilisée "en ligne" par Howard (2006) avec trois robots doté d'une puissance de calcul importante. Un algorithme *FastSLAM* utilisant une grille d'occupation et un filtre particulaire de type Rao-Blackwellized est exécuté sur chaque robot. Quand les positions initiales sont connues les robots mettent en commun leur données extéroceptives. Lorsqu'elles sont inconnues, les cartes individuelles sont fusionnées avec une lecture "en marche arrière" des chemins parcourus. Le partage des données extéroceptives peut être traité dans plusieurs filtres distribués : d'abord pour un petit groupe de robots, dans les coordonnées de référence d'un leader, puis dans un filtre unique qui estime la position de chaque leader Martinelli (2007).

L'approche d'exploration multirobot proposée dans cette thèse nécessite la construction collaborative d'une carte. Le SLAM multirobot étant exécuté à la fin de l'exploration, il restait nécessaire de construire une carte sur chaque robot fondé sur les observations de l'ensemble des robots.

La première approche que nous avons utilisée pour la construction d'une carte globale "en ligne" similaire sur chaque robot consistait à diffuser à l'ensemble de la flottille les scans laser perçus par chaque robot avec la pose calculée par le SLAM local du robot. Les robots, recevant un scan, l'ajoutaient dans leur carte en utilisant la pose associée. Durant l'exploration, chaque robot effectue son SLAM monorobot et le résultat de cette opération est utilisé par l'ensemble des robots. Cette approche n'utilise que très peu de calcul et construit une carte globale avec la résolution du SLAM sur chaque robot.

Cette approche nécessite toutefois de nombreuses communications et cette approche ne passait pas à l'échelle (pas plus de 4 robots). En effet, les scans laser sont légers mais la fréquence d'acquisition (25 Hz pour le LIDAR des MiniRex) nécessite que chaque robot diffuse ces acquisitions à la même fréquence. Nous avons alors développé une approche plus légère de partage des cartes de navigation construites sur chaque robot, présentée ci-après.

Partage des données au niveau carte

Dans cette approche, les cartes locales construites sur chaque robot sont partagées et fusionnées sur chaque robot (ou sur un serveur central). Avec un repère commun, la fusion des cartes sous forme de grille est relativement simple, la correspondance entre les cellules des cartes de chaque robot est directe. Sans repère commun, le partage de carte peut se faire quand les robots se voient car l'association est alors plus facile (Blanco-Claraco, 2009) ou en effectuant une recherche de la transformation entre les différentes grilles d'occupation (Birk and Carpin, 2006).

Communication inter-robot Afin de construire une carte utilisant les perceptions des autres robots, les robots utilisent une diffusion UDP pour transmettre les informations acquises par chaque robot à l'ensemble de la flottille. Ceci est effectué à une fréquence élevée uniquement pour leur position. Cela permet aux robots de connaître précisément la position des autres robots avant d'effectuer leur affectation de tâche et leur planification de trajectoire. En revanche, la carte est diffusée à une fréquence plus basse car plus volumineuse. Elle est calculée pour limiter sa taille en fusionnant les couches LIDAR, sonar et TomThumb de la grille de navigation. Sa réception prend aussi plus de temps. Les cartes reçues sont mises dans une mémoire tampon pendant la navigation et lues avant le calcul d'affectation.

Fréquence des communications Plus la fréquence de communication est élevée, meilleure sera la coopération entre robots car ils disposeront d'informations plus récentes quand ils calculeront leur affectation.

Il faut toutefois limiter la quantité de données transmises pour éviter une congestion du réseau. En effet, quelque soit le protocole de communication, la congestion du réseau entraîne la perte d'une partie des informations transmises. De plus, l'envoi et la réception de données peuvent être coûteux en calcul pour le robot car, en plus de la gestion de la communication, il doit préparer la carte à envoyer et mettre à jour sa carte lorsqu'il reçoit des données, et éventuellement compresser et décompresser les données.

Pour calculer leur affectation en utilisant les dernières informations transmises, les robots doivent réceptionner les communications avant de mettre à jour leur carte. La réception des communications doit donc être effectuée au minimum à chaque fois que le robot calcule son affectation.

En l'absence de communication, les cartes ne peuvent pas être mises à jour avec les acquisitions des autres robots. Les robots utilisent alors les dernières cartes reçues des autres robots fusionnées avec leur propre carte.

En plus de la fréquence d'envoi et de réception des cartes nous distinguons deux contenus différents échangés :

- les coordonnées des robots,
- la mise à jour de carte.

L'envoi par chaque robot de sa position a un coût très faible (la taille de la donnée étant faible). En revanche, la quantité de données nécessaire pour communiquer la mise à jour des cartes est plus importante. Elle dépend de sa taille, de sa résolution, de son format et, si elle est compressée, de son format de compression.

Avant de calculer son affectation, le robot réceptionne les cartes envoyées par les autres robots. L'envoi peut être plus fréquent (que la réception) mais afin de minimiser les communications, nous avons choisi d'envoyer les cartes à la même fréquence que calcul de leur affectation. Ceci est effectué lorsque le robot s'est déplacé d'une distance supérieure à un seuil (1,5m pour le défi Carotte) ou lorsqu'il a atteint la frontière et effectué une observation. À ce moment, nous sommes sûr que des nouvelles informations sont disponibles.

Après avoir calculer son affectation c'est-à-dire la frontière ou la configuration qu'il doit atteindre, le robot planifie une trajectoire pour s'y rendre en utilisant Plan-O-Matic décrit dans la prochaine section.

8.5 Planification de trajectoire (Plan-O-Matic)

La propagation d'une vague calculée pour l'affectation d'une frontière (décrite au chapitre 6) fournit aussi un chemin au robot mais celui-ci ne tient pas compte des contraintes dynamiques et de non-holonomie du robot. Cette section aborde le calcul d'une trajectoire réalisable par le robot. La trajectoire recherchée minimise le temps d'exécution de la trajectoire plutôt que sa distance (par exemple, le robot met plus de temps à tourner sur lui même qu'à aller tout droit).

Pour cela, nous utilisons un algorithme A* sur graphe à 4 dimensions (x, y, orientation, vitesse). Le graphe est généré à partir de la carte de navigation, les nœuds associés à une cellule occupée de la carte de navigation sont interdits. L'orientation du robot est discrétisée selon 16 angles et la vitesse est limitée à deux états (arrêt ou mouvement). Les transitions entre chaque nœud représentent les transitions physiquement possibles pour le robot prenant en compte les contraintes dynamiques du robot. Pour chaque nœud, il y a 3 transitions possibles quand le robot est à l'arrêt (avancer, tourner à droite ou à gauche) et 4 quand il est en déplacement (s'arrêter, tourner à gauche ou droite et aller tout droit). Ces transitions n'engendrent pas forcément un déplacement d'une cellule, par exemple le robot met 2 cellules pour s'arrêter.

Le calcul d'une telle planification (A* en quatre Dimension) est coûteux si l'heuristique utilisée ne donne pas une bonne estimation du coût réel. Le nombre de nœuds à explorer sera grand et, au pire cas, égal à la taille du graphe. L'heuristique du temps nécessaire pour parcourir la distance à vol d'oiseaux est adéquate lorsque l'environnement ne contient pas d'obstacle. En revanche lorsque l'environnement est encombré, il est nécessaire de modifier l'heuristique pour obtenir pour une planification plus rapide afin de l'embarquer sur les robots. Il a donc été choisi de remplacer l'heuristique « du temps nécessaire pour parcourir la distance à vol d'oiseaux » par la valeur du temps nécessaire pour parcourir la distance fournit par la vague calculée au préalable en deux dimensions depuis le but. Cette heuristique fournit une bonne approximation de la distance de la trajectoire à la cible car elle prend en compte les obstacles. Elle est admissible car elle fournit la distance du chemin le plus court vers la cible, qui est toujours plus court que la trajectoire calculée. Ceci nous permet d'avoir un temps de planification acceptable (en général en dessous de la seconde).

