UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et de génie informatique

PLANIFICATION DE CHEMINS POUR ROBOT MOBILE EXPLORATEUR DE PLANÈTE

Mémoire de maîtrise
Spécialité : génie électrique

David GINGRAS

Jury : Jean DE LAFONTAINE, ing., Ph.D.
Guy PAYRE, Ph.D.
Érick DUPUIS, ing., Ph.D.

Sherbrooke (Québec) Canada Août 2010

Your file Votre référence
ISBN: 978-0-494-70760-9
Our file Notre référence
ISBN: 978-0-494-70760-9

À mon père Michel, à qui je dois mon esprit logique et cartésien.

RÉSUMÉ

L'intelligence artificielle implantée sur les robots mobiles explorateurs de planète (*rovers*) a une incidence directe sur la distance que peuvent parcourir ces robots. Dans un futur proche, les *rovers* devront parcourir de plus longues distances qu'ils ne le font actuellement. Pour cela, une partie de la solution consiste à changer les systèmes de vision stéréo passifs actuels par un système laser (*LIDAR*) permettant ainsi aux robots de voir plus loin et plus précisément. Cette modification amène en contrepartie une quantité énorme de données topographiques à traiter. Comme les ordinateurs embarqués sur les robots spatiaux sont généralement limités en capacité de calcul et en mémoire, les données obtenues du capteur doivent être compressées. Cette compression peut conduire à une représentation irrégulière de l'environnement qui implique à son tour de nombreuses complications au niveau des algorithmes de génération de chemin. Comme les robots auront une façon différente de comprendre leur environnement, ils devront utiliser une nouvelle approche pour naviguer et planifier des chemins.

Ce projet de recherche vise développer une méthode de planification de chemin capable d'opérer dans une représentation irrégulière de l'environnement. L'algorithme développé doit générer des chemins qui atteignent les destinations fixées par un algorithme de navigation de plus haut niveau. Ce chemin doit être : sécuritaire, continu, lisse, court, prendre en compte les contraintes mécaniques du robot et engendrer une faible consommation énergétique. L'algorithme quant à lui doit être rapide, robuste et ne doit pas surcontraindre ce problème d'optimisation. Pour atteindre ces objectifs, le candidat propose l'utilisation d'une approche élégante basée sur une analogie à la mécanique des fluides.

L'idée générale est d'utiliser l'environnement du robot comme un bassin de fluide sans viscosité. Dans ce bassin, il entre une quantité constante de fluide à la position initiale du robot et ce même débit ressort à la destination à atteindre. La résolution de l'écoulement stabilisé permet de tracer des lignes de courant qui s'avèrent de bons chemins candidats. Enfin, c'est au moyen de certains critères d'optimisation que le meilleur chemin parmi l'ensemble des lignes de courant est sélectionné. Afin de confronter cette méthode à l'état de l'art, une rigoureuse comparaison a été effectuée avec l'approche de recherche de graphe $A^{\star}$. Cette dernière est largement utilisée depuis des décennies par l'industrie et certaines agences spatiales. Cette comparaison a permis de mettre en lumière les avantages et inconvénients des deux méthodes et a conduit à la fusion des deux afin d'obtenir une approche hybride. Celle-ci permet de faire ressortir les avantages des deux méthodes individuelles et d'atténuer les inconvénients.

Afin de valider la performance des méthodes et de confirmer l'atteinte des objectifs, les algorithmes ont été mis à l'épreuve sur une vaste banque de données de terrains réels mesurés. Sur ces centaines de terrains, les algorithmes ont planifié des chemins générant ainsi de nombreux résultats expérimentaux. Une analyse des résultats a permis de conclure à l'atteinte de l'ensemble des objectifs du projet. Les méthodes proposées ont de plus été implantées avec succès sur un banc d'essai robotisé de l'Agence spatiale canadienne.

Le résultat ultime du projet est une démonstration d'envergure de l'autonomie du robot et donc par le fait même de la fonctionnalité de l'algorithme de génération de chemin. Dans cette démonstration appelée *Avatar Explore*, les positions à atteindre par le robot proviennent d'un signal de la Station spatiale internationale. L'expérience a eu lieu au courant de l'été et de l'automne 2009 avec aux commandes l'astronaute canadien Robert Thirsk. Celui-ci a envoyé de nombreuses consignes à un robot qui utilisait le fruit de ce projet de recherche.

Mots-clés : robotique mobile, planification de chemins, exploration spatiale, équation de Poisson, méthode des éléments finis, mécanique des fluides, écoulement potentiel

REMERCIEMENTS

Je tiens à remercier le professeur de mathématique Guy Payre pour ses influences constructives sur le projet. C'est lui qui a proposé initialement l'idée de résoudre un écoulement potentiel sur un maillage dans le but de générer des chemins. Je lui dois aussi l'algorithme de résolution utilisant la méthode des éléments finis qu'il m'a remis sous forme d'un code en langage *Fortran*.

L'Agence spatiale canadienne est un lieu de recherche privilégié. Cela s'explique par la présence de professionnels aussi compétents que diversifiés dans ce ministère fédéral pas comme les autres. Merci à Sébastien Gemme pour son efficacité à régler mes problèmes récurrents reliés aux systèmes *Linux*. Merci à Pierre Allard pour son aide considérable lors de l'implantation de mes travaux sur le banc d'essai. Merci à Alessio Salerno pour sa rigueur irréprochable lors des nombreux essais expérimentaux, mais aussi pour ses idées bien souvent fort divergentes des miennes qui m'ont ouvert les yeux sur d'autres points de vue. Je remercie spécialement mon prédécesseur Jean-Luc Bedwani qui m'a bien encadré et transmis son savoir en début de projet. Il m'a appris tout ce que je connais sur le robot sur lequel j'ai tant travaillé au cours des deux dernières années.

Enfin, je transmets un merci sincère à Érick Dupuis, chercheur et gestionnaire à l'Agence spatiale canadienne, sans qui je n'aurais pu recevoir l'honneur de travailler pour mon pays. En plus de bien me diriger au besoin, il m'a laissé la liberté et l'autonomie nécessaires à l'émergence d'idées novatrices. Par ailleur, Érick m'a permis d'étudier de nombreux aspects de la robotique externe au projet de maîtrise tels que la localisation visuelle, le contrôle de position, la compression de données et j'en passe. Grâce à lui, j'ai eu la chance de rencontrer plusieurs chercheurs et ingénieurs influents. Ainsi, j'amorce ma carrière en robotique avec un savoir et une expérience significatifs, de même qu'avec un réseau de contacts appréciable au gouvernement et dans l'industrie.

Je remercie finalement mon directeur accadémique, M. Jean de Lafontaine, qui m'a généreusement accueilli dans son groupe recherche à l'Université de Sherbrooke.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ACRONYMES

2D	Dimension 2
2.5D	Dimension 2.5
3D	Dimension 3
A*	Algorithme de recherche de graphe *A Étoile*
ASC	Agence spatiale canadienne
AutoNav	Autonomous navigation with hazard avoidance
CMU	Carnegie Mellon University
DSN	Deep Space Network
GESTALT	Grid-based Estimation of Surface Traversability Applied to Local Terrain
IMP	Imager for Pathfinder
ISS	Station spatiale internationale
JPL	Jet Propulsion Laboratory
LIDAR	Light detection and ranging
NASA	National Aeronautics and Space Administration
MBS	Mobile Base System
MEF	Méthode des éléments finis
MER	Mars Exploration Rovers
MSL	Mars Science Laboratory
MSS	Mobile Servicing System
SPDM	Special Purpose Dextrous Manipulator
SSRMS	Space Station Remote Manipulator System

CHAPITRE 1

INTRODUCTION

PAR l'entremise de l'Agence spatiale canadienne (ASC), le Canada a investi au cours des vingt-cinq dernières années plus de 1.5 milliard de dollars dans le secteur de la robotique spatiale [Lange *et al.*, 2004]. Au-delà de la formation d'une équipe d'astronautes qualifiés, l'agence fédérale a fourni de remarquables contributions au programme de la Station spatiale internationale (ISS). La principale contribution canadienne est l'implantation du programme *Mobile Servicing System* (MSS) dont les trois grands projets visent à assurer la maintenance de la station en orbite. Le premier de ces projets, le *Space Station Remote Manipulator System* (SSRMS), a donné naissance aux fameux bras robotisés canadiens (*Canadarm II*). Le second projet s'intitule le *Special Purpose Dextrous Manipulator* (SPDM) et concerne un robot qui vient s'arrimer à l'extrémité du SSRMS. Enfin, le dernier des projets constituant le programme est le *Mobile Base System* (MBS). Son but est de permettre l'ancrage mobile qui assure l'alimentation et la communication au SSRMS et au SPDM [Sallaberger et Force, 1997]. Ces projets d'envergure témoignent de la grande expertise canadienne en robotique spatiale.

À l'heure actuelle, le savoir-faire spatial canadien s'oriente certainement vers l'exploration planétaire [Béland *et al.*, 2000]. L'intérêt que porte le Canada pour ce type d'exploration s'est concrétisé en mai 2008 alors que la sonde d'exploration *Phoenix* posée en sol martien a manoeuvré de l'instrumentation scientifique conçue et fabriquée au Canada [Daly *et al.*, 2004]. Depuis, l'envoi d'une sonde robotisée entièrement développée par le Canada demeure une vision fort attrayante partagée par plusieurs. C'est dans cet esprit que le groupe de recherche et de développement en robotique de l'ASC poursuit le développement de technologies de navigation autonome vouées à des robots mobiles explorateurs de planètes [Dupuis *et al.*, 2004; Rekleitis *et al.*, 2009, 2008b].

La dernière décennie a été l'hôtesse de brillants succès d'exploration de Mars au moyen de véhicules mobiles robotisés (*rovers*). En 1997, la mission *Pathfinder* a déposé en sol martien le robot *Sojourner* qui a parcouru environ 100 m en 83 sols[1]. Puis, en 2003 et 2004, la mission *Mars Exploration Rovers* (MER) prend la relève et dépose les *rovers Spirit* et *Opportunity*. Toujours en opération, ces robots parcourent une distance jour-

[1]Le sol est une journée martienne qui dure 24h 37m 23s en terme d'heure terrestre.

nalière moyenne d'une dizaine de mètres. Les faibles distances parcourues sont dues à l'intervention de la base terrestre dans les prises de décision, car les capacités d'autonomie de ces robots sont limitées.

Une importante contrainte associée à une mission sur Mars provient de la faiblesse du lien de communication qu'il est possible d'établir entre Mars et la Terre au moyen du réseau d'antennes *Deep Space Network* (DSN) de la *National Aeronautics and Space Administration* (NASA) [M.S., Gatti, 2003]. Les délais de communication peuvent atteindre jusqu'à 26 minutes et la largeur de bande passante est restreinte. Cette limitation exclut donc la possibilité d'une téléopération efficace et sécuritaire d'un robot sur Mars à partir de la Terre [Carsten *et al.*, 2007].

Pour que les retombées scientifiques de l'exploration en justifient l'investissement, les futurs robots explorateurs planétaires devront être munis de capacités d'autonomie qui requièrent une intervention minimale de la Terre. Une part importante de l'autonomie est supportée par la capacité du robot à planifier lui-même sa trajectoire entre une position initiale et une destination. La responsabilité revient maintenant au robot de générer des chemins sécuritaires et optimaux. Pour accomplir une telle tâche, il doit observer son environnement, repérer les obstacles et comprendre ce qui l'entoure. Pour ce faire, le robot est équipé d'un système de vision dédié à la navigation. C'est au moyen de l'information issue de ses capteurs qu'il peut construire une représentation de son environnement.

Présentement, la technologie qui gouverne les robots est l'imagerie stéréo passive. Cette technologie dont la précision est limitée s'avère en plus peu fiable à l'analyse des cratères et des pentes descendantes [Henriksen et Krotkov, 1997]. Aussi, ce capteur n'a généralement qu'une portée de quelques mètres. Dans le but d'accroître l'autonomie, les prochains *rovers* pourraient être équipés de système de vision actif de type *LIDAR*. Contrairement à l'imagerie passive, les caméras qui le constituent ne sont pas sujettes à l'influence de la luminosité et ont une portée beaucoup plus importante [Rekleitis *et al.*, 2009]. Toutefois, la quantité de données retournées par le capteur actif est si importante qu'une compression est nécessaire. Cette compression peut amener une représentation du terrain irrégulière, complexifiant conséquemment la génération de chemins. À ce propos, le candidat propose d'investiguer la question de recherche suivante : «*Comment un robot peut-il, de façon autonome, générer un chemin sécuritaire et optimal*[2]

[2]Le qualificatif «*optimal*» n'est pas utilisé ici au sens d'une solution mathématiquement optimale. La section 3 explique le terme «*chemin optimal*» qui provient d'un processus approximatif d'optimisation.

entre une position initiale et une destination connue dans une représentation irrégulière de son environnement ?»

Le présent document étale en premier lieu une revue de littérature sur les méthodes de planification de chemin adaptées aux robots mobiles afin de comprendre l'état de la question. Ensuite, la définition formelle du projet de même que les objectifs de recherche sont présentés. Le chapitre 4 expose en détail la stratégie permettant d'obtenir une modélisation de terrain à partir de données brutes *LIDAR*. Puis, le chapitre 5 attaque le coeur du projet en exposant la méthode de planification de chemin mise en oeuvre. Ensuite, plusieurs validations expérimentales sur banc d'essai sont présentées au chapitre 6. Finalement, une conclusion termine l'ouvrage.

CHAPITRE 2

ÉTAT DE L'ART

2.1 Définition de termes

2.1.1 L'espace des configurations de Latombe

Le présent document reprend la notation et les termes présentés dans le livre *Robot motion planning* [Latombe, 1991], ouvrage de référence dans le domaine. L'uniformisation des termes techniques présentés ici s'étend aujourd'hui à la plupart des textes traitant de planification de mouvement de robot. Des livres récents tels que *Autonomous Mobile Robots : Sensing, Control, Decision-making, and Applications* [Ge, S.S. et Lewis, F.L., 2006] ainsi que *Planning algorithms* [LaValle, 2006] et bien d'autres se sont pliés à cette notation. C'est dans cet esprit d'homogénéité que l'auteur de ce mémoire présente les termes qui suivent.

Configuration d'un robot (*robot configuration*) La configuration q est un ensemble de variables qui, à eux seuls caractérisent complètement la position et l'orientation de tous les points composant le robot. Par exemple, Ge et Lewis [Ge, S.S. et Lewis, F.L., 2006] expriment la configuration d'un robot manipulateur par la liste des angles de chacune de ses articulations.

Espace des configurations (*configuration space*) Supposons qu'un robot possède d variables caractérisant sa configuration. À partir de là, le robot peut être considéré comme un point dans un espace $\mathcal{C}$ de dimension d appelé espace des configurations.

Espace libre (*free space*) Une configuration q est dite libre si le robot placé à q n'entre pas en collision avec un obstacle (ou avec lui-même dans le cas d'un bras manipulateur). L'espace libre $\mathcal{C}_{free}$ est donc l'ensemble de toutes les configurations libres de $\mathcal{C}$.

Espace des obstacles (*obstacles space*) L'espace des obstacles $\mathcal{C}_{obs}$ est un sous-espace de $\mathcal{C}$ qui comprend l'ensemble des configurations menant sur un obstacle. Donc l'espace des configurations s'exprime ainsi :

$$\mathcal{C} = \mathcal{C}_{free} \cup \mathcal{C}_{obs}$$

Chemin (*path*) Un chemin entre la configuration initiale q_{init} vers une configuration souhaitée q_{goal} est une fonction continue τ telle que :

$$\tau : [0, 1] \rightarrow \mathcal{C},$$

avec $\tau(0) = q_{init}$ et $\tau(1) = q_{goal}$.

Trajectoire (*trajectory*) Lorsque τ est exprimé comme une fonction du temps, il s'agit alors d'une trajectoire.

Holonomie (*holonomy*) Un système robotisé est dit holonome si et seulement si son mouvement est contraint par un système d'équations algébriques uniquement fonction des degrés de liberté du robot et du temps [Nakamura et Mukherjee, 1991].

Contrainte non holonome (*nonholonomic constraint*) Une contrainte est dite non holonome s'il est impossible de l'écrire comme une contrainte algébrique dans l'espace des configurations [Murray et Sastry, 1993]. L'exemple typique d'un système possédant une contrainte non holonome est une automobile. À l'heure actuelle, ce véhicule est mécaniquement incapable d'effectuer une rotation sans changer sa position.

2.1.2 Représentation de l'environnement d'un robot

Maillage triangulaire (*triangular mesh*) Selon l'article *Delaunay Triangulation and Meshing : Application to Finite Elements* [George et Borouchaki, 1998], le maillage triangulaire d'un nuage de points dans $\Re^d$ (avec $d \geq 2$) fait correspondre l'enveloppe convexe de ces points avec un ensemble de cellules assemblées qui, en général, simplifie la réalité. Généralement pour $d = 2$, les cellules sont des triangles et pour $d = 3$ des tétraèdres. Sur chacune des cellules des propriétés sont satisfaites. Par exemple, la continuité du domaine ou la continuité d'une fonction dérivée d'une cellule à l'autre.

Cellule (*cell*) Toujours selon Latombe, l'élément unitaire d'un maillage de C_{free} s'appelle cellule.

2.5D (*two-and-a-half-dimensional*) Le terme 2.5D ou pseudo-3D réfère, dans le cas de la représentation de l'environnement d'un robot explorateur de planète, a un terrain pouvant être décrit par une fonction mathématique de la forme $z = f(x, y)$. Pour chaque coordonnée (x, y), il ne peut exister qu'une altitude z à cette représentation. Par exemple, la représentation d'une grotte par un maillage triangulaire possédant à la fois le plancher et le plafond de la grotte n'est pas 2.5D.

2.1.3 Algorithme

Complétude (*completeness*) Selon Ge et Lewis [Ge, S.S. et Lewis, F.L., 2006], dans le contexte d'une recherche de chemin, un algorithme est dit complet, s'il retourne soit un chemin quand la solution existe, soit un message d'inexistence.

Complexité (*complexity*) La définition qui suit est tirée du livre *Mastering algorithms with C* [Loudon, 2000]. La complexité d'un algorithme est la courbe de croissance des ressources qu'il requiert par rapport au volume de données qu'il traite. La notation O est celle qui est le plus communément utilisée pour exprimer formellement les performances d'un algorithme. Elles sont exprimées sous la forme d'une fonction de la taille des données. Cela signifie que, pour une taille n , la performance est décrite à l'aide d'une fonction $f(n)$. Cependant, s'il est possible de déterminer exactement f, il n'est habituellement pas nécessaire d'être aussi précis : on ne se préoccupe uniquement de la courbe de croissance de f, qui décrit la rapidité avec laquelle les performances d'un algorithme se dégradent à mesure que la taille des données traitées augmente.

2.2 Méthodes de planification de chemin

La recherche de chemin pour robot dans un environnement statique a été étudiée intensivement à travers les dernières quatre décennies. De nombreuses méthodes ont vu le jour. Certaines dédiées aux bras manipulateurs, d'autres aux robots mobiles et certaines compatibles avec tous systèmes robotisés. Ultimement, toutes les méthodes tentent de résoudre un problème unique. En reprenant la nomenclature de Latombe,

cette problématique de recherche de chemin peut se résumer ainsi :

> *La recherche d'un chemin τ dans l'espace C_{free} qui relie les configurations q_{init} à q_{goal} s'appelle planification de chemin.*

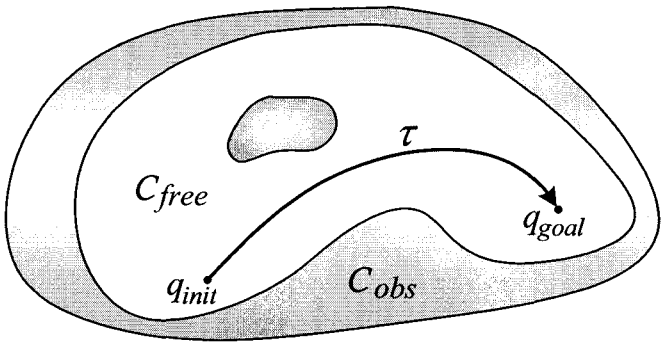


Figure 2.1 Schéma du problème de base de la planification de chemin d'un robot en reprenant la terminologie de Latombe. Les secteurs ombragés représentent l'espace des obstacles et en blanc est illustré l'espace libre.

La figure 2.1 permet d'imager la simplicité d'une représentation par configuration. Cette approche permet de ramener le problème à la recherche d'un chemin pour un point mobile dans C_{free}. Cette formulation ne change en rien la complexité du problème, mais il est plus simple conceptuellement de bouger un point que de déplacer un robot réel pourvu de dimensions et de contraintes physiques. Dans le même ordre d'idée, Lozano-Pérez, père de la formulation en espace des configurations[1], a proposé dans plusieurs articles [Lozano-Perez, 1983; Lozano-Perez et Wesley, 1979] de grossir les obstacles afin d'omettre les dimensions du robot dans l'analyse. D'autres auteurs tels que Spenko, Iagnemma et Dubowsky [Spenko *et al.*, 2004] ont développé des méthodes oú les dimensions du robot et les contraintes mécaniques sont prises en compte.

La section 2.2 présente les familles de méthodes de planification de chemin. Pour chacune d'elle, des exemples concrets d'algorithmes et d'applications réelles sont exposés ce qui permet d'appuyer les explications. D'abord, quelques méthodes *Bug-like* inspirées du comportement des insectes sont présentées. Ensuite viennent les approches *roadmap* et de décomposition cellulaire. Enfin, la méthode du champ de potentiel est exposée au moyen de comparaisons avec le comportement des charges électriques et de la mécanique des écoulements de fluide.

[1]Historiquement, Thomas Lozano-Pérez fut le premier à présenter le problème sous l'angle de l'espace des configurations. Jean-Claude Latombe est celui qui a uniformisé l'approche.

2.2.1 Algorithmes *Bug-like*

La première catégorie d'algorithme présentée s'apparente davantage à une stratégie d'évitement d'obstacle que de génération de chemin. L'algorithme dirige le robot en ligne droite vers la cible. Si au passage, celui-ci rencontre un obstacle, il l'évite en suivant son contour. Ce comportement, relativement limité d'un point de vue intelligence artificielle, rappelle le mouvement de certains insectes. La figure 2.2 illustre un exemple de génération de chemin basé sur l'algorithme *Bug2*.

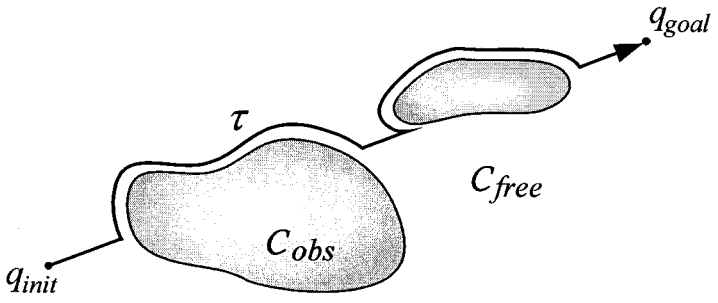


Figure 2.2 Exemple d'un chemin généré par l'algorithme *Bug2*

L'article de Lumelsky et Stepanov [Lumelsky et Stepanov, 1987] explique en détail les méthodes *Bug1* et *Bug2*. Il montre aussi comment un robot piloté par ces algorithmes peut s'échapper d'un labyrinthe. De nombreuses variantes de ces méthodes ont vu le jour. Mentionnons ici l'algorithme *Wedgebug* [Laubach et Burdick, 1999] dont l'efficacité a été démontrée en terrain réel sur un *rover* au centre de recherche de la NASA Jet Propulsion Laboratory (JPL).