Nous comparons, ici, les méthodes de planification A* avec les heuristiques de la distance euclidienne et de la vague.

Pour comparer les heuristiques nous tirons aléatoirement 500 points et nous calculons une

trajectoire passant par ces 500 points dans l'ordre dans lequel ils ont été générés. Sur l'environnement composé de plusieurs pièces les temps de calcul moyen des trajectoires avec A* utilisant différentes heuristiques sont les suivants :

- 264.3 ms avec l'heuristique toujours égale à 0 (A* est alors équivalent à un Dijkstra),
- 20.6 ms avec l'heuristique égale à la distance euclidienne,
- 14.6 ms avec l'heuristique égale à la distance du chemin calculé avec la propagation de vague dont 11.7 ms pour le A* et 2.9 ms pour la vague.

Ces expériences ont été effectuées sur des environnements différents car certains favorisent une heuristique plutôt qu'une autre. Il est apparu que l'heuristique de la vague permet un calcul de trajectoire plus rapide dès lors que des obstacles sont présents dans l'environnement. En effet, dans un environnement sans obstacles la distance euclidienne donne une bonne approximation de la distance à la cible.

Cette première version d'algorithme A* utilisant la vague a montré qu'il pouvait générer des trajectoires passant près des obstacles. Nous avons donc pénalisé le coût des cellules proches des obstacles. Pour cela, un surcoût de passage par ces cellules est ajouté à l'heuristique ainsi qu'au coût réel du A*. Pour chaque cellule, le surcoût est inversement proportionnel à la distance à l'obstacle le plus proche. Notons qu'il est préférable que la distance maximum, à laquelle les cellules sont pénalisées, soit inférieure à la moitié de la largeur des couloirs de l'environnement. Autrement, le coût de calcul devient important car l'algorithme A* explorera tous les nœuds ayant un coût inférieur, soit tous les chemins permettant de contourner le couloir.

L'algorithme fournit une planification qui génère rapidement des trajectoires facilement réalisables par les robots, voir figure 8.8.

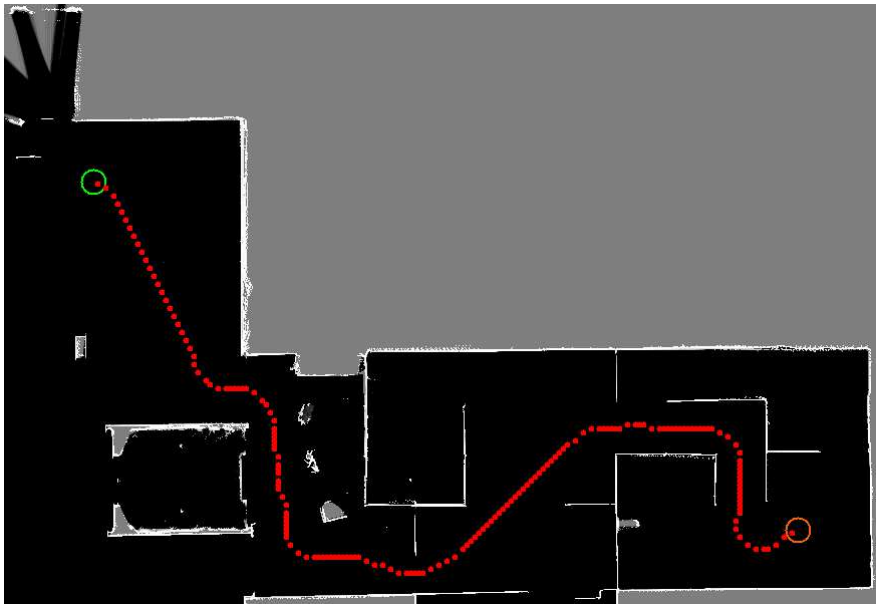


FIGURE 8.8 – Illustration du chemin calculé avec le A* 4D utilisant la vague comme heuristique et la pénalisation des cellules proches des obstacles.

8.5.1 Temporisation de l'affectation des cibles redondantes

En fin d'exploration, il est inévitable que le nombre de robots soit supérieur à celui de frontières. Pour tous les algorithmes respectant la contrainte 3.2 d'assignation d'une frontière à chaque robot, plusieurs robots seront assignées à la même frontière. Avec l'algorithme *MinPos*, quand plusieurs frontières sont dans la même direction, la frontière la plus proche dans cette direction sera assignée à plusieurs robots. Pour éviter les gênes entre les robots assignés aux mêmes frontières, nous avons instauré un mécanisme de coordination simple. Celui-ci consiste à temporiser l'affectation des robots qui sont affectés à une frontière alors qu'un autre robot y est déjà affecté. Pendant le temps d'attente, les robots vérifient les cartes reçues et recalculent leurs affectations, si ils sont affectés de nouveau à une frontière déjà assignée alors ils naviguent dans sa direction.

Une fois la trajectoire planifiée le robot tente de réaliser cette trajectoire le plus fidèlement possible. Ceci est décrit dans la section suivante.

8.6 Navigation

8.6.1 Poursuite de point

L'algorithme A* fournit une liste de nœuds à parcourir pour arriver à l'objectif à partir de l'état initial du robot. La navigation consiste à réaliser physiquement le parcours de cette trajectoire. Pour cela, notre politique est relativement simple, nous avons une liste de points que le robot poursuit avec un asservissement de type proportionnel par rapport à la position renvoyée par le SLAM. Le robot est alors simplement asservi pour aller sur le nœud suivant de la liste, tout en dépilant la liste au fur et à mesure.

8.6.2 Évitement des obstacles rencontrés

Afin d'éviter les collisions, ce suivi de trajectoire peut être interrompu si le robot rencontre l'un des problèmes suivant :

- un obstacle est détecté à l'ultrason
- un obstacle est détecté au limiteur de couple (consommation électrique d'un des moteurs trop élevée)
- un kidnapping est détecté (le robot s'est perdu)
- le SLAM ne détecte pas de déplacement alors que les moteurs tournent
- l'inclinomètre indique que le robot n'est plus à plat (il est donc certainement en train de monter sur un obstacle).

Dans le cas d'une telle interruption, la navigation s'arrête et le robot repart à la recherche d'un point cible en ayant au préalable ajouté un obstacle dans sa carte de navigation pour ne plus y retourner, puis recommence une planification et une navigation. Il est toutefois dangereux de marquer comme obstacle tout élément ayant stoppé le robot. Il peut s'agir d'un autre robot (obstacle temporaire), d'un mur trop proche perçu aux ultrasons, ou encore d'un sol trop rugueux. Le robot peut alors "s'enfermer" et ne plus disposer de chemins pour explorer ou rentrer à l'aire de départ. Pour éviter ce problème nous utilisons la couche TomThumb de la carte de navigation, contenant les zones déjà parcourues physiquement par les robots. Ces zones sont considérées comme libres par la planification, si celle-ci n'a pas trouvé de chemin préalablement. La détection d'obstacles au sonar (ultrason) permet de gérer les obstacles non vus au Lidar. Dans

ce cas, les cellules correspondantes dans la carte de navigation sont notées comme obstacle. Trois types d'obstacles peuvent être ainsi traités :

- les objets de couleur sombre (noir) ou trop fins pour être vus au Lidar. L'ultrason est un moyen de les détecter avant que cela ne soit fait par le limiteur de couple en cas de choc.
- les cloisons transparentes qui sont marquées comme un mur.
- les miroirs marqués comme un mur mais laissant une pièce fantôme.

Durant l'exploration, les gênes entre robots sont rares car notre algorithme d'affectation favorise la dispersion des robots dans des directions différentes. Toutefois, ces gênes peuvent se présenter, en particulier durant le retour à la zone de départ.

Lors de ces rencontres, le premier robot qui détecte la proximité d'un autre robot, s'arrête et re-calcule sa cible ainsi qu'une trajectoire, laissant ainsi le temps à l'autre robot de continuer sa trajectoire. Les trajectoires planifiées, calculées dans la carte de navigation contenant la position des robots, évitent la position des autres robot. Quand les deux robots se détectent simultanément, les deux robots s'arrêtent et re-calculent leur trajectoire en tenant compte des positions courantes.

Le chapitre suivant décrit les résultats expérimentaux utilisant l'architecture et les algorithmes décrits dans cette thèse.

Chapitre 9

Résultats expérimentaux (défi Carotte)

Sommaire

Dans ce chapitre nous décrivons des expérimentations de notre approches d'exploration multirobot avec des robots MiniRex ; nous détaillons les résultats ayant permis de valider notre approche notamment lors du défi Carotte (2012).

9.1 Défi CAROTTE

Le défi Carotte a été organisé par l'ANR-DGA pendant trois années (2009-2012). Annuellement, une compétition robotique avait lieu. Le but était l'exploration exhaustive d'un environnement inconnu avec des robots autonomes. Chaque édition se déroulait sur une semaine durant laquelle des environnements de tests étaient mis en place et l'environnement final, sur lequel se déroulait la compétition, était révélé le dernier jour. Ainsi, chaque année, les cinq équipes participantes, regroupant chercheurs et industriels, s'affrontaient dans ce cadre défini par les organisateurs.

Un classement des équipes participantes était établi à l'aide d'un système de points évaluant les performances lors de la mission robotique, la validité et l'originalité des approchent proposées (à travers différents rapports et présentations). Pour le calcul des points permettant l'évaluation de la mission robotique les critères suivants étaient pris en compte :

- la qualité des cartes 2D et 3D,
- le rapport entre le nombre de pièces visitées et le nombre de pièces total,
- le rapport entre le nombre de robots revenus et le nombre de robots déployés,
- le rapport entre le temps de la mission et le temps de l'équipe la plus rapide,
- l'évitement des zones dangereuses,
- le nombre et l'exactitude des reconnaissances des classes et sous-classes des objets.

Nous étions en compétition avec quatre équipes ayant chacune adopté une stratégie particulière (approche, nombre de robots...) :

1. **Corebots**, regroupant l'entreprise INTEMPORA, le centre de robotique de l'École des Mines de Paris, l'Inria Rocquencourt et le laboratoire d'électronique d'EPITECH, a réalisé un robot unique mobile autonome intégrant SLAM, vision, contrôle et planification.
2. **PACOM**, regroupant l'entreprise GOSTAÏ, l'École Nationale Supérieure des Techniques avancées (ENSTA) et l'Institut des Systèmes Intelligents et Robotique (ISIR), a utilisé un robot avec des capteurs actifs et panoramiques pour la cartographie sémantique d'objets.
3. **ROBOTS MALINS**, regroupant l'entreprise Thalès Optronique, le Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de Caen (GREYC) et l'Inria Sophia Antipolis, utilise une stratégie à deux robots pour la cartographie et la localisation combiné avec une navigation intelligente.
4. **YOJI** (Yeux, Oreilles, Jambes pour l'Inspection), regroupant les entreprises Aldebaran Robotics, VOXLER et CEA LIST, utilisent des robots humanoïdes NAO.

Notre équipe, nommée **Cart-O-Matic** (CARTographie et localisation d'Objets Multirobots, http//5lair.free.fr/Projects/Cartomatic/), regroupant l'entreprise Wany Robotics, le Laboratoire d'Ingénierie des systèmes automatisés (LISA) et l'équipe Maia du Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA), a participé aux défis avec des robots différents chaque année. En effet, l'entreprise Wany Robotics n'a pas pu mener à terme leur construction, ainsi nous avons utilisé des robots construits par nos partenaires du LISA. Nous avons appliqué les différentes méthodes présentées dans cette thèse et notre collaboration fructueuse avec nos partenaires nous a permis de gagner l'édition finale de ce défi.

9.2 Arène LORIA et mesures

Afin de tester nos méthodes et de nous préparer à la compétition, nous avons construit au LORIA une arène d'exploration de 35 m^2 illustrée par la figure 9.1 en haut. Dans cette section nous présentons le déploiement de l'algorithme *MinPos* avec des robots MiniRex et établissons une première évaluation en situation réelle. Les robots démarrent d'une zone de départ connue qui leur permet de travailler dans le même repère orthonormé.

La figure 9.1 (en bas) montre la carte construite lors d'une exploration avec trois robots utilisant l'algorithme *MinPos* pour l'allocation des frontières. La grille d'occupation utilisée à une résolution de 2 cm^2 et celle où les frontières sont identifiées à une résolution de 8 cm^2. Les cellules frontières contigües sont regroupées en une seule frontière pour limiter le nombre de calculs de propagation de fronts de vague (voir section 7.1). Les chemins parcourus par les robots dans cette expérience montrent une bonne répartition spatiale des robots dans l'environnement, c'est-à-dire une absence de redondance de visite des pièces éloignées du hall d'entrée.

Des mesures de temps d'exploration de cette arène ont été menées avec les algorithmes *MinPos* et *MinDist*. Le graphique donné figure 9.2 montre, comme attendu, que *MinPos* permet une exploration totale de l'environnement plus rapide que *MinDist*. L'arène étant petite pour le nombre de robots utilisés (jusqu'à 4), on constate que les écarts se resserrent dès 3 robots. Le système est par ailleurs impacté par les gênes spatiales engendrées par un nombre croissant de robots. En effet, les robots peuvent être amenés à naviguer à proximité les uns des autres dans certaines portions de l'environnement.

MinPos crée toutefois peu de situations de gêne. En effet, en se répartissant mieux dans l'environnement, les situations de conflit sont évitées. Ainsi, lorsque nous augmentons le nombre de robots, le temps d'exploration varie peu tandis qu'avec *MinDist* les conflits entre les robots

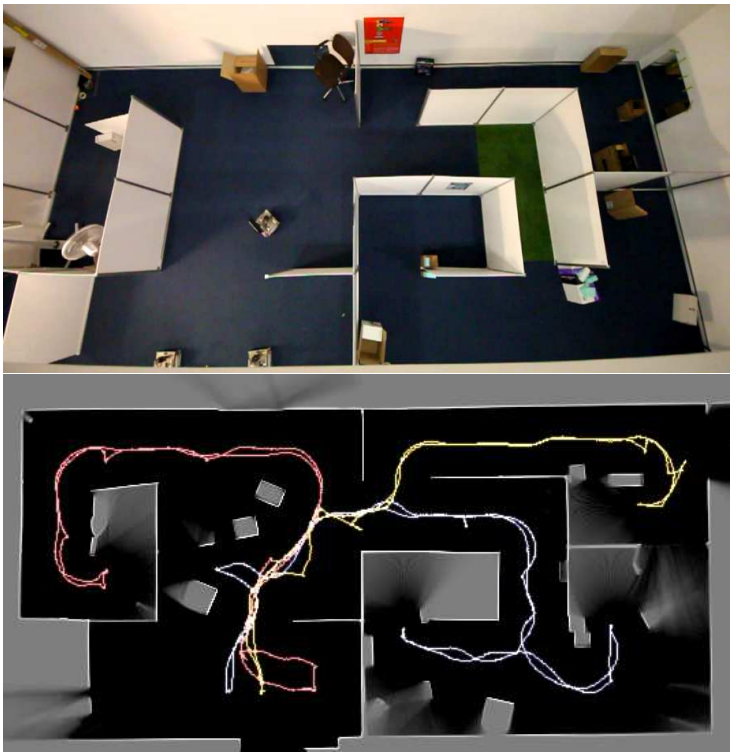


FIGURE 9.1 – Photo de l'arène d'exploration (en haut) et carte et trajectoires résultant de l'exploration de l'arène avec 3 robots où des objets ont été ajoutés (en bas).

entraînent une augmentation du temps d'exploration. Notons que lors de ces expériences le temps mesuré est celui nécessaire aux robots pour explorer l'environnement mais aussi pour retourner à leur point de départ.