Ces approches relativement simples à implanter s'avèrent toutefois limitées en performance. C'est le caractère local de la recherche de chemin qui rend la méthode vulnérable. Effectivement, les méthodes locales peuvent retourner des chemins peu efficaces, car elles ne considèrent pas la totalité du domaine navigable. Elles ne peuvent pas anticiper les obstacles, elles ne font que réagir.

2.2.2 Méthode *roadmap*

Selon le livre *The CRC Robotics Handbook of Mechanical Engineering* [Kreith et Goswami, 1999], les méthodes *roadmap* sont parmi les approches les plus anciennes de planification de chemin. Les auteurs Ge et Lewis définissent le *roadmap* comme une simplification de l'espace libre de l'environnement du robot au moyen d'un réseau de courbes à une dimension [Ge, S.S. et Lewis, F.L., 2006].

Une fois le réseau construit, le robot est contraint à se déplacer uniquement suivant ces courbes. La difficulté de cette méthode réside dans la construction d'un réseau de courbes qui permet au robot de se déplacer en tout point de l'espace libre, tout en limitant le nombre de courbes. Les algorithmes appartenant à cette famille de méthodes sont généralement complets[2]. Les méthodes du *graphique de visibilité* et du *diagramme de Voronoi* qui suivent sont des exemples de méthode *roadmap*.

Graphique de visibilité La méthode par graphique de visibilité a été introduite par Nilsson dans son article *A Mobile Automaton : An Application of Artificial Intelligence Techniques* [Nilsson, 1969]. Le livre *Introduction to Autonomous Mobile Robots* [Siegwart et Nourbakhsh, 2004] explique que ce graphique est formé des lignes droites qui joignent l'ensemble des sommets des obstacles dans une représentation polygonale de l'espace des configurations. Tous les sommets qui sont visibles des autres sommets sont reliés entre eux. Une fois le graphique de visibilité formé, le planificateur de chemin choisi le chemin le plus court entre la position initiale et la cible dans le *roadmap*. La figure 2.3 montre un exemple de chemin généré au moyen de cette méthode.

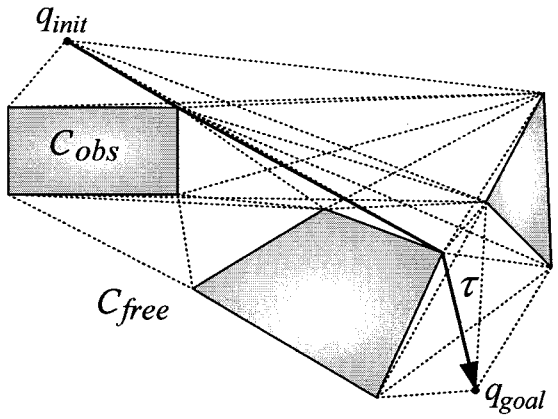


Figure 2.3 Exemple d'un chemin généré au moyen de la méthode de graphique de visibilité

Les courbes pointillées représentent l'ensemble du *roadmap* et la courbe en gras est le chemin sélectionné par l'algorithme. Les solutions retenues par cette méthode tendent toujours à se déplacer le plus près possible des obstacles ce qui résulte en des chemins courts, mais potentiellement dangereux pour le robot. Selon le livre *Computational Geometry : Algorithms and Applications* [De Berg *et al.*, 2008], la charge de calcul

[2]Voir section 2.1 pour connaître la définition de la complétude d'un algorithme.

d'un graphique de visibilité est d'ordre $O(n^2 \log n)$ où n est le nombre total de sommets d'obstacle. C'est donc dire que pour un environnement où la densité d'obstacle est élevée, le graphique peut s'avérer long à calculer. Des méthodes dérivées existent de complexité semblable, applicable sur des obstacles qui ne sont pas nécessairement polygonaux. C'est le cas de la méthode *Visibility complex* présentée dans l'article *The visibility complex* [Pocchiola et Vegter, 1993].

Diagramme de Voronoi Contrairement à l'approche du graphique de visibilité, l'usage du diagramme de Voronoi est une méthode *roadmap* qui maximise la distance entre le robot et les obstacles. L'espace des configurations est décomposé en réseau de courbes formé par le diagramme de Voronoi. Les détails de la construction d'un tel diagramme sont expliqués dans l'article *Voronoi diagrams–a survey of a fundamental geometric data structure* [Aurenhammer, 1991]. La figure 2.4 montre un exemple de cette méthode.

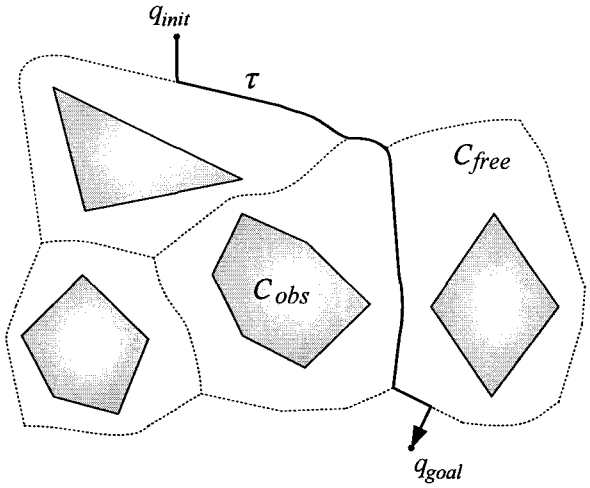


Figure 2.4 Exemple d'un chemin généré au moyen de la méthode du diagramme de Voronoi

Selon l'article *A retroaction method for planning the motion of a disc* [O'Dunlaing et Yap, 1985], cette approche est de complexité $O(n \log n)$. Elle requiert donc moins de ressource informatique que la méthode du graphique de visibilité. Aussi, elle génère des chemins plus sécuritaires, mais moins optimaux que dans le cas de cette dernière méthode. Des chercheurs de l'université Carnegie Mellon (CMU) ont présenté dans l'article *Incremental Reconstruction of Generalized Voronoi Diagrams on Grids* [Kalra *et al.*, 2006], un véhicule agricole robotisé dont le coeur de son générateur de chemin utilise une approche par diagramme de Voronoi.

L'article *The visibility–Voronoi complex and its applications* [Wein *et al.*, 2007] propose une approche hybride. Cet algorithme génère des courbes en prenant le contour des obstacles grossis. Ensuite, il trace les droites de visibilité et ajoute ce que l'auteur appelle des *Voronoi chain points*. Ces derniers permettent de tracer des droites de visibilité supplémentaires. En sortie, le chemin obtenu est plus court que dans le cas de la méthode du diagramme de Voronoi et plus sécuritaire que pour l'approche par graphique de visibilité.

2.2.3 Approche par décomposition cellulaire

L'approche par décomposition cellulaire est probablement la méthode de planification de chemin la plus étudiée [Latombe, 1991]. Elle consiste en la décomposition de l'espace libre du robot en régions simples appelées cellules et généralement en une recherche dans le graphe reliant chaque cellule entres elles. Dans ce graphe, il est possible d'associer un coût pour passer d'une cellule à l'autre. Ce coût peut être basé sur plusieurs critères reliés à la topologie du terrain, aux capacités du robot, à la distance de la destination, etc. Typiquement, il existe deux méthodes pour bâtir une décomposition de l'environnement du robot :

- La décomposition cellulaire *exacte* décompose l'espace libre en cellules qui une fois unies forment exactement l'espace libre du robot ;
- La décomposition cellulaire *approximative* utilise des cellules de forme prédéfinie (p. ex., triangles ou rectangles) qui une fois unies forment un sous-ensemble de l'espace libre.

La figure 2.5 montre un exemple de chemin obtenu par la méthode de décomposition cellulaire exacte.

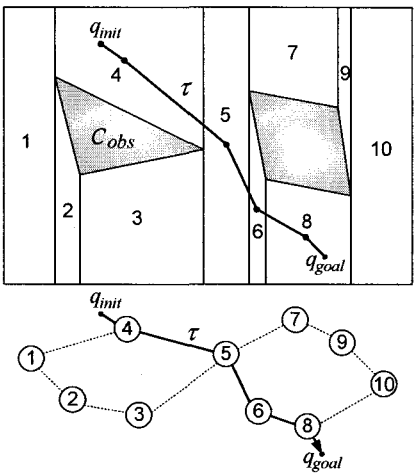


Figure 2.5 Exemple d'un chemin généré au moyen de la méthode de décomposition cellulaire exacte et graphe de connectivité entre les cellules

Dans un premier temps, l'espace libre est décomposé en un assemblage de cellules (numérotés de 1 à 10 à la figure 2.5). Ensuite, le graphe de connectivité est généré et enfin, le chemin τ est obtenu au moyen d'une recherche de graphe. La méthode approximative, quant à elle, est une des techniques de planification de chemin les plus utilisées vue la popularité d'une représentation approximative en «*grille*» de l'espace des configurations des robots [Siegwart et Nourbakhsh, 2004]. La figure 2.6 montre un exemple d'une décomposition en grille et d'un chemin planifié.

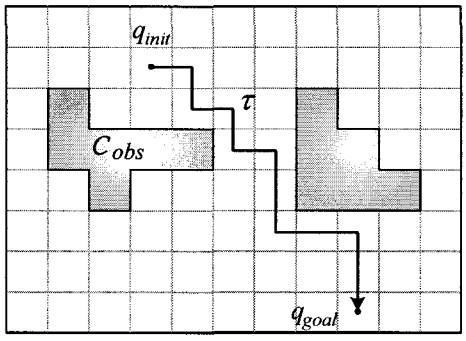


Figure 2.6 Exemple d'un chemin généré au moyen de la méthode de décomposition cellulaire approximative

Recherche de graphe Tel qu'expliqué précédemment, dans une représentation cellulaire de l'espace, il est possible de relier les cellules entre elles par des liens pondérés, c'est-à-dire qu'il est possible d'associer un coût pour se déplacer d'une cellule à l'autre.

S'il y existe une cellule jugée dangereuse, un coût infini peut lui être assigné afin de s'assurer que le robot ne visite pas cette cellule. Un chemin peut être obtenu au moyen d'une recherche dans le graphe de connectivité. Les mathématiques combinatoires proposent quelques algorithmes de recherche dans des graphes. L'algorithme de Dijkstra [Dijkstra, 1959] a jeté les bases de la recherche optimale dans des graphes. Cette méthode permet d'obtenir le chemin le moins «*coûteux*» dans des graphes à pondération positive (dont les poids sont positifs). Cette optimalité a un prix, selon le livre *Algorithmes de graphes* [Lacomme *et al.*, 2003], la complexité de calcul de la recherche peut atteindre $O(n^2)$.

L'algorithme de recherche $A^\star$ est un cas particulier de l'algorithme de Dijkstra qui cherche à réduire l'ampleur de la recherche en incluant une heuristique à la fonction coût [LaValle, 2006]. Par exemple, il est possible d'ajouter à chaque cellule un coût qui est fonction de la distance par rapport à la destination. Cela a pour effet de pénaliser les cellules qui s'éloignent de la position de la cible. Il existe aussi d'autres variantes de la méthode de Dijkstra telle que l'algorithme $B^\star$ qui guide lui aussi la recherche au moyen d'une heuristique [Berliner, 1979].

Méthode *grassfire* La méthode *grassfire* est une méthode de planification de chemin par décomposition cellulaire efficace et simple à mettre en oeuvre [Siegwart et Nourbakhsh, 2004]. L'algorithme envoie une «*onde de feu*» qui démarre de la position de la cible et qui se propage vers l'ensemble des cellules. Au passage de l'onde, chaque cellule est marquée de la distance entre elle et la cible. L'expansion de l'onde cesse lorsqu'elle atteint la position initiale du robot. Ensuite, le planificateur de chemin trace un chemin entre la position initiale et la cible en passant par les cases décroissant le plus rapidement. Comme chacune des cellules n'est visitée qu'une seule fois, la complexité de l'algorithme est de $O(n)$. La figure 2.7 montre un exemple de la méthode *grassfire*.

Cette dernière méthode permet d'obtenir un chemin dont l'optimalité n'est basée que sur la distance. En aucun cas elle ne permet de prendre en compte les contraintes physiques du robot et ne permet pas non plus de minimiser l'énergie consommée, contrairement aux approches par recherche de graphe qui utilisent une fonction coût qui peut inclure des considérations autres qu'uniquement la distance.

En somme, les méthodes de décomposition cellulaire permettent de calculer un chemin pour un robot au moyen d'algorithmes de complexité relativement faible. Néanmoins, dans bien des cas, ce type d'approche induit des contraintes au problème qui dégrade

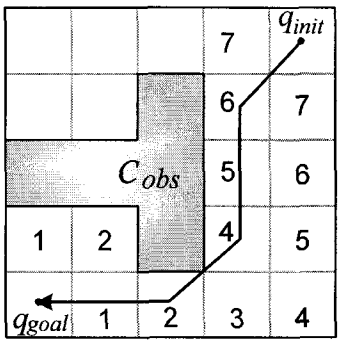


Figure 2.7 Exemple d'un chemin généré au moyen de la méthode *grassfire*

la qualité des chemins. Par exemple, il est coutume de faire passer le robot par le centre des cellules (voir figure 2.5, 2.6 et 2.7) ce qui, dans bien des situations, génère des chemins rugueux (en «*dent de scie*») et peu intuitifs. Dans le cas d'une décomposition irrégulière de l'environnement, il peut arriver que pour atteindre sa destination, le robot doive zigzaguer, car il est forcé à passer par le centre des cellules alors que la ligne droite est peut-être le véritable chemin optimal. L'article [Rekleitis *et al.*, 2009] discute de cette problématique et suggère l'utilisation d'un *corridor de sécurité* dans lequel le chemin rugueux peut être filtré afin d'obtenir un chemin plus lisse.

2.2.4 Méthode du champ de potentiel artificiel

Analogie des charges électriques L'idée de guider un robot au moyen de forces virtuelles a été introduite dans l'article *Real-time obstacle avoidance for manipulators and mobile robots* [Khatib, 1985]. L'auteur suggère d'associer le robot à une charge électrique qui est à la fois attirée par la cible et repoussée par les obstacles. En tout point de l'espace libre, il est possible d'associer un potentiel électrique scalaire proportionnel à la distance au carré de la cible et des obstacles (ou selon d'autres lois). Ce potentiel peut s'écrire ainsi en reprenant la notation de Khatib :

$$U_{art}(x) = U_{x_d}(x) + U_O(x),$$

où $U_{x_d}(x)$ est le potentiel attractif de la cible et où $U_O(x)$ est le potentiel répulsif des obstacles tous deux fonctions de la position. Le potentiel de la cible s'écrit ainsi :

$$U_{x_d}(x) = \tfrac{1}{2}k(x - x_d)^2,$$

où x est la position actuelle du robot sur l'espace libre et x_d est la position de la cible. La constante k est un gain qui peut être ajusté. En tout temps la direction à prendre est donnée par la résultante des forces électrostatiques :

$$\vec{F}_{res} = -\nabla U_{art}(x).$$

L'article *Potential field methods and their inherent limitations for mobile robot navigation* [Koren et Borenstein, 1991] a mis en lumière les limitations de la méthode du potentiel présentée par Khatib. Les principaux problèmes sont les suivants :

- présence de minimums locaux qui coincent le robot ;
- comportement oscillatoire à proximité des obstacles et dans les corridors étroits.

Ces mêmes auteurs proposent une méthode de potentiel couplée à une méthode de décomposition cellulaire qu'ils appellent *Virtual force field*. Cette approche développée pour les robots rapides décompose l'espace en grille cartésienne dans laquelle est associée une valeur de certitude de présence d'un obstacle dans la case. Ces valeurs peuvent être mises à jour à mesure que le robot découvre son environnement. Les forces virtuelles de répulsion deviennent proportionnelles aux valeurs contenues dans la grille.

Certains auteurs dont Volpe et Khosla [Volpe et Khosla, 1990] suggèrent l'utilisation de formes géométriques continues standards (rectangle, triangle, courbe gaussienne, etc.) afin d'encercler un obstacle. L'avantage d'une telle technique réside dans la simplicité du calcul du gradient qui s'obtient analytiquement.

Analogie de la mécanique des fluides L'article *Path planning using Laplace's equation* [Connolly *et al.*, 1990] propose d'utiliser une solution à l'équation de Laplace comme fonction de potentiel. Les auteurs rappellent l'absence de minimum local d'une telle fonction harmonique. Cette propriété vient améliorer les fonctions de potentiel classiques qui présentent des minimums locaux.

L'auteur de *Robot path planning using fluid model* [Li et Bui, 1998] reprend l'idée d'utiliser une solution de l'équation de Laplace comme générateur de potentiel et il y voit une analogie avec les écoulements de fluide. Il place à la position initiale du robot une source de fluide (débit positif) et à la cible, un puits (débit négatif). Les frontières du domaine sont étanches et donc aucun fluide ne peut les traverser. Mathématiquement, cela revient à imposer des conditions de Neumann (dérivées normales) nulles. L'évitement d'obstacle est donc implémenté au moyen des conditions de frontière. Les auteurs montrent que la solution de cette équation de Poisson ne peut admettre de minimum

local qu'à la position du puits. Par la suite, l'équation du milieu continu est résolue numériquement et l'on obtient l'écoulement permanent sous forme d'un potentiel vitesse. Finalement, le chemin est obtenu de la même façon que pour la méthode du potentiel classique, c'est-à-dire en suivant la direction du gradient négatif du potentiel entre la position initiale et la destination. L'approche se résume en trois étapes :

1. Discrétisation de l'espace libre en cellules (maillage triangulaire ou autre).
2. Résolution de l'équation de Laplace sur tout le domaine : $-\nabla^2\phi(x,y) = q$, avec $q = \delta$ (distribution Dirac) à la position initiale du robot, $q = -\delta$ à la position cible et $q = 0$ ailleurs. Aux frontières, les conditions de Neumann $\frac{\partial}{\partial n}\phi(x,y) = 0$ sont imposées afin de garantir l'étanchéité des parois.
3. Le chemin du robot s'obtient en suivant une courbe toujours tangente à la direction du gradient $-\nabla\phi(x,y)$ entre la source et le puits.

Certaines méthodes dérivées ont vu le jour. Par exemple, les auteurs de l'article *The fluid dynamics applied to mobile robot motion : the stream field method* [Keymeulen et Decuyper, 1994] utilisent aussi une approche inspirée des écoulements incompressibles de fluide afin de trouver des chemins lisses par la résolution de l'équation de Laplace. Cependant, dans leur cas, le chemin n'est pas obtenu forcément du gradient de la fonction de potentiel. Aussi, l'article *Vehicle motion planning using stream functions* [Waydo et Murray, 2003] propose une fonction complexe de la forme $\omega = \phi + \psi i$ pour résoudre l'équation de Laplace. Selon les auteurs, cette dernière méthode est mieux adaptée aux environnements dynamiques.

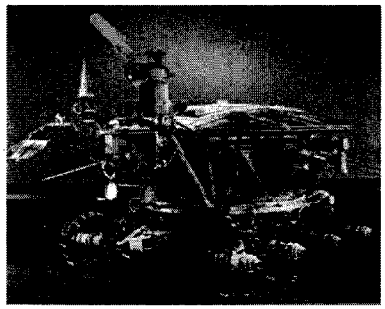
Conclusion sur la méthode du potentiel L'inconvénient le plus fréquemment cité venant d'une méthode de potentiel est la présence possible de minimums locaux. Il existe des méthodes réactives pour tenter de s'échapper de ces zones. Le livre *Computational Principles of Mobile Robotics* [Dudek et Jenkin, 2000] présente toute une série de solutions pour sortir de ces «*trous*»de potentiel. D'autres articles tel que *Real-time obstacle avoidance using harmonic potential functions* [Kim et Khosla, 1992] voit l'usage d'une fonction de potentiel harmonique comme une méthode préventive d'échappement des minimums locaux. C'est donc dire qu'à l'heure actuelle, le problème des minimums locaux est résolu. Enfin, la génération de chemin par la méthode du potentiel artificiel offre l'avantage de s'appuyer sur des phénomènes physiques. Ce constat rend intuitifs les chemins obtenus, car dans la nature les corps en mouvement sont guidés par des lois similaires. Toutes ces lois physiques qui gouvernent les mouvements minimisent

l'énergie impliquée. Par conséquent, un robot utilisant la méthode du potentiel minimise lui aussi, d'une certaine façon, l'énergie qu'il consomme.

2.3 Exemples concrets de missions dans l'espace

Cette section présente un survol des méthodes de planification de chemin qui ont été implantées avec succès sur les *rovers* ayant exploré Mars et la Lune. La présentation des méthodes s'effectue aux moyens de tableaux qui résument pour chacune des sondes quelques détails de la mission, le système de vision dédié à la navigation ainsi que la stratégie de navigation et d'évitement d'obstacle. Le tableau 2.1 présente les *rovers Lunokhod 1* et *2*.

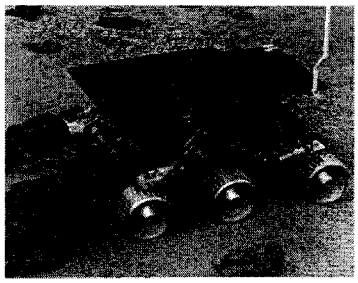
Tableau 2.1 Sommaire de la méthode de navigation de *Lunokhod 1* et *2*

Lunokhod 1* et *2 (Union soviétique)	
Année de lancement	1970 et 1973
Durée d'opération (mois)	14 et 3
Distance parcourue (m)	47000 et 36100
Vitesse moyenne (cm/s)	27 ou 54
Capacité de calcul (MIPS)	–
Système de vision pour navigation	1 caméra télévision
Stratégie de navigation et d'évitement d'obstacle	Les robots sont téléopérés à partir d'une base terrestre par 5 opérateurs [Kring, 2006]. C'est la proximité de la Lune par rapport à la Terre qui permet l'opération en quasi-temps réel (environ 2 s de décalage). Les distances et durées d'opération sont tirées de l'article *Soviet rover systems* [Carrier, 1992].
 Image tirée de [NASA, 2008a]	

Les *rovers Lunokhods* ne sont pas d'un grand intérêt dans l'analyse de l'état de l'art des algorithmes de génération de chemin, car ces robots ne possèdent aucune autonomie.

Ils sont présentés ici, car historiquement ce projet est le premier à avoir utilisé des robots en sol extraterrestre. Le prochain robot présenté est le premier à avoir réussi à explorer le terrain martien. Il s'agit de *Sojourner, rover* du programme *Pathfinder.*

Tableau 2.2 Sommaire de la méthode de navigation de *Sojourner, rover* du programme *Pathfinder*

Sojourner (NASA)	
Année de lancement	1996
Durée d'opération (sol)	83
Distance parcourue (m)	100
Vitesse moyenne (cm/s)	0.3
Capacité de calcul (MIPS)	0.25 [Muirhead, 2004]
Système de vision pour navigation	Passif. Caméra stéréo monochrome
Stratégie de navigation et évitement d'obstacle	Une fois par jour la base terrestre envoie une série d'emplacements à atteindre ainsi qu'un temps maximal de calcul permis. Ensuite, le robot observe son environnement par son système *Imager for pathfinder* (IMP) [Matijevic et Shirley, 1997] et recherche les obstacles devant lui au moyen d'algorithmes de traitement d'image. Pour aider cette recherche, il marque le sol de lignes horizontales à l'aide d'un projecteur laser. C'est l'analyse des photos prises avec et sans les lasers qui permet de déceler les obstacles (roche, trou, etc.). Si la voie est libre, le rover avance en ligne droite sinon, il tourne pour éviter l'obstacle. L'analyse visuelle du terrain est refaite environ à tous les 7 cm de déplacement. C'est ainsi qu'il converge vers ses coordonnées cibles [Mishkin *et al.*, 1998].
 Image tirée de [NASA, 2008b]	

De par sa détection et son évitement des obstacles, ce rover a fait preuve d'autonomie. Une autonomie qui est toutefois grandement limitée par l'intervention soutenue de l'humain dans le processus de navigation et par les limites de son système de vision qui ne permettait pas au robot de comprendre son environnement sur de grandes distances. C'est pourquoi le *rover* n'aura parcouru que 100 mètres en 83 journées martiennes.