9.3 Finale du défi Carotte 2012

Durant la semaine de la dernière édition du défi Carotte à Bourges en 2012, à l'exception de quelques problèmes de locomotion (problèmes mécaniques de déchenillement, robots restés bloqués dans les bacs de terre et de gravier), les robots ont à chaque fois exploré la totalité de l'environnement avant de se replacer à leur position initiale. Ainsi, les résulats de la préparation à l'épreuve finale dans les arènes de tests étaient concluants. Cette implémentation avec des robots MiniRex a montré une grande robustesse de l'approche proposée.

Répartition des robots

Durant la finale de cette édition, nous avons déployé cinq robots pour explorer $120m^2$. Quatre robots sont revenus dans la zone de départ, les cinq trajectoires sont illustrées figure 9.3. Les robots se sont bien répartis dans l'environnement, comme le montre cette figure. Chacune des pièces a été explorée par un seul et unique robot, excepté vers la fin de l'exploration où les robots se sont concentrés pour explorer la dernière pièce (en haut figure 9.3). Ceci est dû au fait que dans l'approche chaque robot a une frontière assignée sans prendre en compte le fait que celle-ci a peut-être déjà été affectée à un autre robot. Nous avons choisi cette stratégie car la taille de l'environnement est inconnue (hypothèse que nous aurions pu relâcher dans Carotte car l'espace

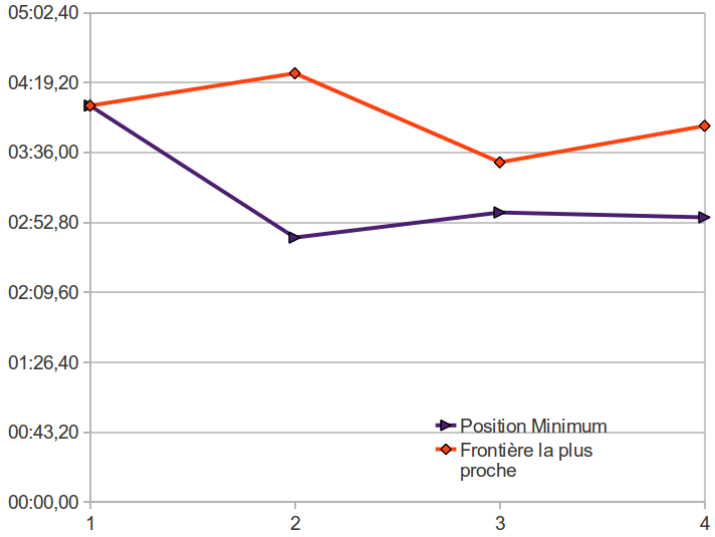


FIGURE 9.2 – Résultat des expériences d'exploration de l'arène en faisant varier le nombre de robots de 1 à 4 (moyenne sur 4 expériences). L'unité en ordonnée est le temps en minutes et secondes et en abscisse le nombre de robots

était borné à $120m^2$). En toue généralité, derrière chaque frontière peut se cacher une largue zone inconnue pouvant être explorée efficacement à l'aide de plusieurs robots. De même si des robots restent bloqués dans des pièges, des robots suiveurs peuvent être utiles pour terminer l'exploration de la pièce.

Robustesse

Un des atouts majeurs de notre approche est son caractère multirobot lui apportant une grande robustesse, en effet, un problème sur un robot ne signifie pas la fin de la mission. Par exemple, un robot ne pouvant plus se déplacer mais continuant à fonctionner peut transmettre aux autres robots des informations sur cette zone dangereuse. Nous avons rencontré ce cas de figure durant la finale du défi. Comme illustré figure 9.4, un robot s'est enlisé dans un bac de terre. Les transmissions effectuées par ce robot ont permis aux quatre autres de l'éviter et d'explorer complètement l'environnement.

Cartographie 3D

L'approche utilisée pour réaliser la cartographie 3D de l'environnement exploré, est basée sur de multiples observations couvrant 360°par rotations (voir section 7.2.1). En plus de leur position, les robots communiquent les positions où ils effectuent ces observations 360°. Ainsi, ils n'effectuent pas de nouvelles observations quand ils se trouvent à moins d'un mètre cinquante d'une précédente observation. Enfin, les robots effectuent une capture unique lorsqu'ils détectent un problème (interruption de la navigation due à un obstacle imprévu). La figure 9.5 illustre les positions des acquisitions Kinect au défi Carotte 2012. L'orientation de l'observation est marquée par un triangle. Les octogones montrent les positions où les observations ont été effectué à 360°. Les triangles isolés sont les positions où une capture unique a été effectuée. Les acquisitions des différents robots sont illustrées avec différentes couleurs. Nous observons une bonne répartition dans l'environnement des positions des acquisitions.

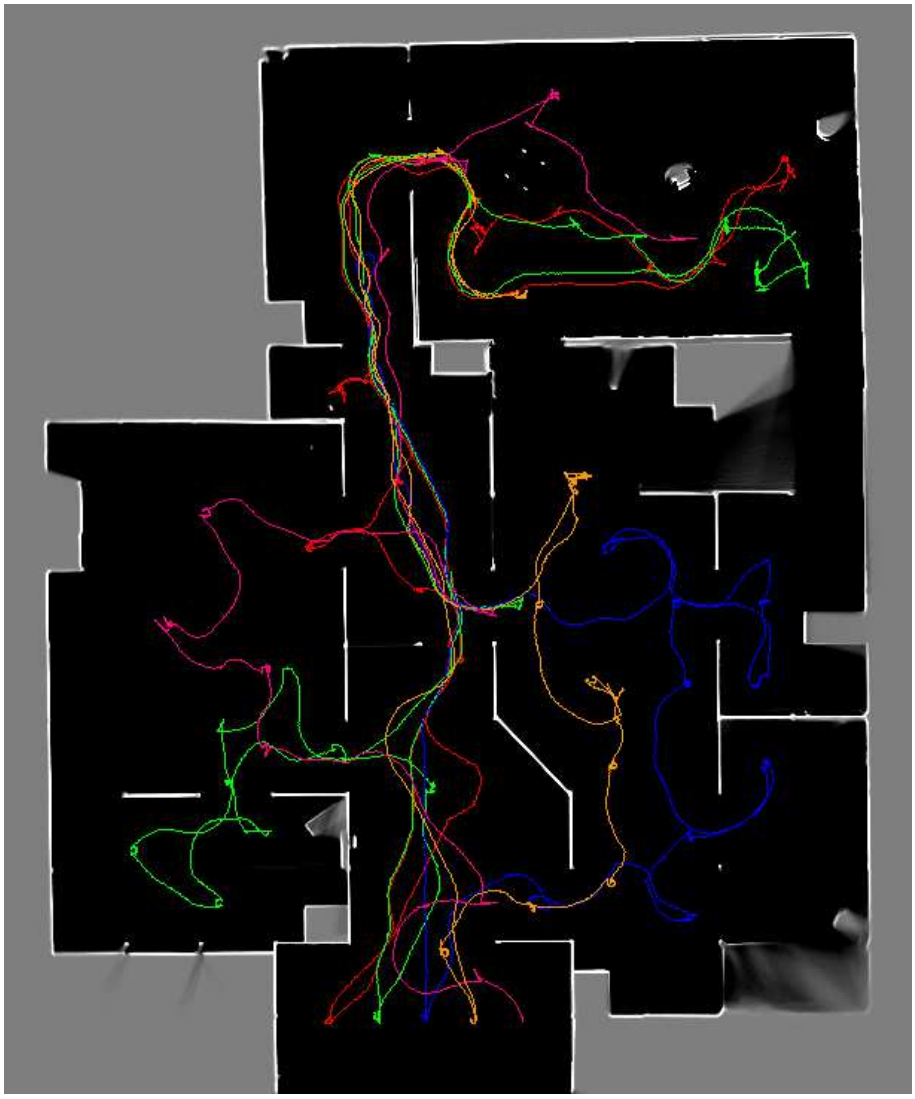


FIGURE 9.3 – Trajectoire des robots lors de la finale du défi Carotte 2012

FIGURE 9.4 – Image prise par la Kinect lors de la finale du défi Carotte 2012, robot enlisé dans un bac de terre

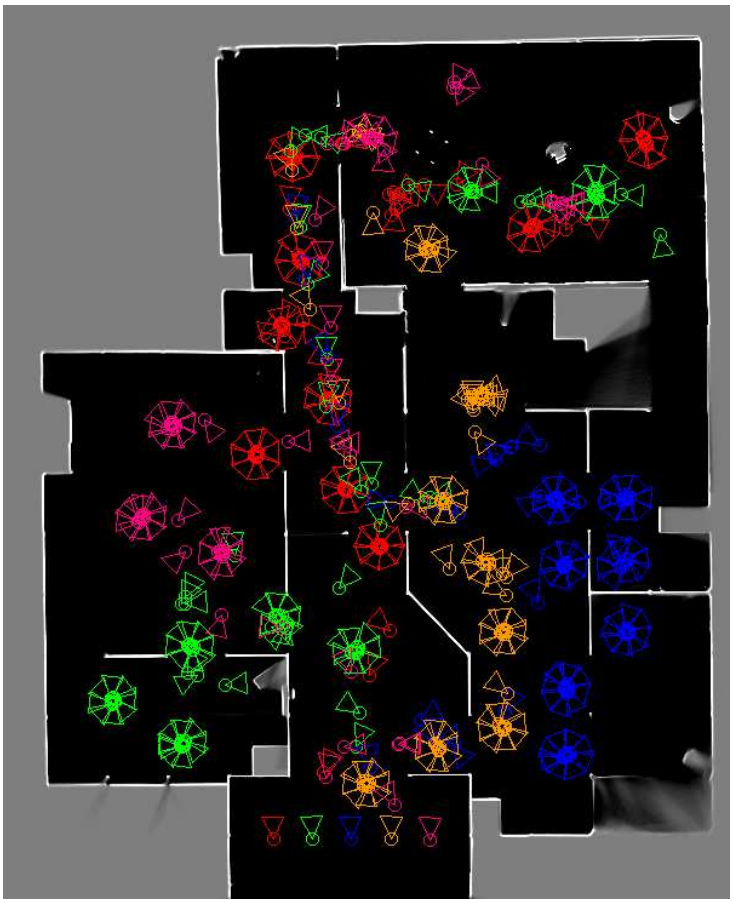


FIGURE 9.5 – Positions des acquisitions Kinect lors de la finale du défi Carotte 2012

La figure 9.6 illustre la carte 3D construite à partir de ces acquisitions. Pour sa construction nous ajoutons le nuage de points 3D obtenu par la Kinect dans une grille 3D dont les voxels ont une taille de $2cm^3$ à partir de la position fournie par le SLAM 2D (x,y,θ) et nous supposons que le robot est à plat sur le sol (z=0). Le roulis et le tangage sont aussi supposés être nuls. L'approche 360° ne garantit pas une exploration 3D complète mais nous pouvons observer que la couverture de l'environnement est presque complète car la carte 3D ne comporte que très peu de "trous" qui correspondent à l'absence d'acquisitions d'une partie de l'environnement. Enfin, l'absence de SLAM 3D, estimant la configuration de la Kinect en 6 dimensions (x,y,z,lacet,roulis,tangage), génère des cellules 3D légèrement non-alignées.

FIGURE 9.6 – Carte 3D construite lors de la finale du défi Carotte 2012

Le cadre applicatif de cette thèse nous a aidé à développer et mettre en pratique des méthodes d'exploration et de cartographie multirobot robustes et efficaces. En effet, les problématiques concrètes rencontrées lors des trois éditions du défi Carotte nous ont permis, avec l'aide de nos partenaires, de développer un système complet d'exploration et de cartographie d'environnements inconnus. Les conditions difficiles imposées par les organisateurs (plexiglas, bac de terre, grilles métalliques...) et renouvelées chaque année, nous ont poussés à anticiper les problèmes en créant des robots capables de naviguer dans de telles conditions. L'approche multirobot que nous avons proposée a permis en particulier de développer un système capable de poursuivre la mission lorsque la mécanique fait défaut sur des robots. Les résultats obtenus lors de la dernière édition du défi, dont nous sommes ressortis lauréats, montrent la validité autant théorique que pratique de nos approches.

Quatrième partie

Conclusion

Chapitre 10

Conclusion

Nous avons abordé dans cette thèse le problème de l'exploration multirobot d'un environnement inconnu. L'exploration consiste à naviguer dans l'environnement selon une stratégie de déploiement afin de le cartographier complètement. C'est un problème riche comportant de nombreux sous-problèmes. En particulier, le problème, fondamental en robotique, de la cartographie et, en intelligence artificielle, celui de la décision des cibles à atteindre par les robots.

Ce chapitre résume les principales contributions de la thèse et décrit les perspectives.

10.1 Synthèse des contributions

Pour le problème de l'affectation aux robots des frontières à visiter, nous avons proposé un **nouveau critère (*MinPos*)** (publié dans (Bautin et al., 2012b)). Celui-ci repose sur la **position** (ou rang) d'un robot vers une frontière. Il s'agit de la position dans la liste ordonnée des robots selon la distance à la frontière considérée.

Les robots sont alors répartis sur les frontières selon leur meilleure position, plutôt qu'en fonction des distances aux frontières comme le font les approches classiques. Cette répartition favorise une bonne distribution des robots vers les différentes directions restant à explorer.

Des mesures de performance de l'affectation minimisant la position, en simulation et sur des robots réels, ont permis de montrer que cette approche est plus efficace en temps d'exploration que l'affectation des robots à leurs frontières la plus proche (*MinDist*). Un gain en performance est aussi obtenu, dans une moindre mesure, sur l'affectation gloutonne mais cette différence dépend de la topologie de l'environnement et du nombre de robots. En conséquence, notre approche est particulièrement intéressante lorsque le nombre de robots est faible en rapport au nombre de frontières potentielles de l'environnement. Autrement, il donne des résultats similaires lorsque le nombre de robots est plus important, tout en étant moins coûteux à calculer.

Pour calculer ce nouveau critère de position, nous avons proposé un **algorithme fondé sur la propagation de fronts de vagues** depuis les frontières (publié dans (Bautin et al., 2012a)). En général, les approches existantes utilisent un algorithme A* pour calculer un chemin de chaque robot vers chaque frontière. La propagation de fronts de vagues depuis les frontières fournit, pour toutes les cellules de l'environnement, les plus courts chemins vers chaque frontière. L'avantage de cette approche est que sa complexité est indépendante du nombre de robots. Elle est donc efficace lorsque le nombre de robots est important. De plus, l'algorithme proposé section 6.2 limite la propagation sur une petite partie de l'environnement, la rendant très efficace en comparaison des approches classiques même si le nombre de robots est faible. Pour cela, la

propagation des fronts est synchronisée sur le nombre de robots rencontrés par chaque front et sur la distance de propagation.

L'analyse de certaines configurations spatiales a montré que l'approche gloutonne donnait une meilleure solution que *MinPos*. Nous avons proposé une **variante de *MinPos* nommée *MPG*** qui consiste à conjuguer *MinPos* avec l'approche gloutonne. Cette variante est un peu plus complexe en calcul que *MinPos* mais permet de combiner les avantages des deux algorithmes. Ses performances sont égales aux meilleures performances des deux algorithmes conjugués indépendamment du nombre de robots et de l'environnement.

Nous nous sommes intéressé à l'impact de la zone de perception des capteurs utilisés par les robots sur l'efficacité des approches par frontière. Suivant la zone de perception des capteurs et la fréquence d'observation, il peut être nécessaire de déterminer les poses ou configurations à atteindre par les robots. Une **stratégie déterministe pour l'exploration avec des capteurs ayant une zone de perception quelconque** a été proposée. La plupart des approches de l'état de l'art sont probabilistes. Elles tentent de maximiser le gain d'information en échantillonnant aléatoirement des configurations et en estimant le gain d'information de chacune (zone inexplorée potentiellement visible depuis la configuration). Notre stratégie consiste à déterminer la pose/configuration permettant l'observation des frontières. Elle est fondée sur la distance aux frontières et la visibilité de la frontière. Elle est simple et fournit de bons résultats.