Les prochains *rovers* présentés sont un exemple de réussite. En effet, les robots du programme *Mars exploration rovers* (MER) sont depuis 2003, toujours en opération sur la planète Mars.

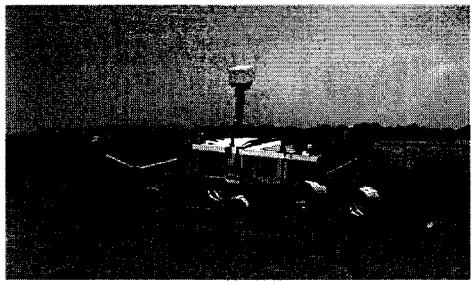
L'algorithme GESTALT est une approche de type *roadmap* de planification de chemin, alors que *Field D** est plutôt une méthode de décomposition cellulaire basée sur une recherche de graphe. Le couplage de deux méthodes de génération de chemin différentes est utile dans le cas de l'échec d'une des méthodes. Cela s'est d'ailleurs produit lors du sol 108, GESTALT a été incapable de trouver un chemin sécuritaire à la suite des 105 minutes de calcul allouées. C'est la méthode globale *Field D** qui a permis de débloquer le robot [Carsten *et al.*, 2007]. L'année 2012 sera l'hôtesse d'un nouveau *rover* américain déployé à la surface de Mars. Il s'agit du *Mars Science Laboratory* (MSL).

Bien que physiquement beaucoup plus imposant que ses prédécesseurs, MSL ne semble pas promettre de grandes innovations en matière de planification de chemin. Essentiellement, ce robot reprend les grandes lignes des algorithmes implantés sur les *rovers* du projet MER.

Tableau 2.3 Sommaire de la méthode de navigation du programme MER

Spirit/Opportunity (NASA)	
Année de lancement	2003
Durée d'opération (sol)	Plus de 2000 (Opportunity est toujours en activité)
Distance parcourue (m)	7730 (Spirit), 21860 (Opportunity)
Vitesse moyenne (cm/s)	5
Capacité de calcul (MIPS)	20
Système de vision pour navigation	*NavCam* : Passif. Caméra stéréo monochrome
Stratégie de navigation et d'évitement d'obstacle	Chaque jour, une série de coordonnées à atteindre sont téléchargées dans le *rover* à partir de la Terre. Lorsque le robot est en mode *Autonomous navigation with hazard avoidance* (AutoNav) [Maimone, 2007], il progresse lentement avec précaution afin d'atteindre ses cibles par un chemin sécuritaire. Avant d'amorcer tout mouvement, il prend des images de son environnement immédiat. L'analyse numérique des images permet de bâtir une carte de traversabilité [Howard et Tunstel, 2006]. Celle-ci est une vue de haut du terrain quadrillé où chaque cellule est qualifiée en terme de facilité à la traverser. Ensuite, un générateur de chemin appelé *Grid-based Estimation of Surface Traversability Applied to Local Terrain* (GESTALT) [Goldberg *et al.*, 2002] génère sur la carte plusieurs chemins potentiels en arc de cercle. L'algorithme évalue chacun des chemins selon les critères suivants : – évitement des obstacles ; – minimisation du temps de rotation des roues ; – longueur du chemin ; – atteinte de l'objectif. Dans le cas où GESTALT ne parvient pas à trouver une solution, c'est le planificateur de trajet global qui prend le relais. Ce dernier utilise l'algorithme *Field D** sur la carte globale (assemblage filtré de carte de traversabilité locale). Contrairement aux algorithmes classiques, *Field D** permet de générer un chemin qui ne passe pas forcément par le centre ou les sommets des cellules [Carsten *et al.*, 2007].
 Image tirée de [NASA, 2008b]	

Tableau 2.4 Sommaire de la méthode de navigation de MSL

Mars Science Laboratory (NASA)	
Année de lancement	2011
Durée d'opération prévue (sol)	670
Distance à parcourir (m)	20000
Vitesse moyenne (cm/s)	5-10
Capacité de calcul (MIPS)	Plus de 200
Système de vision pour navigation	*NavCam* (même que MER) [Forgave *et al.*, 2006]
Stratégie de navigation et d'évitement d'obstacle	En date de rédaction de ce mémoire, JPL prévoit mettre en oeuvre une stratégie de navigation et d'évitement d'obstacle similaire à *Spirit/Opportunity* tel que l'algorithme GESTALT sur lequel on aurait amélioré les capacités en terrain rugueux [Goldberg *et al.*, 2002]. La corrélation d'image stéréo pour générer la carte de traversabilité serait à nouveau implantée dans le système d'autonomie. [Krasner, 2002].
 Image tirée de [NASA, 2008b]	

2.4 Sommaire de l'état de l'art actuel

Depuis les quatre dernières décennies, l'homme a imaginé toutes sortes de méthodes ayant pour but de tracer les chemins des robots mobiles. Une des plus anciennes est la méthode *roadmap* qui permet, au moyen d'un réseau de courbes, de réduire l'ampleur de la recherche de chemin. C'est cette même méthode qui a été implantée avec succès sur les *rovers* du programme *Mars Exploration Rovers*. La difficulté de cette méthode réside en la construction d'un réseau de courbes qui permet au robot d'accéder à l'ensemble de l'espace libre. Il existe beaucoup d'approches, certaines déterministes et d'autres probabilistes, qui permettent la construction du *roadmap*. Une fois le *roadmap* obtenu, l'algorithme sélectionne un chemin considéré acceptable parmi l'ensemble du réseau de courbes.

La méthode de décomposition cellulaire a été, elle aussi, étudiée de façon intensive. Celle-ci décompose l'espace libre du robot en un nombre fini de cellules reliées par un lien de connectivité auquel il est possible d'associer un coût. Chaque cellule peut aussi être marquée d'une valeur indiquant la certitude d'y trouver un obstacle ou bien l'altitude moyenne de la cellule. Puis, un algorithme recherche dans l'assemblage des cellules, le maillage, un chemin qui rencontre certains critères d'optimalité. La faiblesse de cette méthode vient des contraintes qu'elle impose à la solution. En effet, généralement ce type de méthode force le chemin à passer par le centre ou bien les sommets des cellules. Ces contraintes limitent la quantité de solutions envisageables. Il peut arriver qu'un robot, «*cloué*» à son maillage, doive effectuer de nombreux changements de direction pour atteindre sa destination alors que la ligne droite est libre d'obstacle. Aussi, cette surcontrainte rend la solution sensible aux maillages placés en entrée. Deux maillages presque identiques peuvent engendrer deux chemins complètement différents. Un avantage de l'approche réside dans sa faible complexité qui, sous certaines conditions, peut atteindre $O(n \log n)$ ou même $O(n)$.

D'autres chercheurs ont vu dans la physique gouvernant le mouvement des corps, une façon à la fois simple et élégante de guider un robot vers sa cible. Pour ce faire, il est possible d'élever en potentiel la position initiale du robot et d'associer un potentiel minimal à la destination à atteindre. Les obstacles peuvent être potentiellement élevés ou retirés du domaine. Le chemin s'obtient par la recherche d'une courbe qui décroît en potentiel et qui atteint le minimum global, la cible. Cette situation est homologue à une charge électrique (le robot) qui se laisse guider jusqu'à une position d'équilibre (la cible) par l'ensemble des forces électriques qui agissent dessus. Cette façon d'aborder

le problème apporte de nombreux avantages. D'abord, la solution est généralement indépendante de la qualité et de la régularité du maillage. Aussi, les chemins obtenus sont souvent lisses, continus et intuitifs. Cependant, l'inconvénient le plus souvent cité est la présence de minimums locaux dans la fonction potentiel ce qui peut amener le robot à une solution qui n'atteint pas la cible. Il existe des méthodes *réactives* qui, une fois le robot coincé, permettent de sortir des minimums locaux. Une autre solution au problème est qualifiée de *préventive*, car c'est dans la construction de la fonction potentiel que les minimums locaux sont évités. Pour ce faire, il suffit d'utiliser une fonction harmonique et on obtient la certitude mathématique qu'une telle fonction ne présente pas de minimum local. Néanmoins, cette prévention amène une complexité de calcul élevée.

Il existe aussi une catégorie de méthodes dite *intelligentes* qui n'a pas été présentée dans cette revue de littérature. Les méthodes *intelligentes* telles que les algorithmes neuronaux et la logique floue ont été exclues dès le début de la définition du projet. Le caractère imprévisible de ces méthodes les rendent peu utiles à une application spatiale où la stabilité est un requis de la plus haute importance.

CHAPITRE 3

DÉFINITION ET OBJECTIFS DU PROJET

L'analyse de l'état de l'art a montré qu'actuellement les *rovers* explorateurs de planète utilisent généralement une structure de données régulière pour modéliser le terrain et pour planifier des chemins. Il est probable que les prochaines générations de système de vision embarqué (*LIDAR*) retourneront un nuage de points 3D qui, une fois traité, formera une représentation irrégulière de l'environnement des robots. Tel qu'expliqué à la section 2.4, les méthodes typiques de planification de chemins basées sur la décomposition cellulaire sont fort sensibles au conditionnement des maillages rendant ainsi ces méthodes presque qu'inutilisables sur des modèles de terrain irréguliers. Quant aux approches de champ de potentiel, pour être dépourvue de minimums locaux, la fonction de potentiel doit être harmonique. Pour former celle-ci, le calcul est coûteux. De plus, la littérature ne semble pas proposer d'applications des fonctions harmoniques à la génération de chemins sur des maillages irréguliers. L'ensemble des applications relevées calcule la fonction de potentiel harmonique sur une grille cartésienne régulière au moyen de la méthode des différences finies, ce qui n'est pas applicable aux modèles de terrain irréguliers.

Les méthodes actuelles retournent des chemins formés par une liste de points 2D ou 3D formant une ligne. Toutefois, aucun robot n'est apte à suivre parfaitement une ligne vu la possibilité de glissement des roues. Il est donc utile de fournir en plus d'un chemin, un corridor de sécurité. Le corridor permet de quantifier dans quelle mesure le robot peut sortir de son chemin sans compromettre sa sécurité. Pour mieux comprendre le concept du corridor de sécurité, la figure 3.1 est présentée.

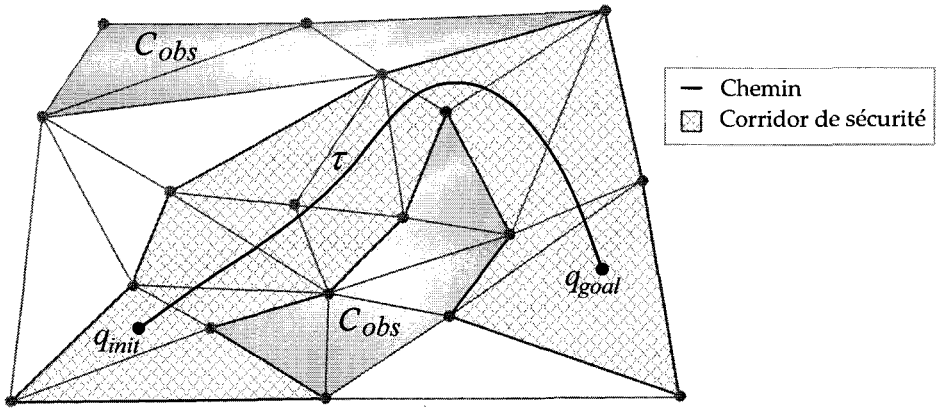


Figure 3.1 Schéma du maillage d'un terrain accompagné d'un chemin et d'un corridor de sécurité (les symboles utilisés reprennent les variables présentées au chapitre 2)

3.1 Objectif principal

L'objectif principal de recherche est maintenant présenté de façon formelle :

> *L'objectif de recherche du projet est de développer et de valider un «algorithme efficace» de génération de «chemins optimaux» entre une position initiale et une destination dans un maillage irrégulier.*

La section 3.2 présentera les critères de performance à atteindre qui permettent de définir les termes «*algorithme efficace*» et «*chemins optimaux*». La figure 3.2 indique les intrants et extrants de l'algorithme à développer.

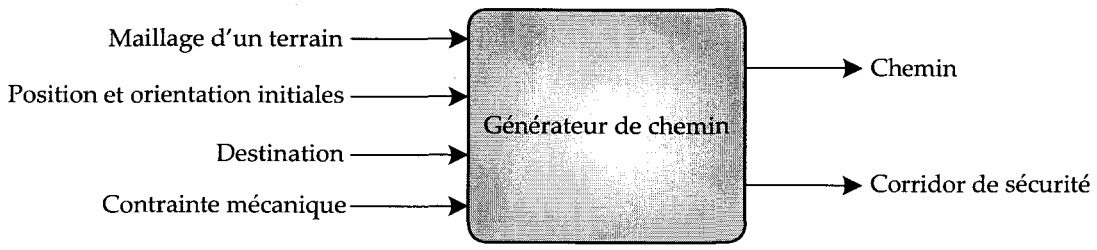


Figure 3.2 Schéma des entrées et sorties de l'algorithme à développer

L'algorithme reçoit en entrée un maillage du terrain, la position et l'orientation initiales du robot de même que la destination à atteindre. Il considère aussi une liste de para-

mètres qui permettent de cerner les contraintes mécaniques du *rover*. Enfin, il retourne le chemin ainsi qu'un corridor de sécurité.

3.2 Critères de performance à atteindre

L'étalement des critères de performance à atteindre permet de mieux comprendre ce que le candidat appelle «*algorithme efficace*» et «*chemin optimal*». Ces deux concepts sont au coeur de l'objectif du projet. Un «*chemin optimal*» doit rencontrer l'ensemble des critères présentés au tableau 3.1. Les critères sont accompagnés d'une pondération basée sur le jugement du candidat qui quantifie l'importance sur une échelle de 0 à 10 (10 étant un critère de haute importance).

Tableau 3.1 Critères de performance d'un «*chemin optimal*»

Critère	Description et explication	Pondération
Atteinte de la cible	Si la destination appartient à l'espace libre, le robot doit atteindre cette cible.	10
Longueur	Le chemin doit minimiser la distance à parcourir.	7
Énergie	Le chemin doit considérer la minimisation de l'énergie consommée par le robot lors du déplacement.	9
Sécurité	Le robot doit suivre un chemin dépourvu d'obstacle (roche, cratère, etc.).	10
Rugosité	Le chemin doit être lisse et dépourvu de changement vif de direction.	6
Contrainte holonome	Le chemin doit respecter les limites mécaniques du robot.	9

Les critères jugés de plus hautes importances pour un «*chemin optimal*» sont l'atteinte de la cible ainsi que la sécurité du robot. L'optimalité du chemin au sens mathématique vient surtout des critères de longueur de chemin et d'énergie consommée. Enfin, l'aspect rugosité permet de minimiser l'accumulation d'erreur odométrique, car chaque fois que le robot change brusquement d'orientation, il glisse et accumule des erreurs de positionnement. Le tableau suivant expose les critères qui permettent de considérer un algorithme comme «*efficace*».

Tableau 3.2 Critères de performance d'un «*algorithme efficace*»

Critère	Description et explication	Pondération
Complexité[1]	La complexité d'un algorithme influence directement le temps de calcul. Ce critère doit être minimisé. Il peut être estimé au moyen du temps de calcul.	7
Robustesse	Un algorithme doit être capable de gérer la presque totalité des situations sans retourner d'erreur à l'exécution.	10
Contrainte d'optimisation	Tout problème d'optimisation impose des contraintes qui limitent la dimension de la recherche de solution. L'algorithme doit imposer le minimum de contrainte dans sa recherche.	8
Complétude[1]	Un algorithme complet retourne une solution lorsqu'elle existe et retourne un message d'erreur dans le cas contraire.	9
Capable d'utiliser un maillage 3D	L'algorithme doit être en mesure d'utiliser un maillage 3D, autrement un traitement est requis pour abaisser la dimension à 2.5D[1].	7
Insensibilité au maillage	Le conditionnement et la qualité du maillage doivent influencer minimalement la solution.	8
Flexibilité de la solution	L'algorithme doit pouvoir adapter la solution au besoin de l'utilisateur (p. ex., favoriser un chemin sur un parcours ensoleillé).	7

[1]Les définitions sont présentées à la section 2.1

La robustesse et la complétude sont les critères les plus importants d'un algorithme. Ceci est d'autant plus vrai dans le contexte d'une application spatiale où un algorithme qui échoue constamment est proscrit. La complexité minimale est importante vu la faiblesse des capacités de calcul des processeurs spatiaux. Néanmoins, un robot immobile dans un environnement statique peut prendre son temps à planifier un «*chemin optimal*». Dans cette situation précise, la qualité du chemin est de plus haute importance que le temps de calcul. Mieux vaut que l'algorithme requiert quelques secondes de plus que d'aventurer le robot sur un chemin incertain ou trop énergivore. Aussi, il est préférable que l'algorithme n'impose pas de contrainte non essentielle à la solution telle que de forcer le chemin à passer par le centre des cellules d'un maillage.

3.3 Objectifs secondaires

Les objectifs secondaires sont les suivants :

- développer une méthode permettant d'obtenir un modèle de terrain à partir de données brutes *LIDAR* ;
- généraliser l'algorithme de planification de chemin afin de permettre son implantation sur n'importe quel robot mobile utilisant un modèle de terrain en maillage triangulaire ;
- déterminer quantitativement la performance de l'algorithme en regard des critères présentés aux tableaux 3.1 et 3.2 ;
- comparer les résultats de l'algorithme développé avec les résultats obtenus d'une méthode de décomposition cellulaire utilisé par l'ASC ;
- implanter l'algorithme sur un banc d'essai robotisé appartenant à l'ASC.

CHAPITRE 4

MODÉLISATION DU TERRAIN

Ce chapitre présente la stratégie retenue pour l'obtention d'un modèle de terrain à partir de données mesurées par un système de vision *LIDAR*. Ce modèle, qui est, à partir de maintenant, appelé *maillage*, doit être construit en tenant compte qu'il sert d'entrée à l'algorithme de génération de chemin. La revue de littérature a montré que suivant la taille du maillage (p. ex., le nombre de triangles) la durée d'un calcul de planification de chemin augmente. L'algorithme de modélisation de terrain doit donc considérer certaines contraintes en ressource de calcul. Ce dernier doit chercher un compromis entre la précision du modèle et la quantité d'information à conserver. D'autres aspects à considérer sont les capacités physiques du robot. Par exemple, dans le cas d'un robot capable de surmonter des roches de dimension importante, celui-ci peut accepter sans risque un maillage ayant perdu l'information des rugosités fines.

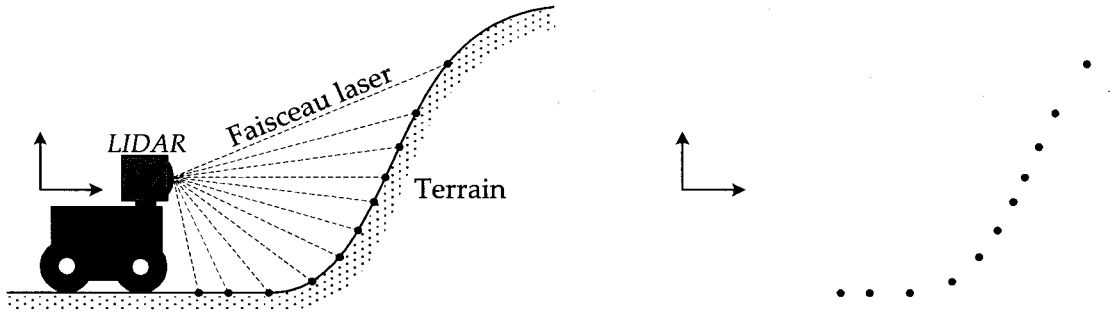
La section 3.2 a mis l'emphase sur la sécurité du robot dans les critères de recherche de chemin. Cette maximisation de la sécurité peut débuter au niveau de la modélisation du terrain. En rejetant les cellules jugées non sécuritaires ou impossibles à atteindre par le robot, ce dernier se tient ainsi à distance des obstacles. Il est aussi possible de grossir les obstacles par retrait de cellules afin que le robot s'éloigne encore davantage des dangers.

Les prochaines sections exposent en détail tout le processus de modélisation du terrain. Chaque étape est appuyée par un exemple concret provenant de données réelles obtenues du banc d'essai de l'ASC.

4.1 Mesure au moyen d'un capteur *LIDAR*

Tel qu'expliqué antérieurement, ce travail de recherche considère l'usage d'un capteur actif de type *LIDAR* pour l'observation du terrain. Il existe plusieurs technologies de *LIDAR* pouvant servir à la mesure de l'environnement d'un robot. La méthode la plus courante consiste à émettre un faisceau laser dans une direction connue et à capter le retour du laser. Le temps de vol indique la distance parcourue par le laser car sa vitesse est connue (vitesse de la lumière). En lançant une grande quantité de faisceaux dans toutes les directions, il devient possible d'obtenir un nuage de points représentant le

terrain. La figure 4.1 montre un exemple de contexte de mesure ainsi qu'un nuage de points résultant.



(a) Contexte de mesure (b) Nuage de points résultant

Figure 4.1 Mesure d'un terrain au moyen d'un *LIDAR*

Le schéma de la figure 4.1 illustre une situation 2D. Dans le contexte d'une exploration planétaire le terrain à mesurer est 3D. La figure 4.2 montre un exemple de données 3D réelles mesurées sur le banc d'essai de l'ASC.