Afin d'évaluer et d'expérimenter notre approche avec des robots, nous avons proposé une **architecture logicielle** pour l'implémentation de ces algorithmes. Elle repose sur une décision locale à chaque robot, qui décide de manière autonome vers quelle frontière il se dirige. Cela confère au système multirobot une forme de décentralisation. Cette autonomie des robots assure une grande robustesse du système. Les robots communiquent leurs cartes individuelles et leurs positions (coordonnées dans un repère commun) à l'ensemble des autres robots mais les communications utilisent un protocole sans garantie de livraison. Des informations peuvent donc être perdues, notamment lorsqu'un robot est hors de portée du réseau de communication. Au moment de calculer leur affectation, les robots utilisent les informations disponibles à ce moment-là, ainsi ils ne sont jamais bloqués dans l'attente d'une information.

Pour la **planification des trajectoires**, nous avons développé un algorithme A* dans un espace à quatre dimensions (positions, orientation et vitesse), pour lequel nous utilisons une **heuristique originale** : le champ de potentiels précédemment calculé pour l'affectation des frontières. Cette heuristique accélère fortement les calculs puisqu'elle donne une distance au but beaucoup plus précise que la distance euclidienne.

Les expérimentations robotiques ont montré la validité et l'efficacité de l'approche proposée. Nos méthodes ont été **mises en pratique lors du défi robotique Carotte** auquel nous avons participé avec **l'équipe Cart-O-Matic, lauréate de l'édition 2012**.

10.2 Perspectives

Dans cette thèse, nous avons exploré de nouvelles stratégies pour le problème de l'exploration multirobot d'environnements inconnus. Ces travaux soulèvent de nouvelles questions ouvrant des directions de recherche intéressantes. De plus, plusieurs améliorations sont envisageables pour accroître les performances du système robotique développé.

10.2.1 Carte hybride métrique/topologique

Dans l'approche que nous avons proposée, les robots communiquent une carte métrique peu précise de l'environnement, assez légère en mémoire. Toutefois, leur taille dépend de la surface explorée. Pour l'exploration de très grands environnements, il faudra réduire la taille des cartes échangées. Mis à part pour les zones de danger, la résolution des cartes actuelles pourrait être encore réduite. Une perspective intéressante concerne le développement d'une carte hybride topologique et métrique permettant de limiter l'échange d'information concernant l'environnement et de réduire le coût des calculs de distance entre les robots et les frontières. En effet, les cartes topologiques sont très légères et peuvent comporter des informations métriques entre les nœuds. Elles sont peu précises mais elles comportent suffisamment d'information pour l'exploration (la direction des frontières).

Il pourrait donc être intéressant de ne communiquer que des cartes topologiques et les positions des robots limitant ainsi la quantité de données échangées entre les robots. Le calcul des affectations par *MinPos* avec la propagation des vagues en parallèle pourrait être adapté aux cartes topologiques. L'algorithme Dijkstra sur graphe est équivalent à la propagation de vague sur une grille. L'idée serait ici de synchroniser plusieurs algorithmes de type Dijkstra calculés depuis chaque nœud frontière sur le nombre de robots rencontrés et sur les distances.

10.2.2 Localisation et cartographie multirobot

La localisation et la cartographie est gênée par les miroirs pouvant se trouver dans l'environnement : apparition de pièces "fantômes" derrière les miroirs. Nous envisageons d'effectuer le SLAM uniquement sur la pièce en cours d'exploration. L'idée est d'ignorer les rayons en dehors de cette pièce. Cela demandera de détecter "en-ligne" les contours de la pièce.

Actuellement, l'algorithme de SLAM utilisé est déterministe (Lucidarme and Lagrange, 2011). Il existe un risque qu'il tombe dans un minimum local qui ne correspond pas à la configuration du robot. Pour éviter cela et améliorer la robustesse du SLAM, la configuration retournée par l'algorithme de SLAM est vérifiée : si la configuration trouvée est trop éloignée de la configuration précédente, il est relancé à partir de toutes les poses possibles (plutôt que depuis de la dernière configuration connue du robot) et cherche alors la configuration du robot la plus probable globalement.

Une alternative à cette démarche serait d'utiliser une technique consistant à l'initialiser le SLAM depuis un certain nombre de configurations probables. Initialiser le SLAM à partir de configurations différentes peut se faire avec, par exemple, un filtre particulaire. Dans un filtre particulaire, chaque particule représente une trajectoire possible et conserve sa propre carte. Elles mettent à jour leur carte selon leurs croyances. Les particules survivent avec une probabilité proportionnelle à la vraisemblance de l'observation relative à sa carte. Cependant, chaque particule peut occuper beaucoup d'espace mémoire, surtout avec l'utilisation de grilles d'occupation. De plus, la mise à jour des données peut être longue. Il faut donc trouver un compromis entre vitesse de calcul et quantité de données stockées, d'un côté, et qualité de l'estimation, de l'autre. Étant donné que l'algorithme de SLAM actuel est rapide, il est envisageable d'ajouter un filtre particulaire pour augmenter sa robustesse.

10.2.3 Performance

Parmi les critères d'évaluation des performances d'une méthode d'affectation (voir section 3.4), *MinPos* et *MPG* ne considèrent pas celui minimisant le coût maximum des frontières

affectées. Par exemple, lorsque deux robots se dirigent vers deux frontières dans une même direction, l'affectation la plus performante dirigera le robot le plus proche des frontières vers la frontière la plus éloignée. Avec les approches actuelles le robot le plus proche se dirige vers la frontière la plus proche et le robot le plus éloigné vers la frontière la plus éloignée. Dans cette situation, un échange de frontière entre les deux robots est souhaitable pour réduire le coût maximum d'exploration des frontières.

Avec l'algorithme *MPG*, il est possible de détecter à quel moment un échange de frontières serait souhaitable. En effet, lorsqu'un robot dispose d'une frontière pour laquelle il est le seul affecté et que sa position vers cette frontière est supérieure à un, alors le robot et les robots ayant une position inférieure à la sienne ont intérêt à échanger leurs frontières si cet échange n'engendre pas une augmentation de la somme des coûts.

La figure 10.1 illustre un exemple où l'échange de frontières est souhaitable pour améliorer les performances.

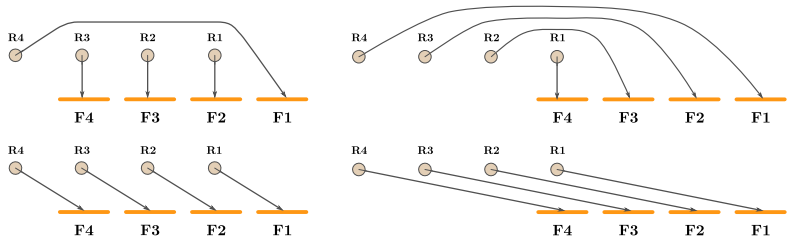


FIGURE 10.1 – Détection des groupes de robots pouvant échanger leur frontière pour améliorer les performances. Le robot $R4$ est en quatrième position vers la $F1$ qui lui est affectée. Un échange de frontière entre les robots ($R1$, en 1^{re} position vers $F1$ $R2$ en 2^{me}, $R3$ en 3^{me} et $R4$) est souhaitable. En haut avant l'échange ; en bas après l'échange les sommes de coûts sont égales mais le coût maximum de l'affectation est inférieur.

10.2.4 Coordination

Lors de l'exploration d'un environnement avec plusieurs robots, certaines trajectoires peuvent "se croiser" et mener les robots à se gêner. Avec l'évitement réactif actuel, il existe quelques situations qui peuvent mener à des ralentissement voire à des blocages dans certains situations particulières, théoriquement possibles, mais non observées en pratique. Par exemple, lors du retour à la zone de départ, les robots peuvent se rencontrer aux intersections.