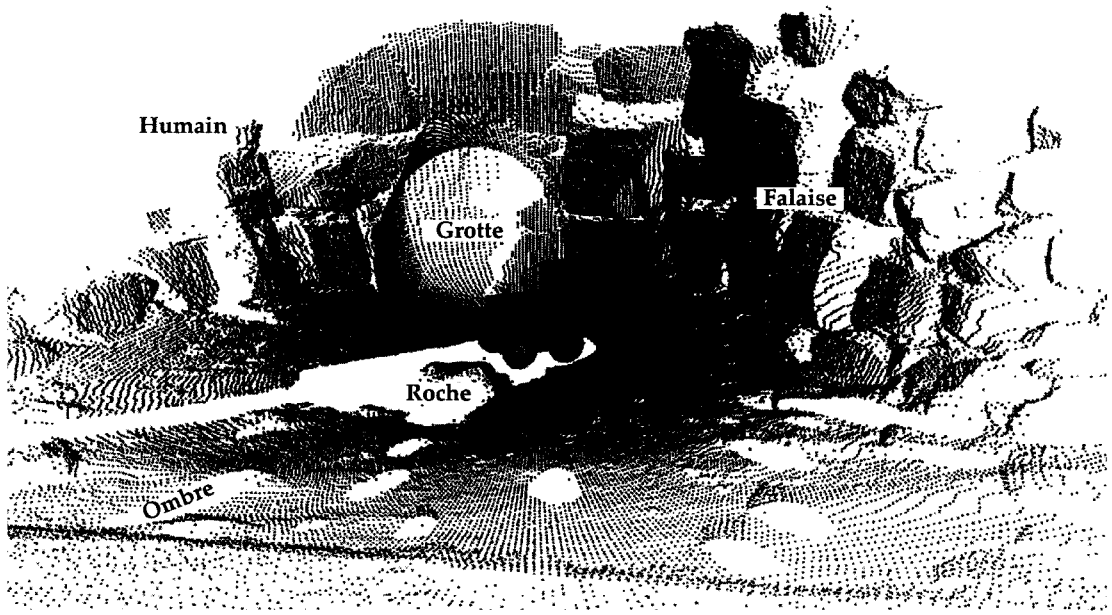


Figure 4.2 Exemple d'un nuage de points 3D obtenu sur le banc d'essai de l'ASC

La figure 4.2 présente un nuage contenant 150 000 points 3D. La scène contient une falaise, une grotte, des roches et un observateur humain. Le robot est présent sur la figure uniquement pour bien saisir le contexte de mesure. La scène contient aussi des

secteurs sans données appelés *ombres* ou zones d'occlusion. Ce sont les endroits où à partir du point du vue du capteur, aucun faisceau laser ne peut se rendre car un object solide (p. ex., une roche, le robot lui-même) bloque la ligne de vision. La figure 4.3 présente un exemple d'une situation de mesure menant à une discontinuité des données (ombre).

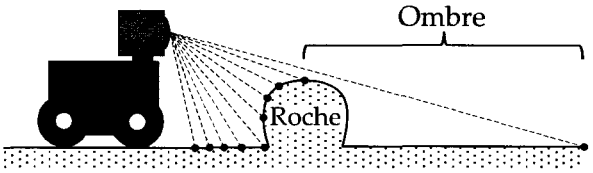


Figure 4.3 Exemple d'une situation de mesure menant à une ombre

Le *LIDAR* ne peut lancer qu'un faisceau laser à la fois dans une direction donnée. Afin de balayer tout l'environnement du robot, le capteur effectue un balayage de 360° dans le plan horizontal (angle d'azimut) et de 180° dans le plan vertical (angle d'élévation) avec un incrément angulaire constant. Chaque mesure de distance résulte en un point dans le nuage. La figure 4.4 illustre les angles balayés lors d'une mesure complète du terrain.

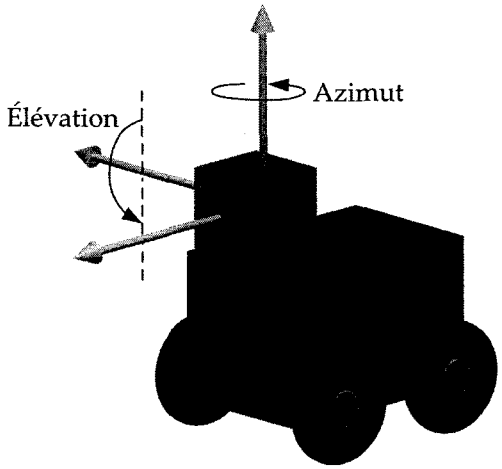


Figure 4.4 Repère du capteur et angles balayés lors d'une mesure complète

Les données brutes retournées par le capteur sont référencées dans le repère local du *LIDAR* en coordonnées sphériques.

4.2 Rejet des données aberrantes

Comme tout capteur opérant dans un environnement de test réel, le *LIDAR* retourne parfois des mesures bruitées. Il existe un bruit venant de l'incertitude de mesure liée à la précision du capteur, mais aussi une forme plus gênante de bruit qui positionne quelques points loin du reste du nuage. Ces points dispersés de la moyenne sont qualifiés d'aberrants. La figure 4.5 montre un exemple réel d'un nuage de points contenant ce type de donnée.

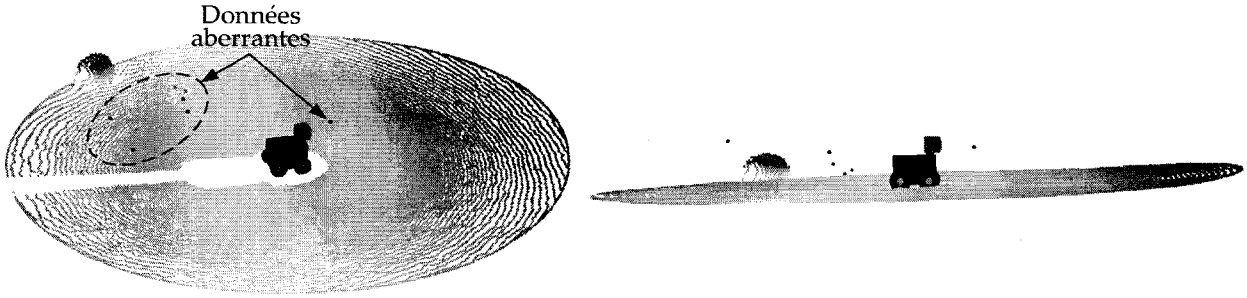


Figure 4.5 Exemple d'un nuage de points contenant des données aberrantes (la couleur est fonction de l'altitude des points)

4.2.1 Filtrage adaptatif médian

Dans le domaine du traitement d'image numérique, ce type de bruit *impulsif* est commun. Il s'agit d'un bruit de *sel et poivre*. Beaucoup de travaux ont été menés sur la reconstruction d'images corrompues par les bruits impulsifs. Le lecteur intéressé peut consulter l'ouvrage [Astola et Kuosmanen, 1997] qui explique les différentes méthodes de filtrage adaptées à ce problème. Le filtre médian est une approche largement utilisée pour corriger les pixels aberrants d'une image, car elle est efficace et requiert peu de ressource de calcul [Huang *et al.*, 1979]. Afin d'utiliser cette technique de filtrage sur les données brutes du *LIDAR*, il faut que ces données soient transformées en une forme d'image numérique.

Une image numérique en nuance de gris s'exprime mathématiquement comme une matrice 2D où chaque terme supporte le niveau de gris d'un pixel. La dimension de la matrice est liée à la résolution de l'image. Par convention, un pixel complètement noir possède une valeur de 0, un pixel blanc 1 et le gris est compris entre 0 et 1 exclusivement. À titre d'exemple, la figure 4.6 montre une image en nuance de gris s'accompagnant des valeurs numériques d'une portion de celle-ci.

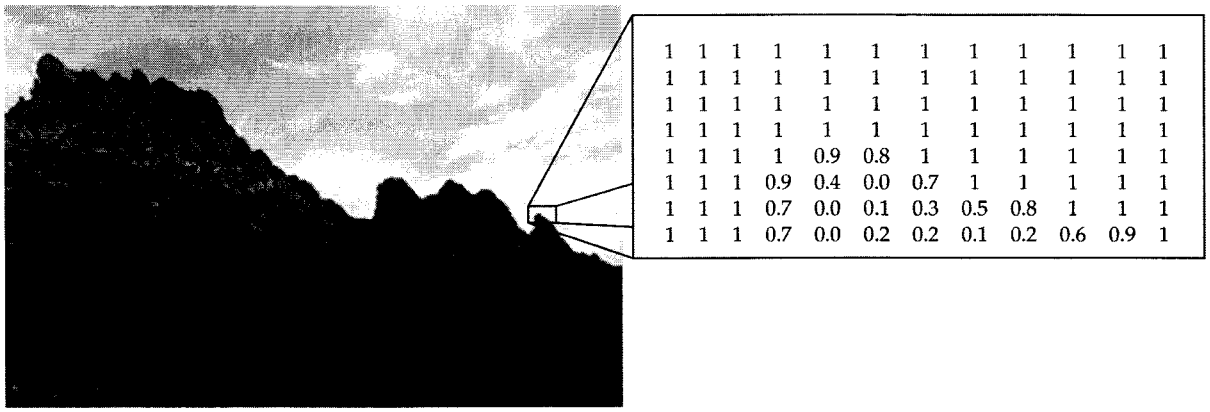


Figure 4.6 Exemple d'une image en nuance de gris s'accompagnant de valeurs numériques

Dans le cas du *LIDAR,* il est possible de placer les mesures de distance normalisées entre 0 et 1 dans une matrice. Dès lors, le nuage de points retourné par le capteur devient une image en nuance de gris et un filtre médian peut y être appliqué. L'équation 4.1 montre comment les données brutes sont positionnées dans une matrice de format $n \times m$:

$$M^*_{brute} = \begin{bmatrix} r^*(\theta_1, \phi_1) & r^*(\theta_2, \phi_1) & \dots & r^*(\theta_m, \phi_1) \\ r^*(\theta_1, \phi_2) & r^*(\theta_2, \phi_2) & & \\ \vdots & \vdots & \ddots & \\ r^*(\theta_1, \phi_n) & r^*(\theta_2, \phi_n) & \dots & r^*(\theta_m, \phi_n) \end{bmatrix}, \tag{4.1}$$

où θ est la coordonnée azimut, ϕ est la coordonnée élévation et r^* est la distance normalisée entre 0 et 1. La normalisation de la distance mesurée r_i se fait ainsi :

$$r^*_i = \frac{r_i}{\max(r)}, \tag{4.2}$$

avec $\max(r)$ la valeur maximale des distances mesurées. Une fois les données en place, celles-ci peuvent être filtrées. Pour bien comprendre la méthode de filtrage médian, un exemple impliquant des données 1D est présenté. Les données suivantes doivent être filtré :

$$X_{brute} = \begin{bmatrix} 1 & 70 & 5 & 4 \end{bmatrix}.$$

Les données brutes sont normalisées suivant l'équation 4.2 :

$$X^*_{brute} = \begin{bmatrix} 0.01 & 1.00 & 0.07 & 0.06 \end{bmatrix}.$$

Ensuite, le filtrage peut s'effectuer au moyen d'une fenêtre de dimension 1×3 qui balaye toute la matrice et qui remplace au passage le pixel du centre par la valeur médiane de la fenêtre. Pour calculer cette valeur médiane, les points contenus dans la fenêtre sont ordonnés en ordre croissant et la valeur centrale est conservée.

$$\begin{aligned} X^*_{filtre}(1) &= \text{mediane}(0.01, 0.01, 1.00) = 0.01 \\ X^*_{filtre}(2) &= \text{mediane}(0.01, 1.00, 0.07) = 0.07 \\ X^*_{filtre}(3) &= \text{mediane}(1.00, 0.07, 0.06) = 0.07 \\ X^*_{filtre}(4) &= \text{mediane}(0.07, 0.06, 0.06) = 0.06. \end{aligned}$$

Lorsque le centre de la fenêtre est au premier ou au dernier terme de la matrice, la valeur en bordure de la matrice est dupliquée tel que montré ci-haut. Le pixel bruité, dont la valeur normalisée est à 1.00, est bien rejeté par le filtre. Enfin, les données filtrées et normalisées sont transformées dans le domaine initial en multipliant par la valeur maximale (70) :

$$X_{filtre} = \begin{bmatrix} 1 & 5 & 5 & 4 \end{bmatrix}.$$

C'est cette méthode de filtrage qui est appliquée aux données brutes du *LIDAR*. Une fenêtre de dimension 4×1 donne de bons résultats comme en témoigne le nuage de points suivant de la figure 4.5 chez qui les données aberrantes ont été éliminées.

Figure 4.7 Exemple d'un nuage de points filtré

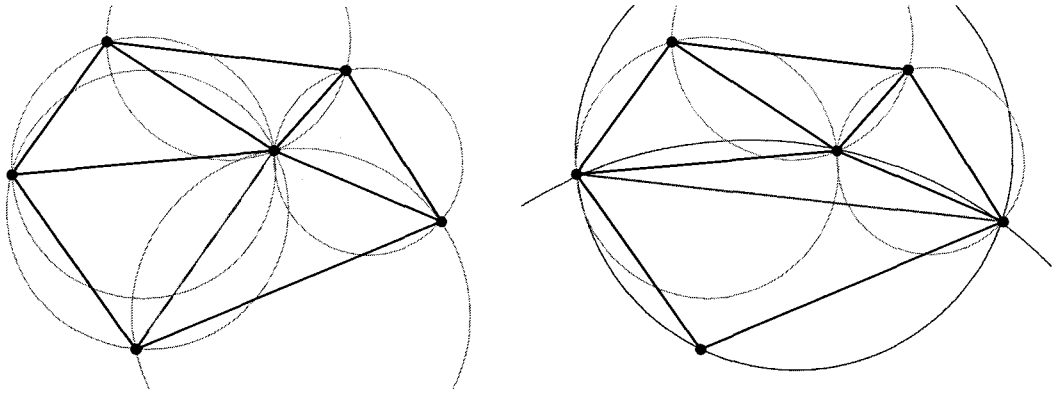
4.3 Génération du maillage

Cette section présente une méthode qui permet l'obtention d'un maillage à partir d'un nuage de points obtenu en coordonnées sphériques. Il existe plusieurs types de maillage pouvant soutenir un ensemble de points discrets. Par exemple, les maillages carrés ou triangulaires. Ce projet de recherche a opté pour un modèle triangulaire du terrain.

Ce choix s'explique par la capacité de ce type de maillage à bien épouser les formes complexes.

4.3.1 Triangulation

Il existe quelques algorithmes de triangulation comme la triangulation de Pitteway [Pitteway, 1973]. Celle-ci permet d'obtenir un maillage triangulaire, cependant elle ne garantit pas l'existence d'une triangulation pour tout ensemble de points [Gold, 1978]. La triangulation de Delaunay publiée initialement en 1934 [Delaunay, 1934], forme aujourd'hui la référence dans la construction d'un maillage triangulaire. Pour s'appeler triangulation de Delaunay, un maillage ne doit posséder aucun sommet de triangle à l'intérieur des cercles circonscrits passant par les sommets de chaque triangle. Cette condition porte le nom du *critère du cercle vide*. Pour comprendre ce critère, la figure 4.8 est présentée.



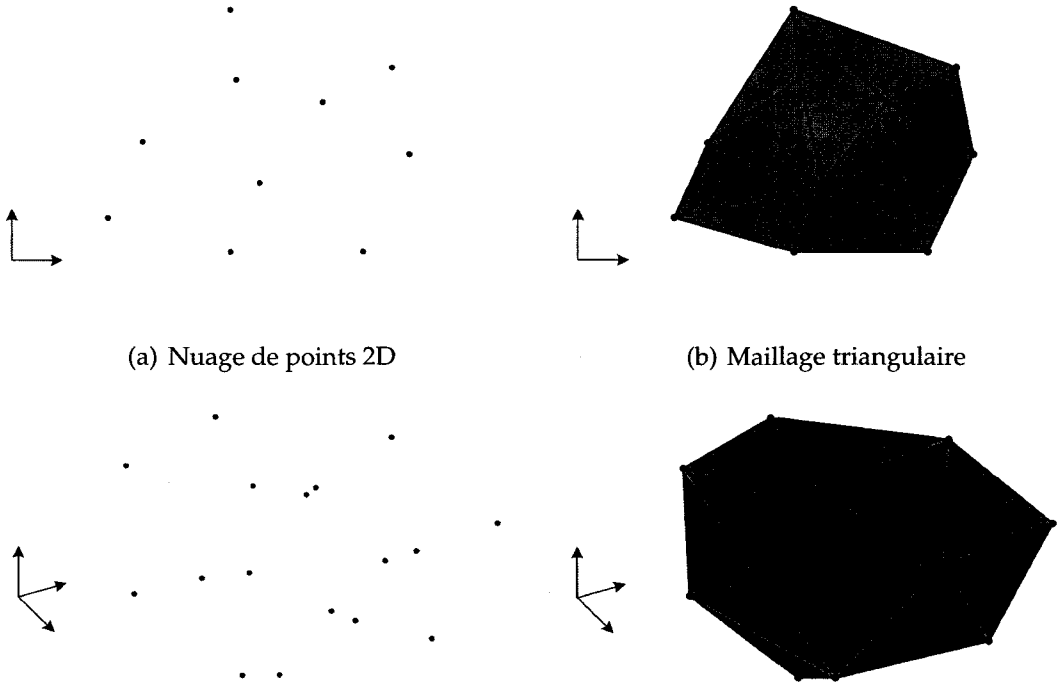
(a) Triangulation de Delaunay (b) Triangulation non Delaunay

Figure 4.8 Comparaison entre deux types de triangulation

La figure 4.8a montre une triangulation qui respecte le *critère du cercle vide*. En effet, les cercles en gris passant par les sommets d'un triangle n'englobent pas de sommets appartenant à d'autres triangles. Au contraire de la triangulation de la figure 4.8b chez qui deux cercles (en rouge) violent le critère. Ce dernier maillage a été formé à partir de la triangulation de Delaunay à laquelle une arête (en bleu) a été déplacée.

L'algorithme, dans sa forme générique, est capable de bâtir une triangulation d'un nuage de points exprimé en tout système de coordonnée de dimension n. En pratique, il est courant de trianguler un ensemble de points cartésiens de dimension 2 ou 3. Dans

le cas 2D, le maillage forme une surface alors que dans le cas 3D, il remplit un volume au moyen de tétraèdres. La figure 4.9 illustre cette distinction.



(a) Nuage de points 2D

(b) Maillage triangulaire

(c) Nuage de points 3D

(d) Maillage tétraèdrique

Figure 4.9 Comparaison entre une triangulation 2D et 3D

Le robot à l'étude navigue sur la surface d'un terrain. C'est donc l'algorithme de Delaunay 2D qui permet d'obtenir le maillage triangulaire du terrain navigable. Tel qu'expliqué à la section 4.1, le nuage de points retourné par le *LIDAR* est exprimé en coordonnées sphériques. C'est donc dans ce repère que le calcul de la triangulation s'effectue. Dans ce cas, l'algorithme de Delaunay 2D prend en entrée les coordonnées indépendantes azimut et élévation et fournit en sortie les triangles. La figure 4.10 montre le maillage 2D du nuage de points présenté à la figure 4.2 exprimé dans le repère sphérique.

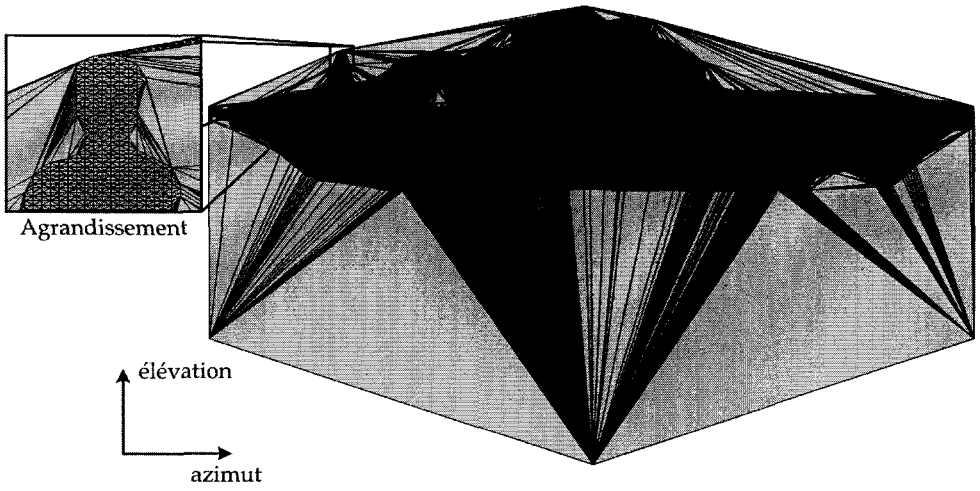


Figure 4.10 Exemple d'un maillage Delaunay 2D sphérique possédant 292 000 triangles ainsi qu'un agrandissement du haut du corps de l'observateur humain présent au moment de la prise de mesure

La compréhension d'un maillage dont les sommets sont exprimés dans un repère sphérique n'est pas triviale comme en témoigne la figure 4.10. Néanmoins, dans certain cas, il est possible de reconnaître certaines formes. Par exemple à la figure 4.10, il est possible de distinguer l'observateur humain qui était présent au moment de la mesure du terrain. À droite de celui-ci, l'entrée de la grotte est aussi décelable. C'est en transformant les sommets des triangles vers un repère cartésien que le maillage devient davantage compréhensible et utilisable. La figure 4.11 montre la triangulation exprimé dans un repère cartésien.

Ce dernier maillage représente bien le terrain réel exposé à la figure 4.2, cependant il possède certains triangles indésirables. Ceux-ci proviennent de l'algorithme de Delaunay qui retourne une surface continue même en l'absence de données en certains endroits (p. ex., ombre).

4.3.2 Rejet des cellules indésirables

Il existe deux catégories de triangles indésirables, les cellules ombrées et les cellules de frontière. La figure 4.12 montre un maillage possédant des cellules ombrées.

Les triangles ombrés ont rempli l'ombre à l'arrière de la roche. Dans le cas des cellules de frontière, ceux-ci proviennent d'un nuage de points concave dans le repère du maillage. Lorsque cette situation se produit, l'algorithme de Delaunay remplit la por-

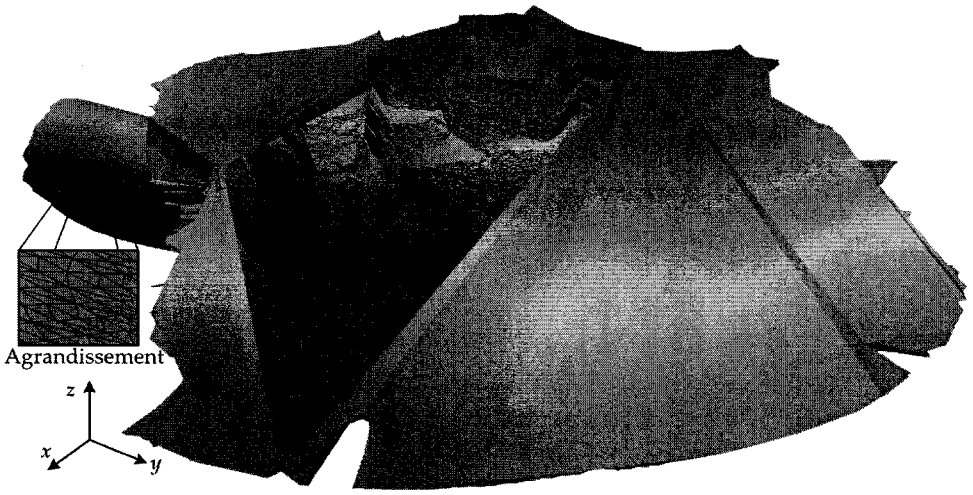
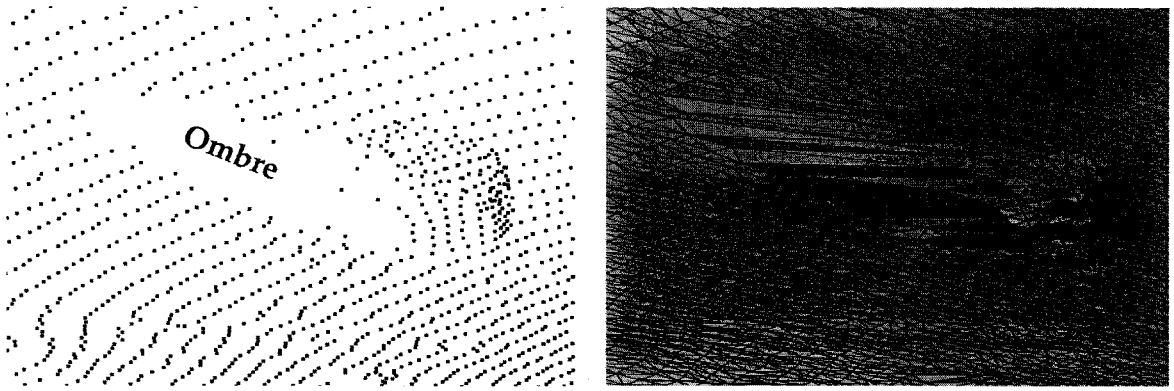


Figure 4.11 Exemple d'un maillage triangulé en coordonnée sphérique et dont les sommets ont été transformés ensuite vers un repère cartésien



(a) Nuage de points possédant une ombre (b) Maillage possédant des cellules ombrées

Figure 4.12 Exemple d'un nuage de points avec une discontinuité de donnée et le maillage résultant

tion concave afin de produire une enveloppe convexe de triangles. Les figures 4.12 et 4.13 illustrent cette situation.

Lorsque le maillage est transformé dans le repère cartésien, il arrive que les triangles de frontière (en blanc à la figure 4.13) deviennent de forte dimension. Les grandes cellules indésirables de la figure 4.11 sont des triangles de frontière. Les deux types de triangle indésirable doivent être retirés du maillage, car ils ne représentent pas le terrain réel ou bien le *LIDAR* n'a pas recueilli de mesure à ces endroits. Il est possible de détecter les triangles d'ombre et de frontière en observant le périmètre des cellules et la direction

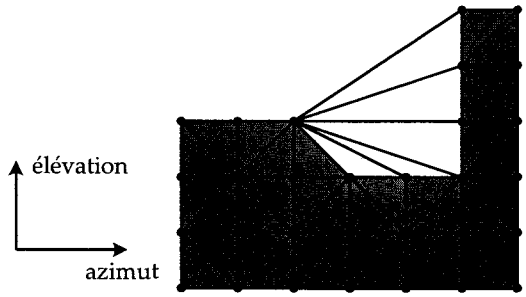


Figure 4.13 Exemple d'un maillage possédant des cellules de frontière (triangles blancs)

des vecteurs normaux aux cellules. Si la valeur du périmètre d'un triangle dépasse un niveau, il est probable que celui-ci soit fautif. De même si le vecteur normal à une cellule ne possède pas une composante pointant vers le capteur, il est possible que ce triangle soit ombré. En retirant ces triangles indésirables, le maillage semble plus fidèle au terrain tel qu'en témoigne la figure 4.14.

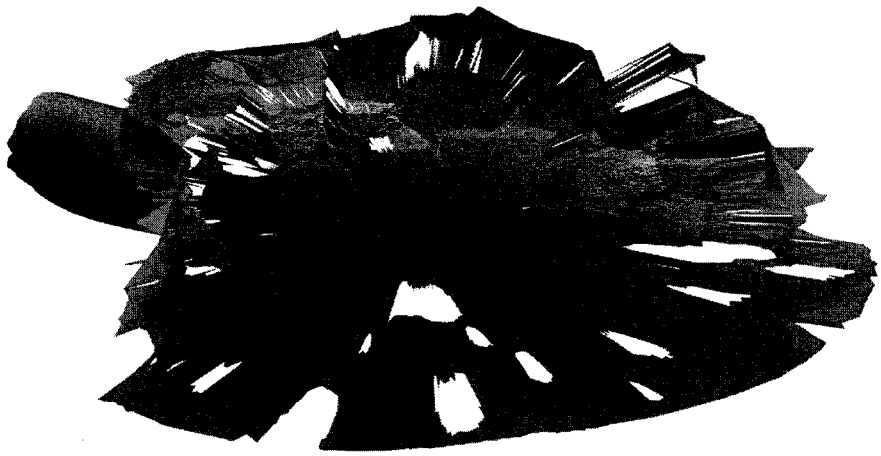


Figure 4.14 Exemple d'un maillage dont la plupart des cellules indésirables ont été retirées

4.4 Extraction du domaine navigable

Dépendemment des capacités mécaniques du robot mobile, il est possible que certaines cellules d'un maillage soient impossibles à atteindre voire dangeureuses pour le robot. En d'autres mots, *l'espace libre associé à un terrain et un robot n'englobe pas l'ensemble de l'espace des configurations*[1]. *L'espace des obstacles est non vide* et cette situation

[1]Voir la section 2.1 pour connaître les définitions des espaces.

arrive fréquemment. Dans le but de favoriser la sécurité du robot et de simplifier la planification de chemin, cette étape s'attarde à l'extraction du domaine navigable d'un maillage.

4.4.1 Rejet des murs et des plafonds

Dans le cas d'un robot n'étant que supporté (et non retenu) par la surface d'un terrain, les murs et les plafonds ne sont pas atteignables. Avant de rejeter les cellules formant les murs et plafonds, il faut d'abord les détecter. C'est l'analyse des vecteurs normaux aux surfaces qui renseigne sur l'orientation des différentes cellules du maillage. La figure 4.15 illustre des vecteurs normaux associés à un mur, un plafond et un plancher.

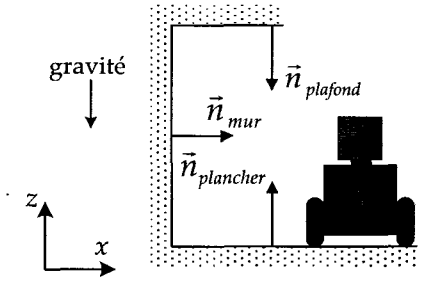


Figure 4.15 Exemple d'un plancher, d'un mur, d'un plafond et des vecteurs normaux associés aux surfaces

Dans le cas de la situation illustrée à la figure 4.15, la composante des vecteurs normaux alignée avec la gravité (z) permet de détecter les surfaces trop pentues. Effectivement, le plafond possède une composante z normalisée de -1, le plancher 1 et le mur possède une composante z nulle. Dans le cas d'un maillage triangulaire, le calcul des vecteurs normaux s'effectue au moyen du produit vectoriel illustré à la figure 4.16.

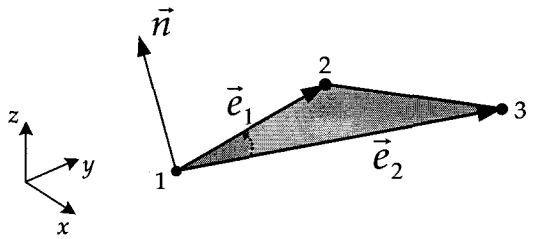


Figure 4.16 Calcul du vecteur normal à un triangle

Mathématiquement, le calcul du vecteur normal s'exprime ainsi :

$$\vec{n} = \frac{\vec{e}_2 \times \vec{e}_1}{\|\vec{e}_2 \times \vec{e}_1\|} \tag{4.3}$$

Afin que l'ensemble des vecteurs normaux soit consistant sur tout le terrain, la structure de données qui supporte les triangles doit numéroter les sommets des triangles dans le même ordre. Par exemple, les sommets du triangle de la figure 4.16 sont présentés en ordre croissant : 1, 2 ,3 dans la structure de données. La numérotation 2, 1, 3 n'est pas consistante, car les sommets ne sont pas présentés dans un ordre défini. Dans le cas d'un seul triangle cela a peu de conséquence. Toutefois, dans le cas d'un assemblage de triangles (maillage), il est nécessaire de respecter l'ordre de numérotation pour l'ensemble des cellules. Dans le cas contraire, il peut se produire la situation d'un triangle considéré comme un plancher qui est voisin immédiat d'un triangle plafond, ce qui n'est pas consistant. Habituellement, les algorithmes de Delaunay retournent une triangulation correctement numérotée.

La figure 4.17 montre un exemple d'un terrain modélisé dont la composante z des vecteurs normaux est illustrée au moyen de couleurs.

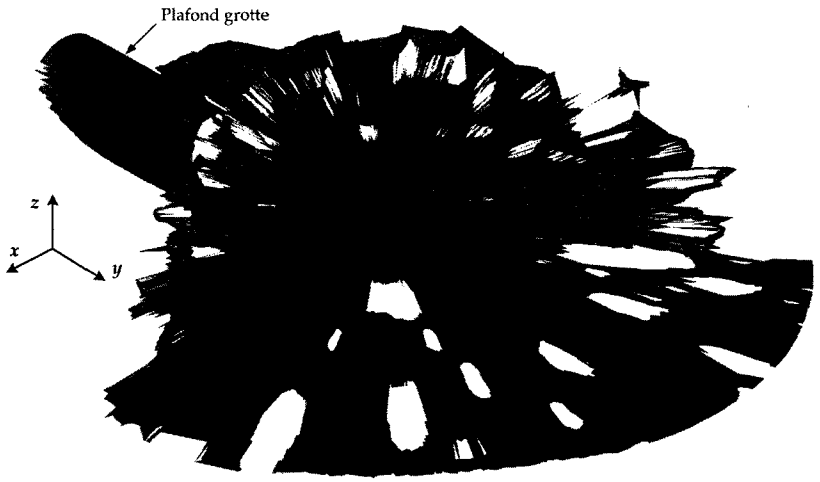


Figure 4.17 Visualisation de l'inclinaison des surfaces d'un terrain. La couleur est fonction de la composante z des vecteurs normaux aux cellules

À la figure 4.17, les triangles du plafond de la grotte possèdent une composante z s'approchant de -1 et ils sont représentés en rouge. Le domaine navigable (plancher en bleu foncé) possède quant à lui des composantes z près de 1. Le reste des cellules ayant des valeurs intermédiaires sont illustrées au moyen des autres couleurs présentes sur le terrain de la figure 4.17. L'étape subséquente est le rejet des cellules trop pentues.

Pour ce faire, les triangles possédant une composante z en deçà d'un seuil sont éliminés. La figure 4.18, montre le terrain de la figure 4.17 auquel les cellules possédant une composante z inférieure à 0.5 ont été éliminées.

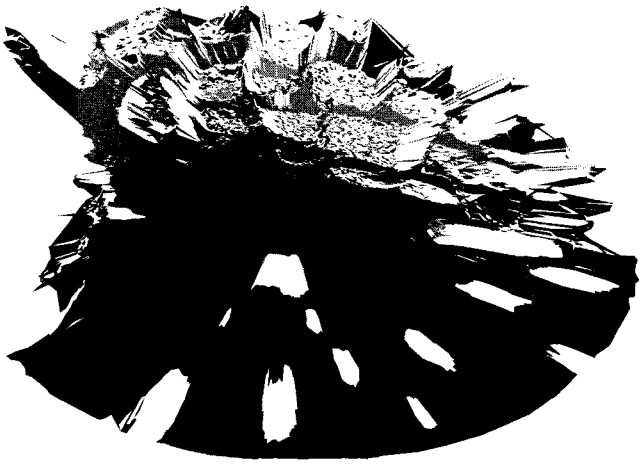


Figure 4.18 Exemple d'un maillage auquel les cellules ayant une composante z des vecteurs normaux inférieure à 0.5 ont été retirées. Chaque groupe de cellules est associé à une couleur.

Le plafond de la grotte a correctement été retiré. Toutefois, en enlevant les triangles trop inclinés, cela a créé des groupes de cellules déconnectés des uns aux autres. La figure 4.18 montre les différents groupes au moyen de couleurs. Ces groupes de cellules isolés, n'ayant pas de connectivité avec les triangles suppportant le robot, sont considérés inatteignables et sont rejetés. La figure 4.19 montre le maillage ne conservant que les triangles connectés aux cellules sous le robot.

Figure 4.19 Exemple d'un maillage auquel les groupes déconnectés ont été retirés

La figure 4.19 montre un modèle de terrain où les murs, les plafonds, les roches de dimensions importantes ainsi que les secteurs inatteignables ont été rejetés pour ne conserver que le domaine *navigable*.

4.5 Rééchantillonnage du maillage

La section 4.1 a mis en lumière le fonctionnement général du *LIDAR* en expliquant notamment que le capteur balaye les angles d'un repère sphérique par incrément angulaire constant. Cette procédure de mesure conduit à un maillage dont la résolution (distance entre 2 noeuds consécutifs) est constante dans le repère sphérique. Cependant, lorsque que les données sphériques sont converties dans le repère cartésien, la résolution devient varie en fonction de la distance par rapport au *LIDAR*. La figure 4.20 illustre cette situation.

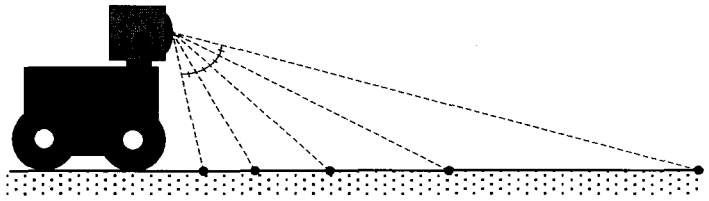


Figure 4.20 Nuage de points dont la densité est inversement proportionnelle à la distance du capteur

La figure 4.20 montre bien qu'une incrémentation constante de l'angle élévation conduit, dans le cas d'un terrain plat, à un nuage de points dont la résolution diminue en s'éloignant du capteur. Pour une mesure complète sur 360° en azimut, le maillage résultant contient une forte densité de cellules près du robot et moins de cellules plus loin. À titre d'exemple, la figure 4.21 est présentée. Celle-ci montre le maillage d'un terrain plat où la variation de densité de cellule est observable.

Figure 4.21 Exemple d'un maillage dont la densité diminue suivant la distance par rapport au capteur (centre du maillage). Le cercle pointillé blanc délimine un secteur contenant 80 % des triangles du maillage.

La forte densité de triangles à proximité du robot amène une précision excessive du modèle de terrain. En effet, une résolution trop élevée par rapport aux capacités du robot est inutile et alourdit les étapes subséquentes de calcul. Pour corriger cette variation de résolution, le maillage est rééchantillonné afin d'obtenir une densité uniforme de cellules.

La première étape est la construction d'une grille 2D cartésienne à résolution constante qui couvre tout le domaine x et y du maillage (voir figure 4.23a). Cette grille de points s'appelle *grille d'interpolation*. Ensuite, l'altitude (coordonnée z) de chaque point de la grille est déterminée par interpolation linéaire sur le maillage. Pour ce faire, il faut rechercher quel triangle du maillage supporte chacun des points de la grille (noeuds) et en déduire son altitude par interpolation (voir figure 4.23b). La figure 4.22 schématise ce calcul d'interpolation.

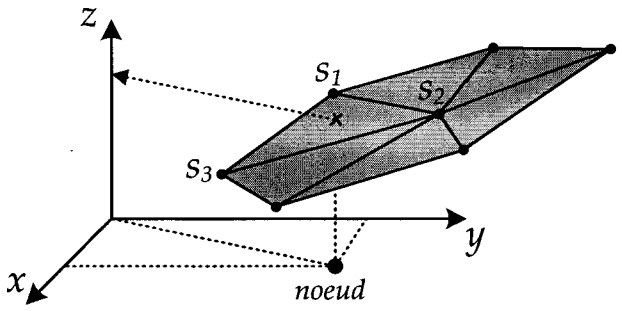


Figure 4.22 Schématisation du calcul d'interpolation sur un triangle

Lorsque le triangle supportant le noeud d'interpolation est connu, le calcul de l'altitude peut s'effectuer. D'abord, il faut calculer l'équation du plan qui engendre ce triangle. L'équation typique d'un plan est la suivante :

$$z = ax + by + c, \tag{4.4}$$

avec a, b et c des coefficients constants sur tout le plan. Ces coefficients s'obtiennent par la résolution du système d'équation ci-bas :

$$\begin{aligned} z_{s_1} &= ax_{s_1} + by_{s_1} + c, \\ z_{s_2} &= ax_{s_2} + by_{s_2} + c, \\ z_{s_3} &= ax_{s_3} + by_{s_3} + c. \end{aligned} \tag{4.5}$$

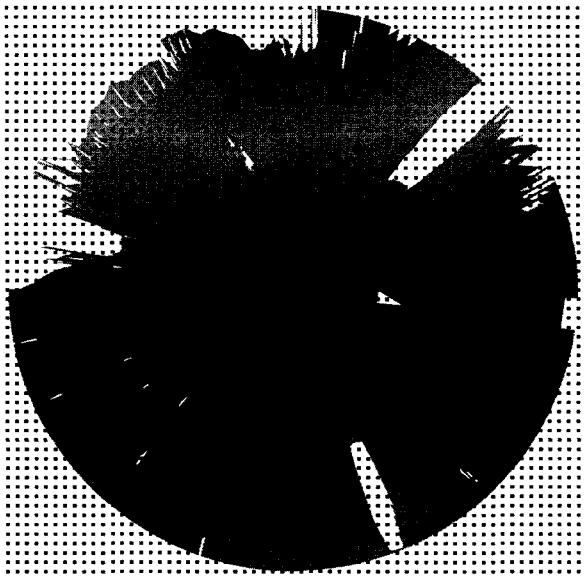
où x_{s_i}, y_{s_i} et z_{s_i} sont les coordonnées cartésiennes des sommets s_1, s_2 et s_3 du triangle.

Une fois que le système d'équations est résolu, les coefficients a, b et c sont connus pour ce triangle et il est possible de connaître l'altitude de tous points (x, y) projetés sur le plan. Enfin, l'altitude z_{ni} du noeud d'interpolation se calcule ainsi :

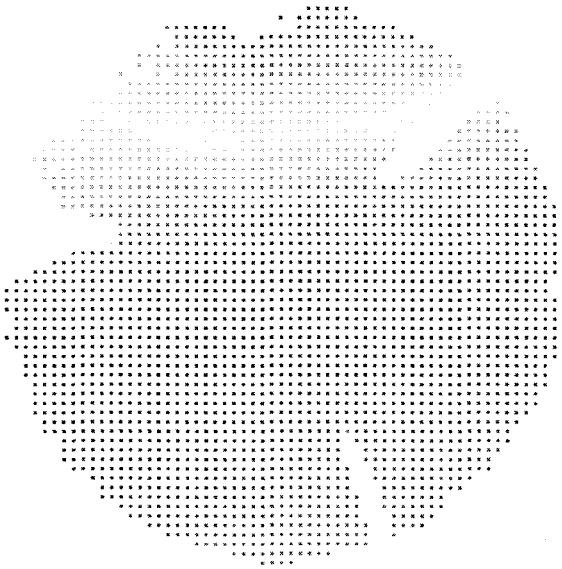
$$z_{ni} = ax_{ni} + by_{ni} + c. \tag{4.6}$$

Si pour un noeud, il n'existe pas de triangle pouvant le supporter, ce noeud est rejeté. Une fois l'altitude de tous les noeuds d'interpolation obtenue, la triangulation de Delaunay de ces points est effectuée (voir figure 4.23c). Tel qu'expliqué précédemment, l'algorithme de Delaunay remplit de triangles l'enveloppe convexe du nuage de points. C'est donc dire que les secteurs ombrés (où il n'y a pas de données) sont remplis. Toute-

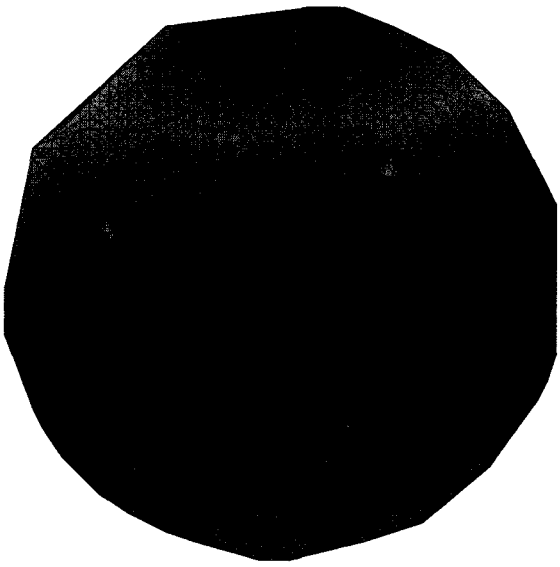
fois, ces triangles indésirables sont facilement détectables, car ils possèdent plus d'une arête dont la longueur dépasse la résolution de la grille fixée. La dernière étape menant au rééchantillonnage du maillage est donc le rejet des cellules indésirables. La figure 4.23 présente un exemple du rééchantillonnage d'un maillage.



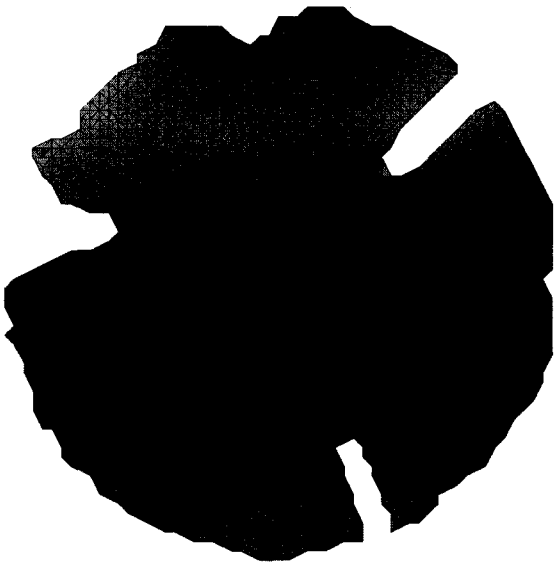
(a) Grille d'interpolation (résolution 30 cm)



(b) Altitude de la grille



(c) Triangulation de Delaunay de la grille



(d) Maillage dont les cellules ombrées ont été rejetées

Figure 4.23 Processus séquentiel de rééchantillonnage d'un maillage (la couleur est fonction de l'altitude)

4.6 Agrandissement des frontières

L'algorithme de planification de chemin doit ultimement générer un chemin qui maintient le robot à l'intérieur du maillage. Le chemin doit demeurer certes sur le modèle de terrain, mais ne doit pas s'approcher trop près des frontières. Un chemin frôlant les limites d'un maillage amène le robot à sortir en partie des frontières, car celui-ci possède une dimension non nulle. La figure 4.24 montre comment un chemin demeurant sur le modèle de terrain peut quand même faire sortir le robot du maillage.