Pour améliorer la coordination des robots, une approche serait de communiquer les trajectoires qu'ils ont planifiées. Par exemple, Desaraju and How (2012) utilisent une planification de trajectoires multirobot où chaque robot envoie aux autres robots la trajectoire qu'il a planifiée. Les robots planifient l'un après l'autre leur trajectoire respectant une contrainte de distance avec les autres robots le long de la trajectoire. Un système d'enchères est utilisé pour déterminer quel robot effectue la prochaine planification. Mais une telle approche peut être coûteuse en communication. Afin de permettre un passage à l'échelle, nous envisageons une planification des trajectoires décentralisée par **groupe de robots**. Les groupes de robots seraient formés des robots se dirigeant dans la même direction, c'est-à-dire des robots pouvant être potentiellement en conflit. Avec *MinPos* et *MPG*, ces groupes de robots peuvent être identifiés facilement : lorsqu'un robot est assigné à une frontière pour laquelle il n'est pas en première position alors tous les robots plus proches de cette frontière (ayant une position inférieure) font partie d'un

groupe navigant dans la même direction.

10.2.5 Environnements dynamiques

L'application de la méthode proposée à des environnements dynamiques présente plusieurs problèmes pour la cartographie et la navigation. Ce sont des environnements dans lesquels des obstacles mobiles sont présents, comme par exemple, des humains, d'autres robots ne faisant pas partie de la flottille ou des animaux. Ce type d'environnement est plus proche des environnements réels existants.

Pour la cartographie, il est nécessaire de distinguer les objets statiques des objets dynamiques afin de ne pas intégrer les objets dynamiques dans la carte (Dubois et al., 2011). La localisation et la cartographie en environnement dynamique sont des problèmes ouverts. En environnement statique, le principal problème est d'estimer le déplacement du robot en mettant en correspondance les observations du robot. Quand l'environnement est dynamique, cette mise en correspondance est problématique car des parties de l'environnement peuvent s'être déplacées. Dans ce contexte, l'utilisation de plusieurs robots est avantageuse car la détection des objets mobiles est plus facile en utilisant des observations de points de vues différents (Zou and Tan, 2013).

Une fois ces obstacles dynamiques identifiés, il peut être nécessaire d'estimer leurs trajectoires pour avoir une navigation sûre. La planification de trajectoire devra alors prendre en compte les zones dangereuses (où un obstacle dynamique pourra se trouver). Une technique envisageable consiste à interdire dans le A* qui calcule la trajectoire, le développement des nœuds pouvant mener à une collision (Bautin et al., 2010).

10.2.6 Exploration 3D

Il serait intéressant d'étendre ce travail à l'exploration en trois dimensions, problème également d'actualité en robotique. Une approche consisterait à utiliser la recherche des configurations d'observation que nous avons proposée au chapitre 7 (recherche de la pose d'observation de la frontière sur des cercles concentriques autour de la frontière). Son adaptation devrait être assez simple car la recherche des configurations reste identique, seule l'identification des frontières (composées de voxels frontières au lieu de pixels frontières) ainsi que la vérification de visibilité seraient modifiées. Ils s'effectueraient en trois dimensions au lieu de deux.

10.2.7 Hétérogénéité des capacités de déplacement des robots

Une adaptation de l'algorithme *MPG* pour une flottille de robots ayant des capacités de déplacement différentes est envisageable. Si les robots naviguent à des vitesses différentes, il est possible d'adapter la méthode d'affectation pour les prendre en compte. Les vagues fournissant la distance aux frontières, il suffirait de la diviser par la vitesse pour obtenir la durée nécessaire pour atteindre la frontière. Les robots seraient alors affectés selon leur positions dans un classement robot-frontière en fonction du temps et non plus en fonction des distances.

Toutefois cette approche ne permettrait plus d'utiliser la propagation des vagues en parallèle. De plus, elle peut générer des conflits car les robots pourraient avoir des trajectoires se croisant. Son application est donc indissociable du développement d'une planification de trajectoire multirobot.

10.2.8 Autres applications

Enfin, plus généralement, le critère de position proposé pourrait être utile pour l'affectation de tous les types de tâches situées dans un espace métrique. L'exploration multiagent de graphe (par exemple des pages Web) pourrait être une application intéressante.

Bibliographie

Arkin, R. C. (1989). Motor schema—based mobile robot navigation. *The International Journal of Robotics Research 8*(4), 92–112.

Balakirsky, S., S. Carpin, A. Kleiner, M. Lewis, A. Visser, J. Wang, and V. A. Ziparo (2007). Towards heterogeneous robot teams for disaster mitigation : Results and performance metrics from robocup rescue. *Journal of Field Robotics 24*(11-12), 943–967.

Barraquand, J., B. Langlois, and J.-C. Latombe (1991, jun). Numerical potential field techniques for robot path planning. In *Advanced Robotics, 1991. 'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, pp. 1012 –1017 vol.2.

Bautin, A., L. Martinez-Gomez, and T. Fraichard (2010). Inevitable collision states : a probabilistic perspective. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 4022–4027. IEEE.

Bautin, A., O. Simonin, and F. Charpillet (2012a, October). Minpos : A novel frontier allocation algorithm for multi-robot exploration. In *ICIRA'12 Proceedings of the 5th international conference on Intelligent Robotics and Applications*, Volume 7507 of *Lecture Notes in Computer Science*, pp. 496–508. Springer Berlin Heidelberg.

Bautin, A., O. Simonin, and F. Charpillet (2012b, November). Stratégie d'exploration multirobot fondée sur les champs de potentiels artificiels. *Revue d'Intelligence Artificielle 26*(5), 523–542.

Birk, A. and S. Carpin (2006). Merging occupancy grid maps from multiple robots. *Proceedings of the IEEE 94*(7), 1384–1397.

Blanco-Claraco, J.-L. (2009, November). *Contributions to Localization, Mapping and Navigation in Mobile Robotics*. Ph. D. thesis, Universidad de Malaga.

Bresenham, J. (1977). A linear algorithm for incremental digital display of circular arcs. *Communications of the ACM 20*(2), 100–106.

Brooks, R. A. (1985). A robust layered control system for a mobile robot. Technical report, Cambridge, MA, USA.

Burgard, W., M. Moors, and F. Schneider (2002). Collaborative exploration of unknown environments with teams of mobile robots. In M. Beetz, J. Hertzberg, M. Ghallab, and M. Pollack (Eds.), *Plan-Based Control of Robotic Agents*, Volume 2466 of *Lecture Notes in Computer Science*. Springer Verlag.

Burgard, W., M. Moors, C. Stachniss, and F. Schneider (2005, june). Coordinated multi-robot exploration. *Robotics, IEEE Transactions on 21*(3), 376 – 386.

Butzke, J. and M. Likhachev (2011). Planning for multi-robot exploration with multiple objective utility functions. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 3254–3259.

Chaimowicz, L., M. Campos, and V. Kumar (2002). Dynamic role assignment for cooperative robots. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, Volume 1, pp. 293 – 298 vol.1.

Chatila, R. and J. Laumond (1985). Position referencing and consistent world modeling for mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, Volume 2, pp. 138–145. IEEE.

Crowley, J. (1985). Navigation for an intelligent mobile robot. *Robotics and Automation, IEEE Journal of 1*(1), 31–41.

de Hoog, J., S. Cameron, and A. Visser (2010). Selection of rendezvous points for multi-robot exploration in dynamic environments. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Desaraju, V. R. and J. P. How (2012). Decentralized path planning for multi-agent teams with complex constraints. *Autonomous Robots 32*(4), 385–403.

Dornhege, C. and A. Kleiner (2011). A frontier-void-based approach for autonomous exploration in 3d. In *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, pp. 351–356.