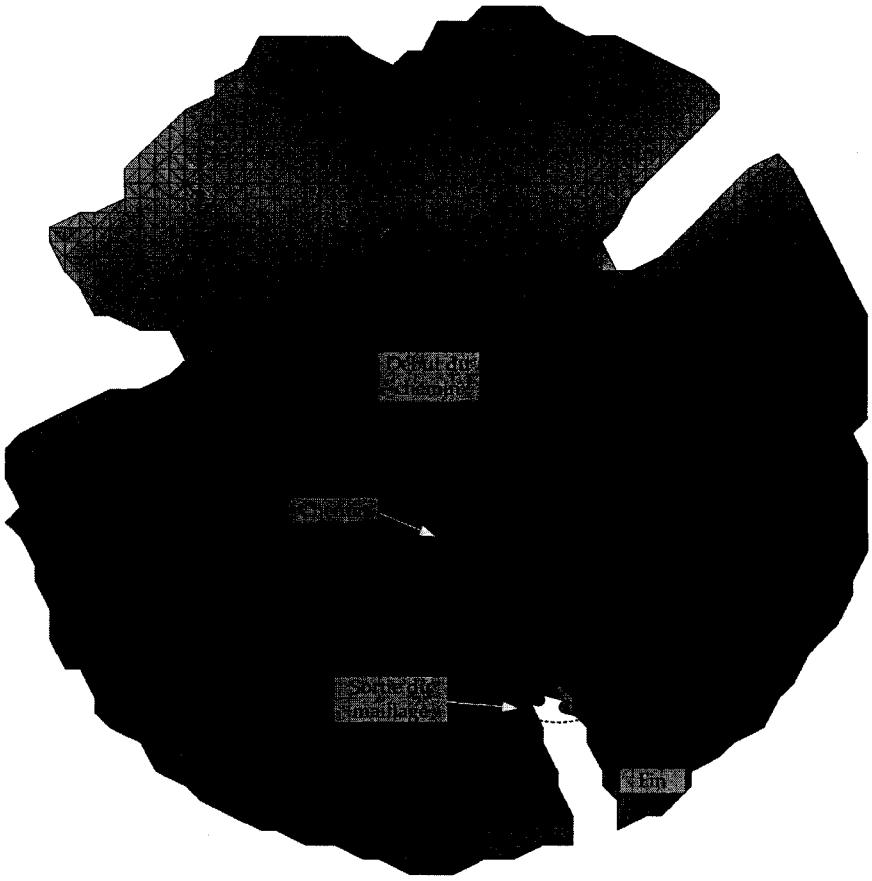


Figure 4.24 Exemple d'un chemin faisant sortir le robot du maillage

Afin d'éviter la situation présentée à la figure 4.24, l'algorithme de génération de chemin peut prendre en compte les dimensions du robot dans sa recherche. Cependant, cela ajoute des contraintes à la recherche de chemin et par le fait même complexifie l'algorithme. Une façon simple d'éviter la sortie du robot du maillage est de grossir suffisamment les frontières. Dès lors, le robot peut être considéré comme un point dans la recherche d'un chemin. Cette idée a été amenée par Thomas Lozano-Pérez [Lozano-Perez et Wesley, 1979] qui proposait de grossir les obstacles afin de négliger les dimen-

sions d'un robot dans le recherche d'un chemin libre d'obstacle. La figure 4.25 montre l'effet de l'agrandissement des frontières sur la planification de chemin.

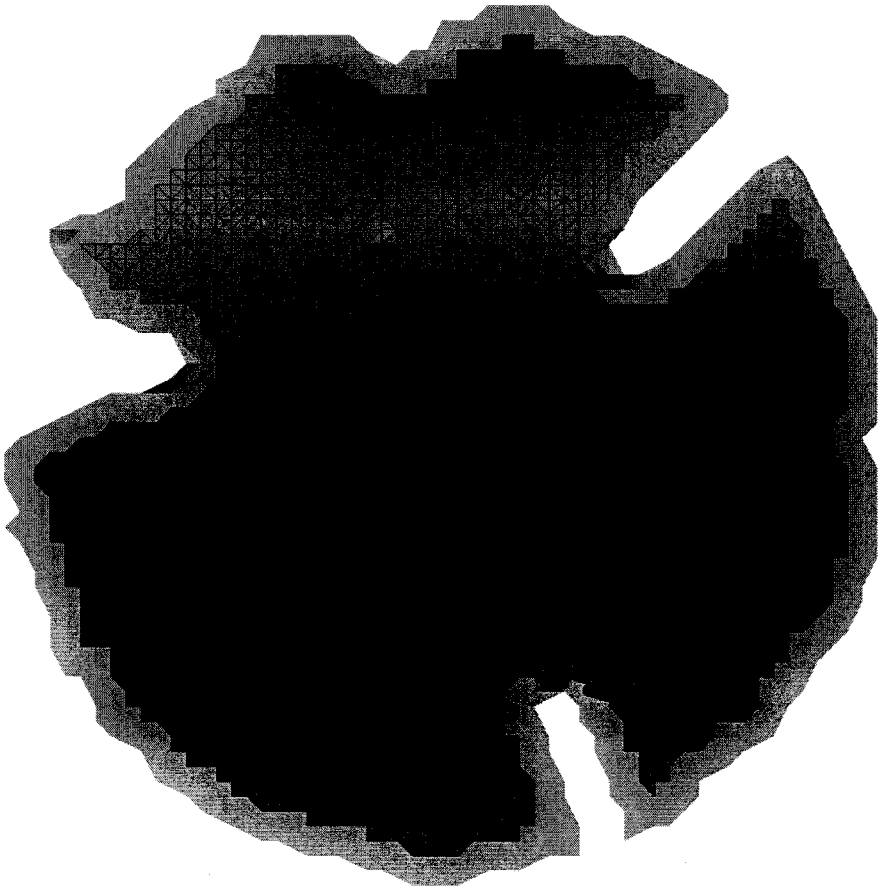


Figure 4.25 Exemple d'un chemin conservant le robot sur le maillage (délimité par le secteur ombré) malgré la proximité du chemin avec les frontières du maillage servant à la planification de chemin (secteur coloré)

La figure 4.25 illustre une situation dans laquelle le planificateur de chemin utilise le maillage dont les frontières ont été grossies. L'algorithme peut dès lors négliger les dimensions du robot. Il a retourné un chemin qui frôle les frontières de son maillage de travail mais qui conserve le robot à l'intérieur du domaine mesuré par le capteur *LIDAR* (secteur ombré à la figure 4.25).

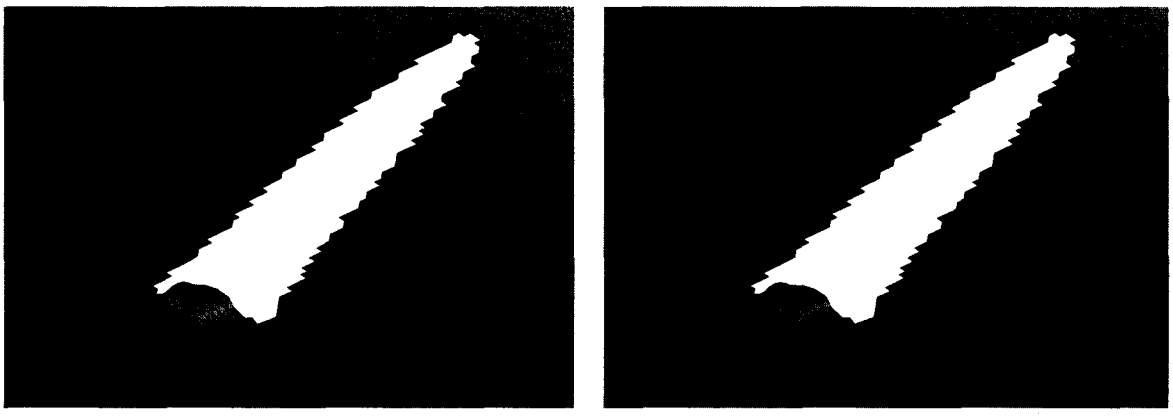
4.7 Conditionnement du maillage

Les exemples des sections 4.6 et 4.5 montrent des maillages dont la résolution est grossière (30 cm) afin de bien montrer les triangles. Toutefois, ceux-ci ont une précision

insuffisante pour une application de guidage d'un robot mobile tel que celui considéré dans ce présent projet. Dans le cas du banc d'essai de l'ASC, le robot est capable de franchir une rugosité de l'ordre de 5 cm. C'est cette valeur qui permet de fixer l'ordre de grandeur de la résolution des maillages. Maintenant, avec une résolution de 5 cm et un rayon limite de 6 m, un tel maillage peut contenir environ 100 000 cellules, ce qui est beaucoup pour les algorithmes qui utilisent ce modèle de terrain. Tel qu'expliqué précédemment, la complexité des méthodes de génération de chemin est fonction du nombre de cellules. Ainsi, plus il y a de triangles dans un maillage, plus l'étape de planification de chemin est lente.

4.7.1 Réduction du nombre de cellules

La littérature relate plusieurs méthodes qui peuvent être utilisées afin de réduire le nombre de triangles que contient un maillage sans perdre l'information utile. L'article de Cignoni [Cignoni *et al.*, 1998] propose une comparaison assez exhaustive des approches de simplification de maillage. L'algorithme retenu fait parti de la famille de méthode *vertex clustering* qui regroupe de façon itérative plusieurs noeuds d'un maillage en un nouveau noeud représentatif (appelé en anglais un *cluster*). Galand et Heckbert ont proposé l'algorithme *QSlim* [Garland et Heckbert, 1997] capable de simplifier un maillage de façon efficace en terme de charge de calcul [Cignoni *et al.*, 1998]. La figure 4.26 montre un maillage auquel le filtre *QSlim* a réduit de 90 % le nombre de triangles.



(a) Maillage initial dont la résolution est constante (5 cm)

(b) Maillage simplifié dont le nombre de triangles a été réduit de 90 %

Figure 4.26 Exemple de l'application d'un filtre de simplification sur un maillage

Le filtre *QSlim* est capable de réduire de façon importante la quantité de cellules d'un maillage tout en préservant le relief, comme en témoigne la figure 4.26.

4.7.2 Adoucissement laplacien

La dernière étape de ce traitement est un filtrage ayant pour but de réduire le bruit aléatoire de mesure et d'augmenter la qualité du maillage. Pour ce faire, un filtre laplacien est utilisé. Celui-ci déplace les noeuds d'un maillage de façon itérative dans la direction suivie par un processus de diffusion obéissant à l'équation :

$$\frac{\partial}{\partial t}X = \lambda \nabla^2 X, \tag{4.7}$$

où X représente les coordonnées des noeuds, ∇^2 est l'opérateur laplacien et λ est un paramètre régissant la vitesse de diffusion. Cette méthode est largement utilisée et est une des plus rapides [Belyaev et Ohtake, 2003]. Elle permet en quelques itérations de réduire le niveau de bruit en agissant comme un filtre passe-bas. En plus, elle permet généralement d'augmenter la qualité d'un maillage en favorisant l'atteinte de cellules dont la forme s'approche d'un triangle équilatéral. Ce filtre ne modifie pas la connectivité d'un maillage et ne change pas le nombre de noeuds. Il ne fait que déplacer légèrement la position des sommets des triangles.

4.8 Synthèse de la modélisation de terrain

Les données brutes reçues du capteur *LIDAR* sont d'abord traitées par un filtre médian qui rejette les données aberrantes. L'usage de cette méthode de filtrage adaptatif requiert une transformation des données brutes sphériques en une forme équivalente d'image en teintes de gris.

Puis, la triangulation 2D de Delaunay est construite à partir des coordonnées angulaires du nuage de points sphériques. Les coordonnées des sommets des triangles sont ensuite transformées vers un repère de travail cartésien. Comme l'algorithme de Delaunay retourne une triangulation continue, les discontinuités dans le nuage de points ont été remplies de cellules. Ces triangles indésirables sont ensuite détectés puis éliminés.

Ultimement, le maillage alimente une application de guidage qui recherche un chemin sécuritaire à l'intérieur du modèle de terrain. C'est dans l'esprit de promouvoir la sécurité du *rover* que les cellules trop pentues sont rejetées de la triangulation. C'est au

moyen de l'analyse des vecteurs normaux aux cellules que les triangles fautifs sont décelés. Toutefois, le rejet des cellules inclinées peut conduire à une quantité importante de regroupement de triangles déconnectés les uns des autres. L'étape subséquente est l'extraction du groupe de cellules connectées à celles qui supportent le robot pour ne conserver que le domaine navigable.

Ce domaine navigable est supporté par une triangulation dont la densité de cellules n'est pas constante. Effectivement, à proximité du *LIDAR*, la résolution est élevée et décroît suivant la distance du capteur. Pour corriger cette inconstance dans la distribution des cellules, le maillage est rééchantillonné par interpolation linéaire sur une grille cartésienne à résolution constante.

Comme le robot possède des dimensions physiques non nulles, il est possible qu'un générateur de chemin propose un chemin demeurant sur le maillage mais qui fait sortir une partie du robot. Cette situation n'étant pas sécuritaire, les cellules à proximité des frontières sont retirées. Dès lors, l'algorithme de génération de chemin peut négliger les dimensions du robot mobile dans sa recherche.

La dernière étape consiste en un conditionnement du maillage. D'abord le nombre de cellules du maillage est réduit au moyen d'un algorithme de simplification. Enfin, le modèle de terrain est filtré au moyen d'un filtre laplacien qui déplace légèrement les noeuds afin de réduire le bruit et d'augmenter la qualité du maillage.

La figure 4.27 propose un schéma synthétisant l'approche de modélisation de terrain expliqué précédemment.

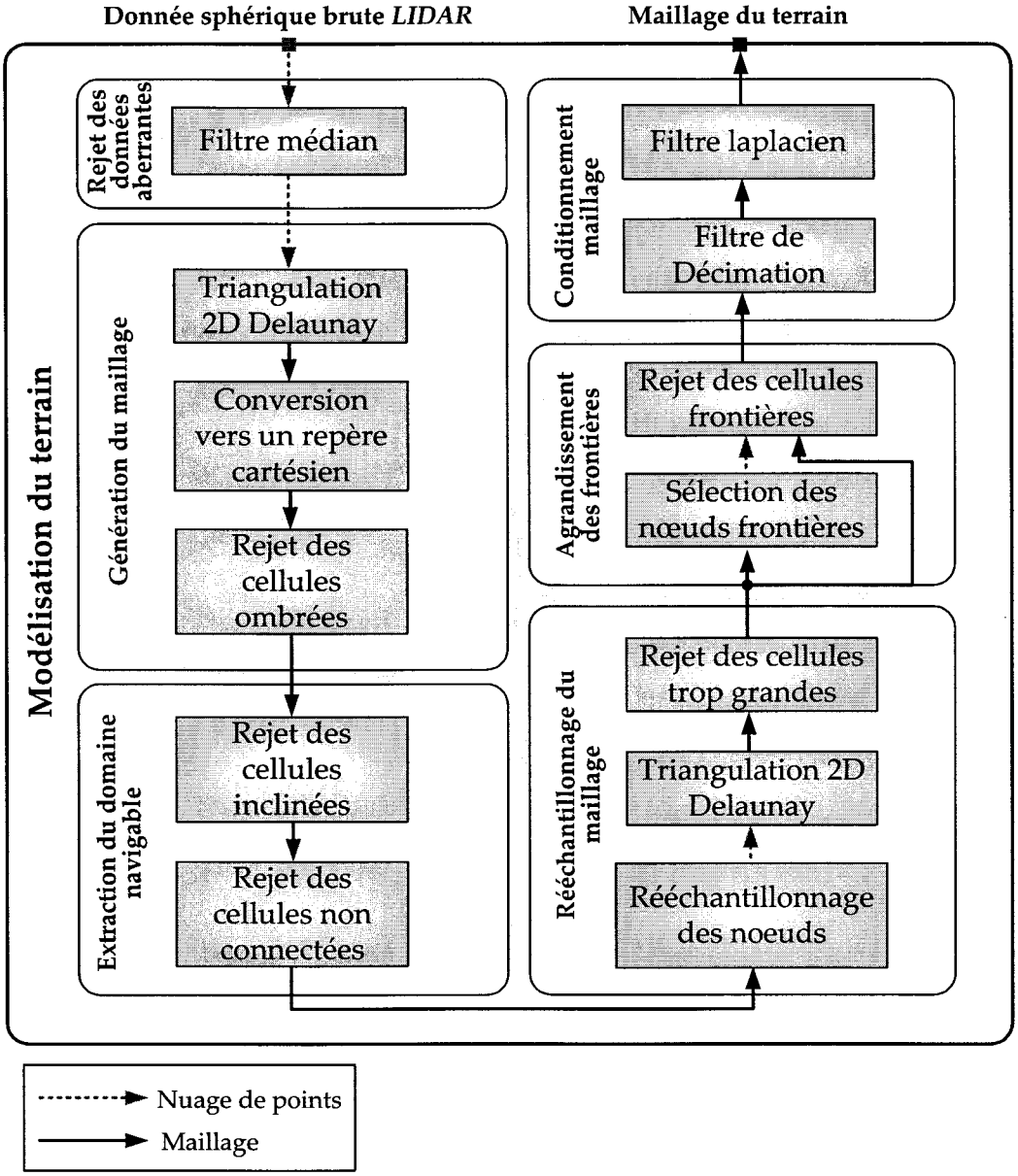


Figure 4.27 Diagramme de flux de la modélisation de terrain

La figure 4.28 présente un nuage de points brut superposés au modèle de terrain généré par l'approche de modélisation précédemment expliquée.

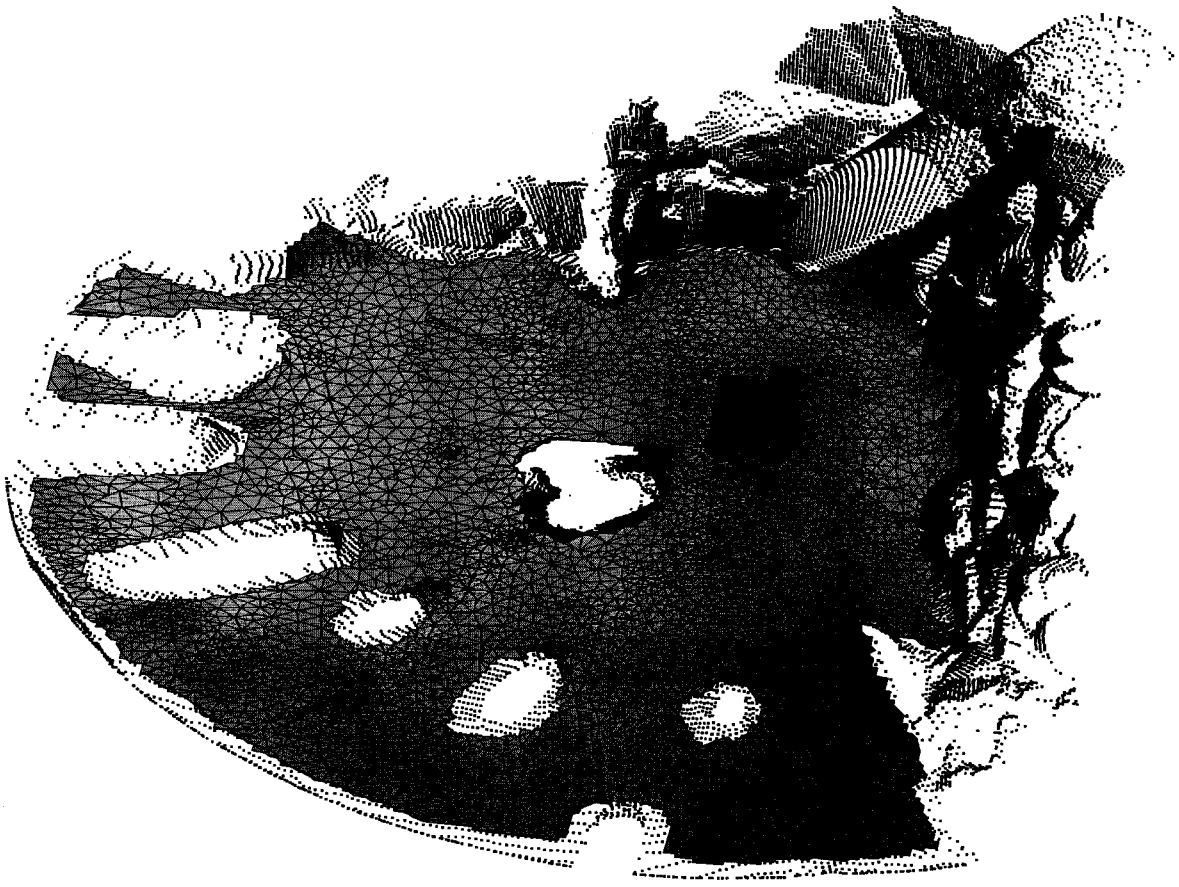


Figure 4.28 Comparaison entre un nuage de points brut et la modélisation de terrain associée

CHAPITRE 5

PLANIFICATION DE CHEMIN

Ce chapitre présente la stratégie retenue pour la planification de chemin exposée à la section 3.1. La première section expose et justifie les hypothèses de travail permettant la simplification du problème. Par la suite, une approche connue et largement utilisée est présentée. Celle-ci permet d'évaluer de façon comparative la performance de la méthode développée. Puis, l'algorithme principal de ce projet de recherche est présenté. Les algorithmes sont ensuite appliqués à des situations concrètes permettant de démontrer les forces et faiblesses de ceux-ci. Enfin, la dernière section montre comment un croisement entre la méthode de référence et la méthode développée permet d'atteindre des performances supérieures.

5.1 Simplification du problème

Un *rover* réel est une machine complexe et l'environnement du robot l'est tout autant. Il est important de définir dès le début jusqu'à quel niveau de détail l'algorithme doit travailler. Est-ce qu'il est pertinent de considérer le robot dans ses moindres détails (dynamique, rigidité, interaction entre les roues et le sol, etc.) ? Est-ce acceptable de considérer le robot uniquement comme un point ? Qu'est-ce qu'un obstacle ? Cette section expose les hypothèses permettant de répondre à ces questions.

5.1.1 Hypothèses liées au modèle de terrain

Tel qu'expliqué à la section 4.1, le modèle de terrain a été bâti uniquement au moyen d'un nuage de points obtenu d'un *LIDAR*. Aucune information permettant l'analyse de la composition ou de la nature du sol n'est disponible. Le modèle de terrain suppose donc une homogénéité dans la composition de l'environnement. Les roches, le sable, les trous et tout autre objet présents au moment de la prise de mesure sont considérés uniformes et de même nature. De plus, tout l'environnement est supposé statique et indéformable. Une modélisation de terrain ne contenant que des données 3D contient généralement jusqu'à trois types d'obstacles :

1. *Rugosité* : les roches, les trous ou tout autre changement excessif d'altitude du sol forment ce type d'obstacle ;
2. *Pente* : les secteurs du terrain où la pente moyenne est trop élevée pour les capacités du robot. Cela comprend les pentes abruptes infranchissables et aussi les pentes transversales qui peuvent engendrer un culbutage du robot ;
3. *Méconnaissance* : les secteurs du maillage dépourvus d'information (p. ex., les ombres) sont considérés comme des obstacles.

Le traitement de maillage présenté à la section 4.4 retire de la triangulation les cellules trop pentues. Les obstacles de pente peuvent ainsi être négligés dans l'algorithme de génération de chemin.

5.1.2 Modélisation du robot

La géométrie d'un robot peut être complexe. Par exemple, elle peut contenir des pattes, des antennes, des caméras, etc. Une approche conservatrice de modélisation consiste en l'approximation de la géométrie complexe par une forme simple qui englobe la totalité du robot. La figure 5.1 montre un schéma du robot de l'ASC ainsi que son approximation géométrique.

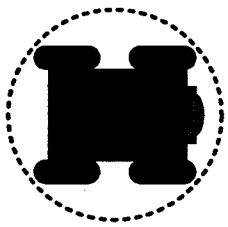


Figure 5.1 Approximation géométrique du robot

C'est donc au moyen d'un cercle que le robot est modélisé. Les dimensions du cercle ne changent pas, car le robot est supposé rigide.

L'algorithme à développer doit fournir un chemin et non une trajectoire faisant intervenir le temps[1]. Les propriétés dynamiques du robot peuvent donc être négligées dans la recherche d'un chemin. La tâche de convertir le chemin en trajectoire revient au système de commande. C'est celui-ci qui doit prendre en compte la dynamique du robot dans l'émission des consignes de positionnement dans le temps.

[1] Voir section 2.1 pour la définition de trajectoire

5.2 Méthode de recherche de graphe

La méthode de génération de chemin basée sur une recherche de graphe est une des approches les plus largement utilisées en robotique. La NASA a implémenté un tel algorithme au sein du planificateur global de chemin des robots MER [Carsten *et al.*, 2007]. La compagnie conceptrice des bras canadiens robotisés MacDonald, Dettwiler and Associates (MDA) utilise aussi ce type d'approche [Liu *et al.*, 2008] de même que l'ASC [Rekleitis *et al.*, 2009]. Cette section présente un algorithme de recherche de chemin développé par l'ASC qui utilise la recherche de graphe. C'est cette méthode qui permet d'obtenir les résultats d'étalonnage servant de référentiel comparatif.

5.2.1 Détails de la méthode

L'essentiel des explications qui suit est tiré de l'article *Path Planning for Planetary Exploration* [Rekleitis *et al.*, 2008a]. L'algorithme prend en entrée le maillage triangulaire du terrain, la position initiale du robot, la destination ainsi qu'une série de paramètres liés à la modélisation du robot. Le résultat est un chemin qui s'exprime comme une liste de points 3D à atteindre.

Graphe de connectivité

L'algorithme construit d'abord un graphe de connectivité à partir du maillage. Les cellules du maillage forment les noeuds du graphe et les arêtes communes entres les cellules deviennent le lien de connectivité entre les noeuds. La figure 5.2 illustre ce concept.

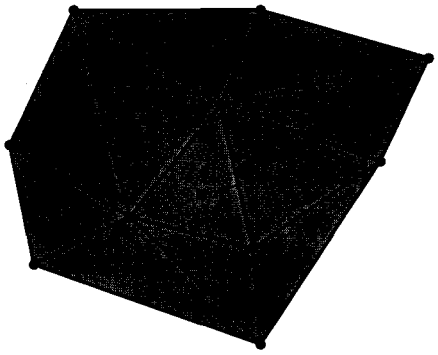


Figure 5.2 Exemple d'un graphe de connectivité associé à un maillage de terrain (les carrées rouges sont les noeuds du graphe et les lignes pointillées sont les arêtes de connectivité)

Fonction Coût

Une fois le graphe construit, il est ensuite possible d'associer un *coût* à chacune des arêtes de connectivité. Ce coût permet de quantifier la facilité avec laquelle le robot peut se déplacer d'une cellule à l'autre. Dans le cas d'un lien entre deux triangles plats, un faible coût peut y être assigné. Le coût pour passer d'une cellule à l'autre peut aussi devenir infini si physiquement il est impossible pour le robot de suivre cette connectivité. Le problème à résoudre comporte plusieurs contraintes telles que la distance, la pente, la rugosité, etc. Pour chacunes des contraintes d'optimisation, il est possible de bâtir une fonction coût. Une première fonction coût peut être basée uniquement sur la distance entre deux cellules. De cette façon, l'algorithme de recherche favorise les chemins courts. Le coût Q_d associé à une arête de connectivité arbitraire s'exprime mathématiquement ainsi :

$$Q_d = \|X_i - X_j\|, \tag{5.1}$$

avec X_i et X_j les vecteurs positions du centre des cellules i et j. Cette fonction coût permet une certaine minimisation de l'énergie, mais ne prend pas en compte les contraintes physiques du robot. Une seconde fonction peut être basée à la fois sur la distance et sur les pentes du terrain. Le calcul s'effectue ainsi :

$$Q_{d\&p} = \alpha\beta \|X_i - X_j\|, \tag{5.2}$$

où α et β sont des coefficients pénalisant la pente moyenne du terrain suivant l'arête de connectivité. Le détail du calcul de ces coefficients est détaillé dans l'article de Rekleitis [Rekleitis *et al.*, 2008a]. Celui-ci considère de façon séparée les angles de roulis et tangage qu'implique le terrain sur le robot et il associe une valeur infinie aux coefficients aussitôt que les pentes dépassent les contraintes physiques du robot. Cette fonction coût est incomplète, car elle ne considère pas la rugosité du sol qui peut entrer en interférence avec le robot. Tel qu'expliqué à la section 5.1.2, le robot est représenté par un cercle. La prochaine fonction coût Q_e mise à l'étude à l'ASC prend en compte la rugosité du terrain à l'intérieur de ce cercle et est formulée ainsi :

$$Q_e = \alpha\beta \|X_i - X_j\| \gamma e^{\frac{\|X_i - X_j\|}{A_i + A_j}}, \tag{5.3}$$

avec γ un paramètre qui dépend de la rugosité, A_i et A_j l'aire des cellules i et j. Le terme exponentiel permet de favoriser un chemin passant par des cellules larges au lieu de zigzaguer à travers les petits triangles. Le coefficient γ vaut 1 si la rugosité n'est pas excessive et vaut l'infini dans le cas contraire. Des essais physiques sur le banc d'essai de l'ASC ont montré que c'est la fonction coût Q_e qui permet d'obtenir les meilleurs résultats. C'est donc celle-ci que ce projet de recherche utilise pour la suite des calculs.

Recherche dans le graphe

Maintenant que la fonction coût est déterminée pour l'ensemble du graphe, il reste à utiliser un algorithme de recherche de graphe permettant d'obtenir le chemin le moins coûteux. Un des algorithmes les plus utilisés pour ce type de recherche est l'algorithme de Dijkstra publié dans l'article *A note on two problems in connexion with graphs* [Dijkstra, 1959]. Cette méthode de recherche explore le graphe et retourne le chemin minimisant la fonction coût. C'est une solution dite *optimale* qui est obtenue par la méthode de Dijkstra.

Il a été démontré que cette approche de résolution est relativement lente. Tel qu'expliqué à la section 2.2.3, sous certaines conditions, cet algorithme possède une complexité de $O(n^2)$ (n est le nombre de noeuds du graphe). Il est possible de réduire le nombre d'opérations exécuté en ajoutant une heuristique à la recherche. L'idée est de guider la recherche dans une direction qui rapproche de la destination. Pour ce faire, la fonction coût est modifiée ainsi :

$$Q_{A^\star} = Q_e + Q_h. \tag{5.4}$$

Le terme heuristique Q_h se calcule de cette façon :

$$Q_h = \|X_j - X_f\|, \tag{5.5}$$

où X_f est le vecteur position de la cellule qui supporte la destination à atteindre. Cette méthode de recherche de graphe guidée s'appelle $A^\star$. Elle permet une recherche accélérée qui n'est toutefois plus optimale au sens mathématique.

5.2.2 Exemples d'utilisation

La section 5.2.2 présente quelques exemples d'utilisation de l'algorithme de recherche $A^\star$ sur des données obtenues du banc de test de l'ASC. Dans chacun des exemples, le maillage est généré au moyen de l'approche exposée au chapitre 4. Le premier cas prend naissance en terrain plat et dépourvu d'obstacle. Le robot est initialement positionné au centre du maillage et cherche à atteindre une destination devant lui. La figure 5.3 presente le résultat de ce premier exemple.

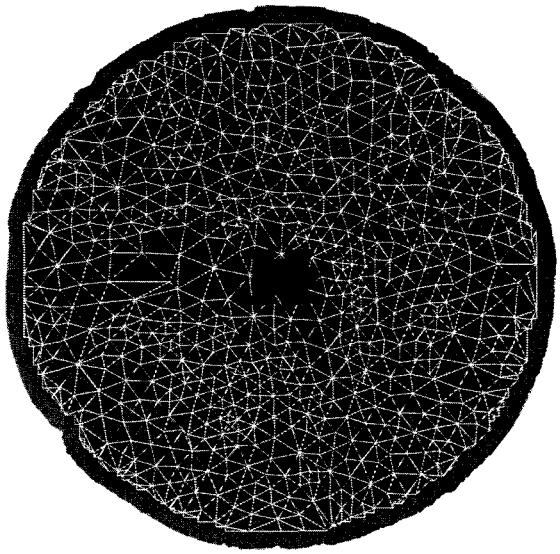


Figure 5.3 Planification de chemin au moyen de l'algorithme de $A^\star$ sur un terrain plat sans obstacle (le maillage brut du terrain est en couleur et possède un rayon de 3 m, les arêtes du maillage ayant servi au calcul de chemin sont en blanc, le chemin obtenu est en noir et la destination est au point rouge)

Le chemin proposé atteint la destination et semble acceptable. Il passe par le centre de chacune des cellules croisant son passage tel qu'imposé par la méthode. Le second exemple exposé utilise un maillage plus grand comportant de nombreux obstacles. La figure 5.4 montre le résultat.

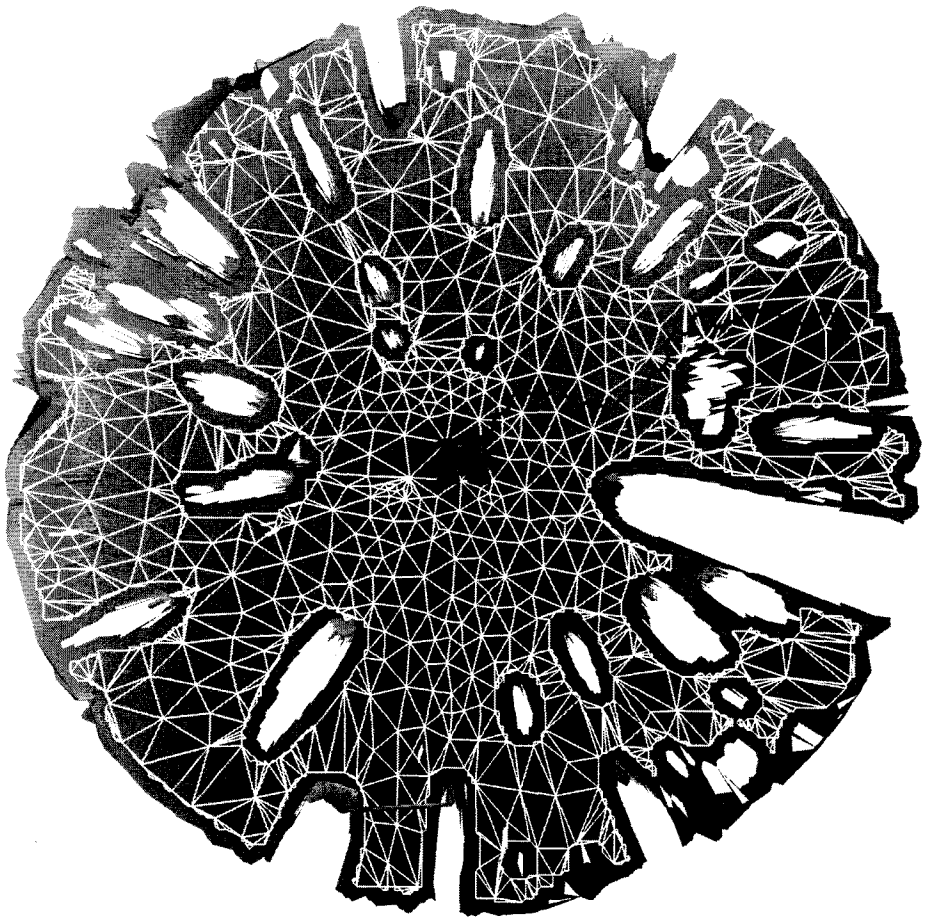


Figure 5.4 Planification de chemin au moyen de l'algorithme de A^* sur un terrain plat avec obstacles (le maillage brut du terrain est en couleur et possède un rayon de 6 m, les arêtes du maillage ayant servi au calcul de chemin sont en blanc, le chemin obtenu est en noir et la destination est au point rouge)

5.2.3 Conclusion sur la méthode de recherche de graphe

Les deux chemins présentés en exemple suivent des directions relativement acceptables, toutefois le détail du tracé rend ces chemins inutilisables directement par un robot mobile. De tels chemins en *zigzag* peuvent occasionner des problèmes notamment au niveau de la localisation du robot. En effet, chaque fois que le robot change d'orientation, celui-ci glisse et le système d'odométrie accumule davantage d'erreurs. Un chemin efficace limite les courbures d'un chemin et, idéalement, proscrit les changements vifs de direction. Certains utilisateurs des méthodes de graphe voient dans le lissage (p. ex. *B-spline*, interpolation cubique, etc.) la solution au problème de *zig-*

zag. Le problème est que rien ne garantit que le lissage d'un chemin sécuritaire permet d'obtenir à nouveau un tracé sécuritaire.

Le fond du problème de la méthode de graphe vient de l'imposition de contraintes artificielles dans le calcul d'optimisation. Une recherche limitée au graphe de connectivité limite certes la dimension de recherche et par le fait même le temps de calcul, mais amène une dépendance de la solution au conditionnement du maillage. Un maillage bien conditionné avec des triangles de dimensions similaires conduit généralement à une solution acceptable retournée par $A^{\star}$. Dans le cas contraire, c'est-à-dire avec un maillage mal conditionné possédant des cellules déformées, le chemin obtenu est inutilisable directement.

La méthode offre toutefois une souplesse intéressante quant à la nature des chemins recherchés. Par exemple, il est possible pour le robot de demeurer en haute altitude afin de conserver un lien de communication. Dans ce cas, il suffit d'adapter la fonction coût en récompensant les cellules à haute altitude. Il est aussi possible de forcer le robot à se déplacer en territoire exposé au soleil si l'information d'ensoleillement est disponible. La méthode de recherche de graphe offre des avantages certains, c'est ce qui explique sa grande popularité. La section suivante présente une méthode qui permet de générer des chemins sans l'imposition de contrainte artificielle reliée au maillage.

5.3 Approche par mécanique des fluides

La présente section expose le coeur des travaux de ce projet de recherche qui consiste en l'amélioration des techniques de planification de chemins pour un robot mobile explorateur de planète. Une méthode capable de retourner une solution indépendante du conditionnement du maillage est à privilégier. Celle-ci doit aussi générer des chemins lisses, continus et dépourvus de changements vifs de direction en plus de respecter la totalité des critères présentés au tableau 3.1. L'atteinte de ces critères est possible notamment au moyen d'une analogie à la mécanique des fluides.

L'approche associe au modèle de terrain un bassin dans lequel entre un débit de fluide à la position du robot et où ce même débit quitte le bassin à la destination à atteindre. Les lignes d'écoulement du fluide de la source jusqu'à la sortie représentent une famille de chemins candidats que le robot peut suivre pour atteindre la destination. Les prochaines sections détaillent cette approche innovatrice, une des principales contributions de ce projet de recherche.

5.3.1 Théorie sur les écoulements potentiels

La vitesse en tout point d'un écoulement est la variable d'état à calculer. L'écoulement est supposé stationnaire et incompressible. La viscosité du fluide est de plus négligée. Ce type d'écoulement est complètement caractérisé au moyen d'une équation de Poisson :

$$-\nabla^2\phi = q(x,y). \tag{5.6}$$

L'annexe A démontre, à partir de la première loi de la thermodynamique et de quelques hypothèses, comment l'équation 5.6 permet de résoudre l'écoulement non visqueux de tout fluide incompressible. La résolution d'une telle équation requiert l'imposition de conditions aux frontières. Ce sont des conditions de Neumann homogènes qui sont appliquées. Ce type de condition permet de garantir l'étanchéité des parois et donc aucune particule ne peut s'échapper du bassin. En langage mathématique ces conditions de frontière s'expriment ainsi :

$$\left.\frac{\partial\phi}{\partial n}\right|_{\Gamma} = 0, \tag{5.7}$$

où Γ représente les frontières et n est une direction normale aux frontières. Le potentiel ϕ étant l'unique variable à calculer, il est nécessaire d'imposer des valeurs sur $q(x,y)$ en tout point du domaine. Cette fonction scalaire permet de prendre en compte les sources externes de fluide. Par convention, q prend la valeur d'une *distribution de Dirac* positive δ dans le cas d'un ajout constant de fluide (source) et une valeur négative $-\delta$ pour une sortie de fluide (puits). Ailleurs sur le domaine, q est nul. La figure 5.5 présente une schématisation du problème d'écoulement potentiel à résoudre.

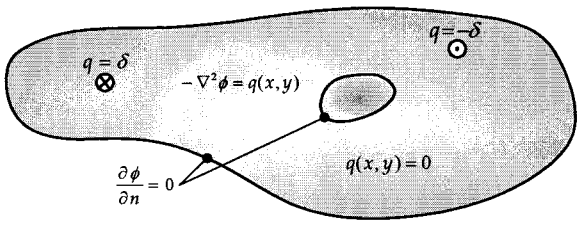


Figure 5.5 Schéma du problème d'écoulement de fluide à résoudre (le bassin est en bleu, la source est symbolisée par $\otimes$ et le puits par $\odot$)

La résolution du problème illustré à la figure 5.5 peut s'accomplir au moyen de plusieurs méthodes. Dans le cas d'une géométrie simple, il est possible d'utiliser des approches analytiques. Le livre de Kreyszig [Kreyszig, 1993] montre comment résoudre une fonction de potentiel sur un domaine rectangulaire au moyen des séries de Fourier. Cependant, cette méthode n'est pas utilisable dans le cas des domaines complexes tels que ceux rencontrés par les robots mobiles en terrain extérieur. Les approches numériques telles que les *éléments de frontières* ainsi que les *éléments finis* se prêtent davantage à un domaine discret tel qu'un maillage triangulaire.

Tel qu'expliqué à l'Annexe A, une fois le potentiel calculé sur tout le domaine, le champ de vitesse $\vec{V}$ s'obtient du gradient négatif du potentiel :

$$\vec{V} = -\vec{\nabla}\phi. \tag{5.8}$$

5.3.2 Résolution numérique du potentiel

La *méthode des éléments de frontières* est une approche numérique de résolution d'équations aux dérivées partielles qui utilise le *théorème de Green* afin de réduire la dimension d'un problème. Par exemple, l'analyse d'un volume peut être réduite à l'analyse d'une surface et un problème de surface peut se ramener à une ligne. Selon l'auteur du livre *The finite element method for engineers* [Huebner, 2001], cette méthode a connu un certain succès dans le domaine de l'acoustique. Toutefois, elle demeure peu utilisée par l'industrie, car d'autres méthodes concurrentes s'avèrent plus efficaces.

La *méthode des éléments finis* (MEF) est une technique d'analyse numérique permettant d'obtenir une solution approximative à une vaste gamme de problèmes d'ingénierie. Initialement développée pour l'étude des structures complexes d'aéronautique, la méthode a depuis été étendue à la plupart des domaines liés aux *milieux continus*. Depuis les années 1960, des milliers d'articles ont été publiés sur le sujet. *Zienkiewicz* propose un sommaire exhaustif de l'état de l'art de la méthode des éléments finis dans l'article *The generalized finite element method - state of the art and future directions* [Zienkiewicz, 1983]. C'est cette approche qui est utilisée pour résoudre l'équation 5.6 sur un maillage triangulaire.

Un milieu continu de toute dimension possède un champ de variable (potentiel, vitesse, pression, etc.) s'exprimant en un nombre infini de valeurs. En conséquence, le problème a un nombre infini de variables à résoudre. La discrétisation d'un domaine

en éléments finis permet de réduire le nombre d'inconnues à un nombre fini qui dépend du nombre de noeuds d'un maillage, du nombre de degrés de liberté de chaque noeud et de l'ordre de précision du schéma de résolution. L'équation de Poisson est un des problèmes les plus courants en analyse numérique. Il existe une grande quantité d'algorithmes disponibles pour la résolution d'une fonction de potentiel par éléments finis. Le livre *The finite element method - Fluid dynamics* [Zienkiewicz et Taylor, 2000] propose un code écrit en language *Fortran* capable de résoudre ce type d'écoulement. Ce projet a toutefois utilisé un code fourni gracieusement par le professeur Guy Payre[2]. L'algorithme reçu fut initialement écrit en *Fortran* et a été par la suite traduit en langage *C* compatible avec le logiciel *Matlab*™.

À titre d'exemple d'utilisation, la figure 5.6 montre un maillage de terrain auquel une source et un puits sont appliqués sur un noeud. C'est sur ce domaine que la méthode des éléments finis résoud la fonction de potentiel. La solution est présentée à la figure 5.7.

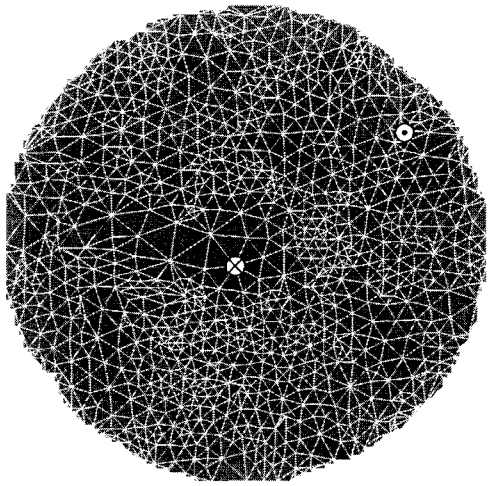
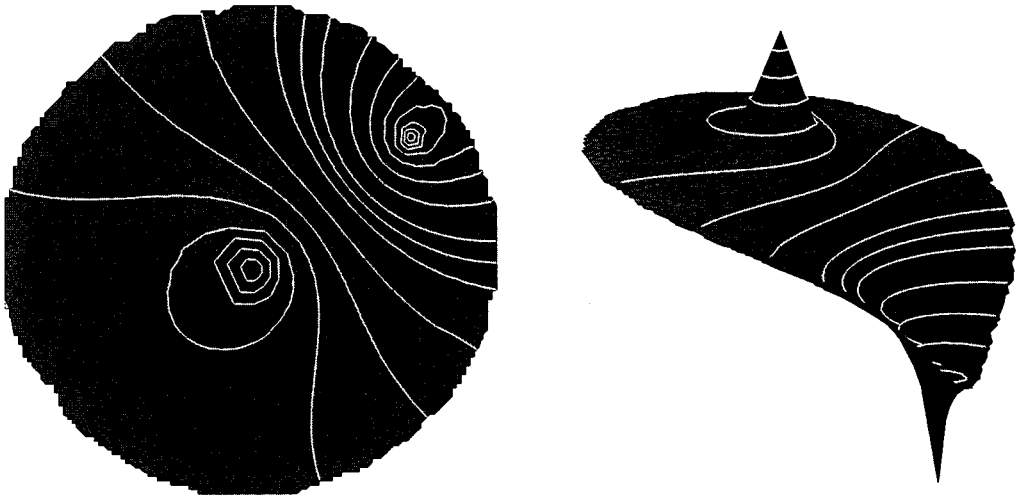


Figure 5.6 Exemple d'un problème de potentiel à résoudre par la MEF sur un maillage triangulaire (le bassin est en bleu, la source est symbolisée par ⊗ et le puits par ⊙)

La figure 5.7 montre une solution ne possédant qu'un seul minimum situé au noeud supportant le puits. L'absence de minimum local est une propriété des fonctions harmoniques. L'auteur de l'article *Robot path planning using fluid model* [Li et Bui, 1998] rappelle cette propriété qui permet d'éviter de coincer le robot dans un minimum. La section 2.2.4 détaille l'importance d'utiliser une fonction de potentiel dépourvue d'extremum locaux.