Dubois, A., A. Dib, and F. Charpillet (2011). Using hmms for discriminating mobile from static objects in a 3d occupancy grid. In *Proceedings of the 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, ICTAI '11, Washington, DC, USA, pp. 170–176. IEEE Computer Society.

Duckett, T. and U. Nehmzow (1997). Experiments in evidence based localisation for a mobile robot. In *Proceedings of the AISB Workshop on Spatial Reasoning in Mobile Robots and Animals, Manchester, UK, 1997.*

Elfes, A. (1989, Jun). Using occupancy grids for mobile robot perception and navigation. *Computer 22*(6), 46–57.

Faigl, J., M. Kulich, and L. Preucil (2012). Goal assignment using distance cost in multi-robot exploration. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 3741–3746.

Feder, H. J. S., J. J. Leonard, and C. M. Smith (1999). Adaptive mobile robot navigation and mapping. *The International Journal of Robotics Research 18*(7), 650–668.

Ferber, J. (1995). Les systèmes multi-agents, vers une intelligence collective. *InterEditions, Paris*.

Franchi, A., L. Freda, G. Oriolo, and M. Vendittelli (2009, april). The sensor-based random graph method for cooperative robot exploration. *Mechatronics, IEEE/ASME Transactions on 14*(2), 163 –175.

Gerkey, B. P., R. T. Vaughan, and A. Howard (2003). The player/stage project : Tools for multi-robot and distributed sensor systems. In *In Proceedings of the 11th International Conference on Advanced Robotics*, pp. 317–323.

Glad, A., O. Buffet, O. Simonin, and F. Charpillet (2009). Self-organization of patrolling-ant algorithms. In *Self-Adaptive and Self-Organizing Systems, 2009. SASO'09. Third IEEE International Conference on*, pp. 61–70. IEEE.

González-Banos, H. H. and J.-C. Latombe (2002). Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research 21*(10-11), 829–848.

Gossage, M., A. P. New, and C. K. Cheng (2006). Frontier-graph exploration for multi-robot systems in an unknown indoor environment. *Distributed Autonomous Robotic Systems 7*, 51–60.

Holz, D., N. Basilico, F. Amigoni, and S. Behnke (2010). Evaluating the efficiency of frontier-based exploration strategies. In *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pp. 1–8.

Howard, A. (2006). Multi-robot simultaneous localization and mapping using particle filters. *Int. J. Rob. Res. 25*(12), 1243–1256.

Howard, A., M. J. Mataric, and G. S. Sukhatme (2002). Mobile sensor network deployment using potential fields : A distributed, scalable solution to the area coverage problem. In *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems (DARS02)*, pp. 299–308. Citeseer.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research 5*(1), 90–98.

Ko, J., B. Stewart, D. Fox, K. Konolige, and B. Limketkai (2003). A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, Volume 4, pp. 3232–3238. IEEE.

Koenig, S., B. Szymanski, and Y. Liu (2001). Efficient and inefficient ant coverage methods. *Annals of Mathematics and Artificial Intelligence 31*(1), 41–76.

Korf, R. E. (1990). Real-time heuristic search. *Artificial intelligence 42*(2), 189–211.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly 2*(1-2), 83–97.

Kuipers, B. (1978). Modeling spatial knowledge. *Cognitive science 2*(2), 129–153.

Kuipers, B. and Y.-T. Byun (1991). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and autonomous systems 8*(1), 47–63.

Le, V. T. (2010, October). *Coopération dans les systèmes multi-robots : Contribution au maintien de la connectivié et à l'allocation dynamique de rôles.* Ph. D. thesis, Université de Can.

Lozano-Pérez, T. and M. A. Wesley (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM 22*(10), 560–570.

Lucidarme, P. and S. Lagrange (2011, June). Slam-o-matic : Slam algorithm based on global search of local minima. Brevet num. FR1155625.

Makarenko, A., S. Williams, F. Bourgault, and H. Durrant-Whyte (2002). An experiment in integrated exploration. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, Volume 1, pp. 534–539 vol.1.

Mantaras, R. L. D., J. Amat, F. Esteva, M. Lopez, and C. Sierra (1997). Generation of unknown environment maps by cooperative low-cost robots.

Martinelli, A. (2007). Improving the precision on multi robot localization by using a series of filters hierarchically distributed. In *IROS*, San Diego Etats-Unis d'Amérique.

Mei, Y., Y.-H. Lu, C. Lee, and Y. Hu (2006, may). Energy-efficient mobile robot exploration. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 505 –511.

Moravec, H. and A. Elfes (1985). High resolution maps from wide angle sonar. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, Volume 2, pp. 116–121. IEEE.

Puig, D., M. Garcia, and L. Wu (2011). A new global optimization strategy for coordinated multi-robot exploration : Development and comparative evaluation. *Robotics and Autonomous Systems 59*(9), 635–653.

Rekleitis, I., G. Dudek, and E. Milios (2001, May). Multi-robot collaboration for robust exploration. *Annals of Mathematics and Artificial Intelligence 31*(1-4), 7–40.

Rekleitis, I. M., G. Dudek, and E. E. Milios (1997). Multi-robot exploration of an unknown environment, efficiently reducing the odometry error. In *IJCAI'97 : Proceedings of the Fifteenth international joint conference on Artifical intelligence*, San Francisco, CA, USA, pp. 1340–1345. Morgan Kaufmann Publishers Inc.

Renzaglia, A. and A. Martinelli (2009). Distributed coverage control for a multi-robot team in a non-convex environment. In *IEEE IROS 09 3rd Workshop on Planning, Perception and Navigation for Intelligent Vehicles*, St. Louis États-Unis d'Amérique.

Roy, N. and G. Dudek (2001, September). Collaborative robot exploration and rendezvous : Algorithms, performance bounds and observations. *Auton. Robots 11*(2), 117–136.

Simmons, R., D. Apfelbaum, W. Burgard, M. Fox, D. an Moors, S. Thrun, and H. Younes (2000). Coordination for multi-robot exploration and mapping. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Austin, TX. AAAI.

Smith, R. C. and P. Cheeseman (1986). On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research 5*(4), 56–68.

Stachniss, C., D. Hahnel, W. Burgard, and G. Grisetti (2005). On actively closing loops in grid-based fastslam. *ADVANCED ROBOTICS 19*, 2005.

Stachniss, C., O. Martinez Mozos, and W. Burgard (2008). Efficient exploration of unknown indoor environments using a team of mobile robots. *Annals of Mathematics and Artificial Intelligence 52*(2-4), 205–227.

Strand, R. and N. Normand (2012). Distance transform computation for digital distance functions. *Theoretical Computer Science 448*(0), 80 – 93.

Vazquez, J. and C. Malcolm (2004). Distributed multirobot exploration maintaining a mobile network. In *Intelligent Systems, 2004. Proceedings. 2004 2nd International IEEE Conference*, Volume 3, pp. 113–118. IEEE.

Wagner, I. A., M. Lindenbaum, and A. M. Bruckstein (1999). Distributed covering by ant-robots using evaporating traces. *IEEE Transactions on Robotics and Automation 15*, 918–933.

Wurm, K. M., C. Stachniss, and W. Burgard (2008). Coordinated multi-robot exploration using a segmentation of the environment. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 1160–1165. IEEE.

Yamauchi, B. (1997, jul). A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., IEEE International Symposium on*, Washington, DC, USA, pp. 146 –151. IEEE Computer Society.

Yamauchi, B. (1998). Frontier-based exploration using multiple robots. In *AGENTS '98 : Proceedings of the second international conference on Autonomous agents*, New York, NY, USA, pp. 47–53. ACM.

Yguel, M., O. Aycard, and C. Laugier (2006). Efficient gpu-based construction of occupancy girds using several laser range-finders. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 105–110.

Zlot, R., A. Stentz, M. Dias, and S. Thayer (2002). Multi-robot exploration controlled by a market economy. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, Volume 3, pp. 3016 –3023.

Zou, D. and P. Tan (2013). Coslam : Collaborative visual slam in dynamic environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*(2), 354–366.

Table des figures