[2]Guy Payre, professeur en mathématique à la Faculté de génie de l'Université de Sherbrooke



(a) Vue de haut (b) Vue isométrique

Figure 5.7 Résultat du potentiel calculé aux noeuds par la MEF (les lignes blanches sont des courbes de niveau)

5.3.3 Calcul du champ de vitesse

Tel qu'expliqué antérieurement, le but de tous ces calculs est d'obtenir un champ vectoriel permettant de guider le robot entre sa position initiale est sa destination. C'est le champ de vitesse associé à l'écoulement potentiel qui sert de guide au robot. L'équation 5.8 permet d'obtenir ce champ de vitesse à partir de la fonction de potentiel. Il suffit de calculer le gradient négatif du potentiel pour calculer le vecteur vitesse du fluide en un point du domaine. Le livre *The finite element method in engineering* [Rao, 2005] propose une méthode pour calculer le gradient d'une fonction sur un élément triangulaire. Une fonction d'interpolation permet d'estimer la valeur du potentiel en tout point de l'élément. La méthode d'*interpolation de Lagrange* se prête bien à ce type de calcul et s'exprime ainsi pour un élément linéaire possédant les sommets i, j et k :

$$\phi^e(x,y) = \phi_i N_i^e(x,y) + \phi_j N_j^e(x,y) + \phi_k N_k^e(x,y), \tag{5.9}$$

où $\phi^e(x,y)$ est la valeur continue du potentiel sur l'élément e, ϕ_i est la valeur discrète du potentiel au noeud i et $N_i^e(x,y)$ est la fonction de base valide sur l'élément e associée noeud i. Sur chaque élément, il existe une fonction de base par noeud. Ces fonctions valent 1 sur le noeud associé, valent 0 sur les autres noeuds et s'exprime linéairement

sur le reste de l'élément . Pour l'élément e, la fonction de base associée au sommet i s'écrit ainsi :

$$N_i^e(x,y) = a_i x + b_i y + c_i. \tag{5.10}$$

Pour calculer les coefficients a_i, b_i et c_i, il faut résoudre le système suivant :

$$\begin{aligned} a_i x_i + b_i y_i + c_i &= 1, \\ a_i x_j + b_i y_j + c_i &= 0, \\ a_i x_k + b_i y_k + c_i &= 0. \end{aligned} \tag{5.11}$$

Une fois les 3 fonctions de base associées à l'élément e calculées, le gradient du potentiel peut maintenant s'exprimer de cette façon :

$$\vec{\nabla}\phi^e(x,y) = \frac{\partial}{\partial x}\phi^e(x,y)\vec{i} + \frac{\partial}{\partial y}\phi^e(x,y)\vec{j}. \tag{5.12}$$

Le développement algébrique de l'équation 5.8 est lourd. Il a été effectué au moyen du logiciel de calcul symbolique *Maple*™. Le détail du calcul est présenté à l'Annexe B. Sur la feuille de calcul *Maple*™, il est intéressant de constater que le gradient du potentiel est constant sur l'élément. Cette observation s'avère juste puisque le calcul du gradient dérive le potentiel interpolé à l'ordre un et la dérivation d'une fonction linéaire ramène bien à une constante.

Le calcul du gradient du potentiel sur l'ensemble des cellules conduit au champ de vitesse de l'écoulement. La figure 5.8 montre le champ de vitesse associé au potentiel présenté antérieurement à la figure 5.7.

Figure 5.8 Champ de vitesse calculé aux éléments (les lignes de contour du potentiel sont présentées par les traits en couleur)

Ce champ de vitesse s'apparente à un champ magnétique émanant d'un dipôle électrique puisque l'équation d'un potentiel magnétique est aussi une équation de Poisson 5.8.

Tel qu'expliqué précédemment, le champ de vitesse suggère plusieurs directions à suivre qui permettent d'atteindre la destination. La sous-section suivante montre comment, à partir du champ, tracer les lignes de courant qui servent ensuite de chemin candidat.

5.3.4 Calcul des lignes de courant

En mécanique des fluides, une *ligne de courant* est une courbe tracée dans l'espace décrivant le mouvement d'un fluide. En chacun des points où s'écoule un fluide, l'écoulement possède une orientation qui est décrite par le champ de vitesse. Les lignes de courants sont formées par l'ensemble des courbes qui sont toujours tangentes aux vecteurs vitesse. Dans le cas d'un écoulement potentiel stabilisé, une ligne de courant peut se tracer simplement en partant dans une direction près de la source et en suivant la direction des vitesses rencontrées jusqu'au voisinage du puits. Pour tracer plusieurs lignes de courant, il suffit de répéter plusieurs fois l'exercice, mais en partant dans une

direction différente à la source. L'implémentation proposée utilise un cercle de faible rayon (10 à 20 cm) centré à la source de fluide sur lesquels les points de départ des lignes de courant sont échantillonnés. En pratique, une vingtaine de lignes de courant donnent de bons résultats. La figure 5.9 montre des lignes de courant associées à l'écoulement présenté à la figure 5.8.

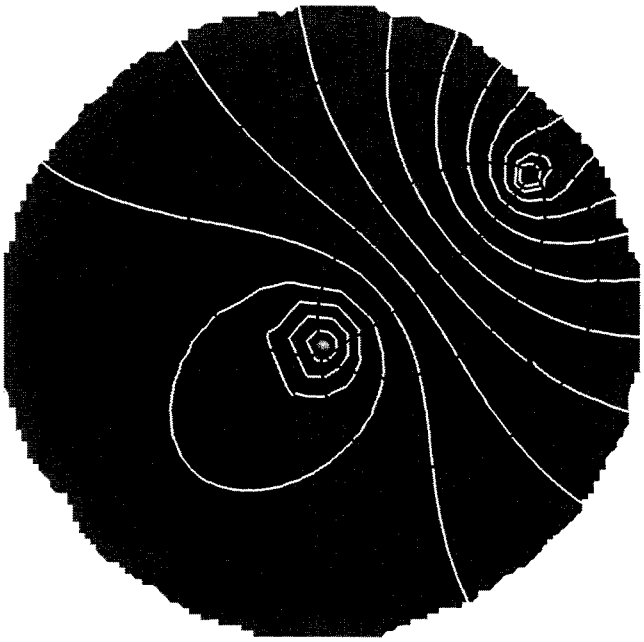


Figure 5.9 Lignes de courant décrivant un écoulement potentiel (les lignes de courant sont noires et les courbes contours de potentiel sont blanches)

Si l'écoulement est correctement calculé, les lignes de courant coupent perpendiculairement les courbes contours de potentiel, ce qui semble être en grande partie le cas à la figure 5.9.

5.3.5 Construction du *roadmap*

La section 2.2.2 a présenté la notion en robotique du *roadmap* qui est globalement un ensemble de chemins candidats entre une position initiale et une destination. Le *roadmap* peut se former en projetant dans l'espace 3D du terrain les lignes de courant 2D. Pour ce faire, il suffit de calculer l'altitude sur le maillage du terrain de chaque point formant une ligne de courant en appliquant la méthode d'interpolation présenté à la figure 4.22 et au moyen de l'équation 4.6.

Une fois toutes les lignes de courant projetées sur le terrain 3D, il faut vérifier la faisabilité de chacun des chemins candidats afin de s'assurer que le robot est en mesure de

suivre sécuritairement le chemin qui sera choisi. Pour ce faire, chaque point formant les chemins est testé en fonction de deux critères définissant un obstacle présenté à la section 5.1.1, soit : la rugosité maximale et la sortie du robot du maillage. Comme les triangles trop pentus ont été retirés du modèle de terrain, les obstacles de pente sont maintenant ignorés. Dans le cas d'un maillage dont les frontières ont été grossies tel que présenté à la section 4.6, l'obstacle de méconnaissance peut aussi être omis dans l'analyse. La caractérisation d'un point d'un chemin peut se faire au moyen d'une empreinte discrète géométrique du robot. La figure 5.1 montre la géométrie du robot (cercle). Cette forme est discrétisée sur grille régulière telle qu'exposée à la figure 5.10.

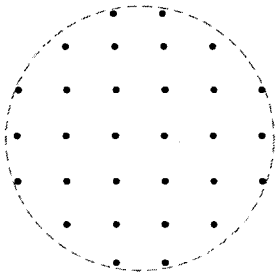
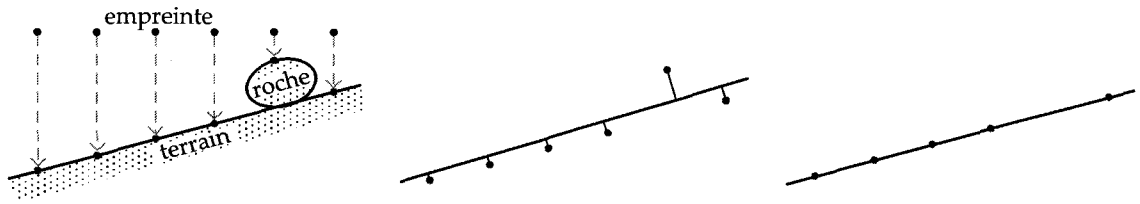


Figure 5.10 Empreinte discrète du robot

Pour chaque point des chemins candidats, l'empreinte est projetée au sol de la même façon que pour les lignes de courant. En d'autres termes, l'altitude de chacun des points de l'empreinte est calculée par interpolation linéaire. Par la suite, la pente moyenne locale est obtenue au moyen de la méthode *intelligent surface mean plane* discutée par Jean de Lafontaine dans l'article *Autonomous planetary landing using a lidar sensor : The navigation function* [de Lafontaine et Gueye, 2004]. La méthode calcule d'abord un plan moyen au sens des moindres carrées qui passe par les points 3D de l'empreinte. Ensuite, elle calcule la distance *point-surface* entre le plan et chacun des points de l'empreinte de même que l'écart type. La relation permettant le calcul de la distance point-plan est présentée à l'équation 5.21. Les points éloignés de la moyenne forment possiblement des roches et faussent le calcul de pente. Ainsi, les points à l'extérieur de quelques écarts-types (2 ou 3) sont rejetés et le calcul de plan moyen est refait sur les points restants. La figure 5.11 schématise cette démarche au moyen d'un exemple 1D.



(a) Projection de l'empreinte au sol

(b) Calcul du plan moyen (ligne bleue) et des distances *point-plan* (lignes noires)

(c) Rejet des points éloignés de la moyenne et recalcul du plan moyen (ligne bleue)

Figure 5.11 Schématisation de la méthode *Intelligent surface mean plane*

L'exemple de la figure 5.11 montre bien comment la méthode calcule le plan moyen de la surface sous les obstacles. Le calcul de plan par moindre carré est relativement simple et peut se faire au moyen de l'algèbre linéaire. L'équation 5.13 rappelle l'équation générale d'un plan antérieurement présentée à la section 4.5 :

$$z = ax + by + c. \tag{5.13}$$

Si l'empreinte possède n points, il est possible de bâtir le système d'équations matricielles suivant à partir de l'équation 5.13 :

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} = a \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + b \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} + c, \tag{5.14}$$

qui peut se réécrire ainsi :

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \tag{5.15}$$

ou de façon plus compacte :

$$Z = \Psi\gamma. \tag{5.16}$$

Le système d'équations 5.16 possède n équations pour 3 inconnues (a, b et c), il est donc surdéterminé. Le livre *Numerical methods for least squares problems* [Björck, 1996] propose une solution au système au sens des moindres carrés :

$$\gamma = \text{inv}\left(\Psi^{t}\Psi\right)\Psi^{t}Z. \tag{5.17}$$

Les coefficients a, b et c minimisant l'erreur entre l'empreinte 3D et le plan moyen étant connus, l'orientation de cette surface peut ensuite être calculée. Il faut d'abord déterminer le vecteur normal au plan moyen. Une façon simple est de changer l'expression du plan vers une formulation *normale*. Ainsi, l'équation normale d'un plan s'exprime par l'équation 5.18 :

$$n_x x + n_y y + n_z z + d = 0, \tag{5.18}$$

avec n_x, n_y et n_z les composantes du vecteur normal au plan et d est la distance entre le plan et l'origine. En comparant l'équation 5.18 avec l'équation 5.13, il est possible d'exprimer les composantes normales en fonction des coefficients a, b et c :

$$\begin{aligned} n_x &= a, \\ n_y &= b, \\ n_z &= -1. \end{aligned} \tag{5.19}$$

Il est possible d'utiliser le vecteur normal au plan pour calculer l'angle moyen de l'empreinte par rapport à la l'horizontale. Pour aider la compréhension, la figure 5.12 illustre dans l'espace un vecteur normal référencé par rapport à un repère aligné avec la gravité.

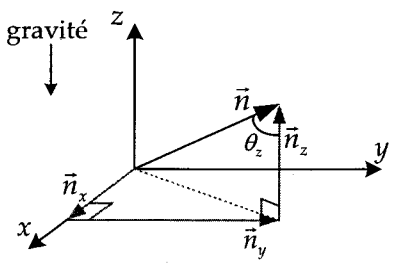


Figure 5.12 Représentation dans l'espace d'un vecteur normal

L'angle recherché entre la normale et la verticale est l'angle θ_z. Par trigonométrie, il est possible de déduire l'angle θ_z ainsi :

$$\theta_z = \arctan\left(\frac{\sqrt{n_x^2 + n_y^2}}{n_z}\right). \tag{5.20}$$

Le critère à vérifier quant à la faisabilité du chemin analysé est la rugosité au niveau de l'empreinte. La rugosité s'obtient du calcul de la distance 3D entre chaque point de l'empreinte au sol et le plan moyen. Le livre *Geometric Tools for Computer Graphics* [Schneider et Eberly, 2003] propose l'équation suivante pour le calcul de la distance point-plan :

$$r = \hat{n} \cdot P + d, \tag{5.21}$$

où $\hat{n}$ est le vecteur normal unitaire du plan, P est un vecteur position d'un point 3D de l'empreinte et d est encore la distance entre le plan et l'origine. Pour chacun des points formant l'empreinte au sol, la distance r est calculée. Si la valeur maximale de r rencontrée dépasse la limite opérationnelle du robot en rugosité, le chemin candidat est rejeté.

Il est possible que le robot possède des contraintes nonholonomes[3] limitant ainsi sa capacité à changer son orientation et sa translation. Afin de s'assurer que le chemin respecte cette contrainte, l'algorithme vérifie que l'angle entre 2 segments de droite consécutifs liant les points formant le chemin ne dépasse pas un angle maximal défini par les capacités du robot. La figure 5.13 schématise l'angle à vérifier α .

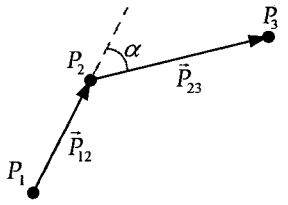


Figure 5.13 Représentation de l'angle entre deux segments d'un chemin

La figure 5.13 montre un chemin composé uniquement de trois points (P_1, P_2 et P_3) qui forment deux segments de droite pouvant être décrits par les vecteurs $\vec{P}_{12}$ et $\vec{P}_{23}$.

[3]La section 2.1 présente la définition de contrainte nonholonome

L'angle α peut se calculé au moyen de la définition géométrique du produit scalaire entre 2 vecteurs 2D :

$$\alpha = \arccos\left(\hat{P}_{12} \cdot \hat{P}_{23}\right), \tag{5.22}$$

avec $\hat{P}_{12}$ et $\hat{P}_{23}$ les vecteurs unitaires qui engendrent les segments de droite. Les angles α associés à tous les points des chemins sont vérifiés. Un chemin est rejeté aussitôt qu'un angle α dépasse la limite de courbure fixée. Une fois que tous les points formant tous les chemins candidats sont analysés au moyen de l'empreinte au sol et de l'angle α, il ne reste à la fin que les chemins faisables et sécuritaires formant le *roadmap*. L'étape subséquente est le choix du chemin optimal dans le *roadmap*.

5.3.6 Sélection du meilleur chemin

Le tableau 3.1 a présenté les critères permettant l'analyse d'optimalité des chemins au moyen de critères et de pondérations. Les chemins contenus dans le *roadmap* satisfont tous les critères suivants, car ils ont été vérifiés lors de la construction du *roadmap* :

- Atteinte de la destination ;
- Sécurité ;
- Respect des limites nonholonomes.

La sélection du chemin peut se faire à l'aide des critères du tableau 3.1 restant :

- Longueur ;
- Énergie.

Pour chacun des chemins du *roadmap*, l'algorithme de sélection associe deux valeurs, une fonction de la longueur du tracé et une fonction de l'énergie consommée à suivre le chemin. La longueur l d'un chemin s'obtient simplement par la somme des longueurs 3D des segments dont la projection discrétise la ligne de courant, ou mathématiquement :

$$l = \sum_{i=2}^{n} \|P_i - P_{i-1}\|, \tag{5.23}$$

où P_i est la coordonné 3D de la position du point i exprimé dans le repère du maillage. La variable n est le nombre de points formant le chemin. Pour l'analyse de l'énergie,

seule l'énergie potentielle gravitationnelle associée à une élévation d'altitude est considérée. Ainsi pour chaque chemin, le gain en altitude l_{z+} est calculé au moyen de l'équation suivante :

$$l_{z+} = \sum_{i=2}^{n} \delta_i, \tag{5.24}$$

avec δ_i défini ainsi :

$$\delta_i = \begin{cases} (P_{i_z} - P_{i-1_z}), & \text{si } (P_{i_z} - P_{i-1_z}) > 0, \text{sinon} \\ 0. \end{cases} \tag{5.25}$$

Le terme P_{i_z} est la composante z du vecteur position du point i appartenant à un chemin. Une fois tous les chemins du *roadmap* analysés par les critères exprimés aux équations 5.23 et 5.24, un coût peut leur être associé. Il faut d'abord normaliser à l'unité les valeurs de longueur et d'énergie. Pour ce faire, chacune des longueurs l et l_{z+} calculées est divisée par les valeurs maximales rencontrées dans l'ensemble du *roadmap*. De cette façon, les longueurs des chemins et les valeurs d'énergie sont toutes comprises entre 0 et 1 et sont notées $\hat{l}$ et $\hat{l}_{z+}$. Puis, un coût peut être associé à chacun des tracés au moyen de la relation 5.26 :

$$C = \beta_l \hat{l} + \beta_{l_{z+}} \hat{l}_{z+}, \tag{5.26}$$

où β_l et $\beta_{l_{z+}}$ sont les pondérations associées aux critères. Les valeurs numériques sont présentées au tableau 3.1. Finalement, le chemin qui ressort de l'algorithme de planification de chemin est celui possédant le coût le plus faible.

5.3.7 Exemples d'utilisation

Un premier exemple d'utilisation reprend le scénario antérieurement présenté à la figure 5.4. Dans l'exemple présenté à la figure 5.14, le robot est positionné au centre du maillage et souhaite atteindre la destination illustrée par un point rouge.

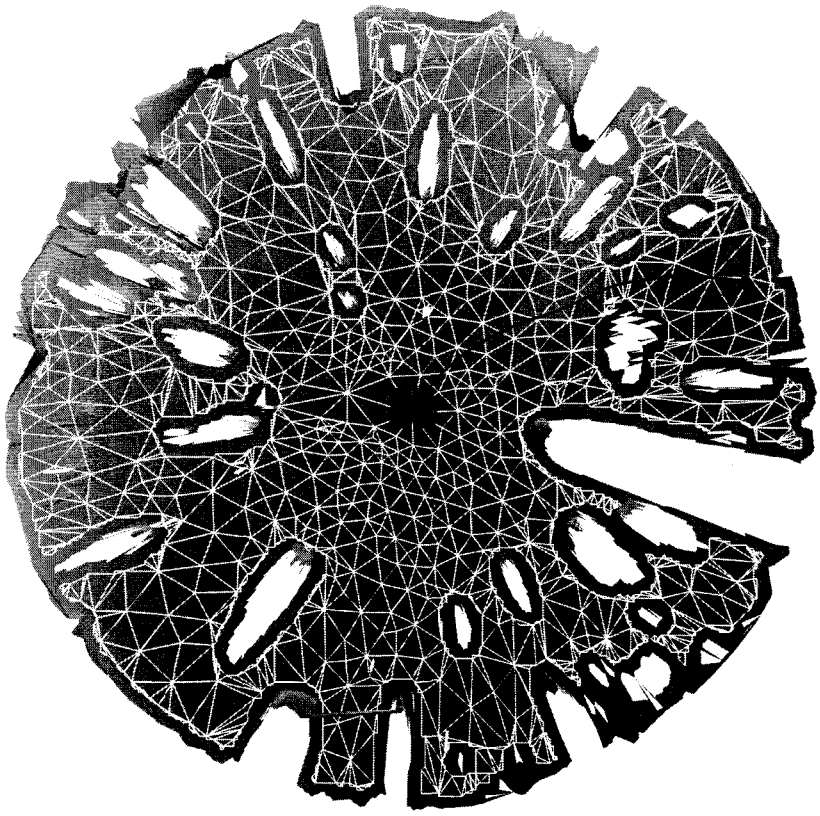


Figure 5.14 Planification de chemin au moyen de l'approche par mécanique des fluides sur un terrain plat avec obstacles (le maillage brut du terrain est en couleur et possède un rayon de 6 m, les arêtes du maillage ayant servi au calcul de chemin sont en blanc, le chemin obtenu est en noir et la destination est au point rouge)

L'exemple de la figure 5.14 montre le chemin calculé par l'algorithme de planification de chemin. Celui-ci atteint bien la cible, il semble sécuritaire et tout à fait acceptable du point de vue de l'optimisation. En effet, il passe par un chemin relativement direct sans détour inutile. De plus, il est lisse et il demeure loin des frontières. La figure 5.15 présente un second exemple d'application de la méthode sur un maillage réel obtenu aussi de données expérimentales.

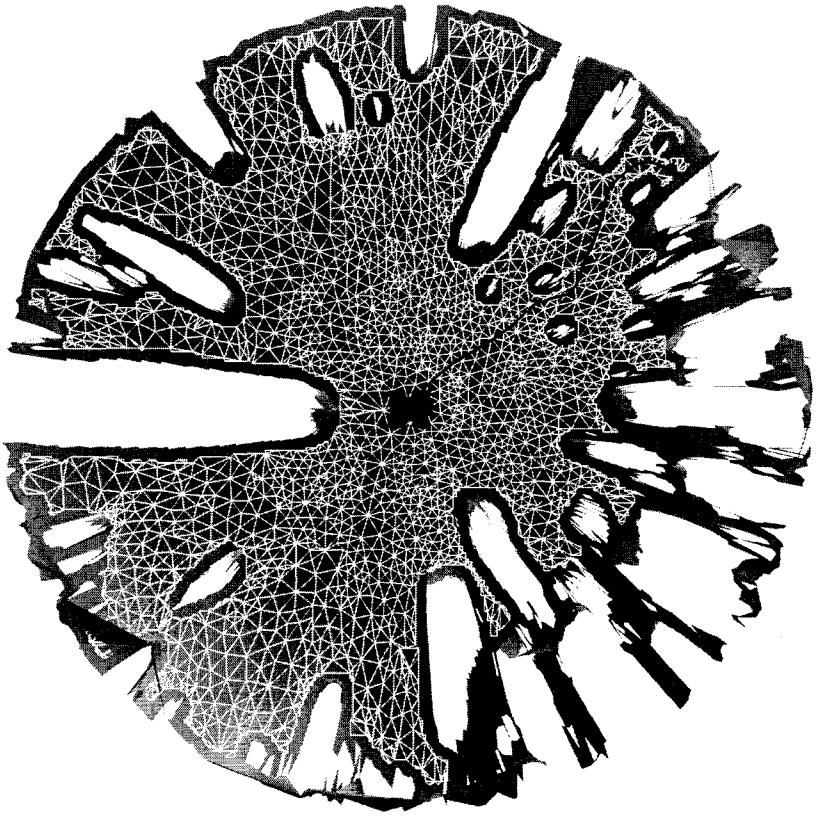


Figure 5.15 Second exemple de planification de chemin au moyen de l'approche par mécanique des fluides sur un terrain plat avec obstacles

Ce dernier exemple est une situation plus complexe pour l'algorithme, car pour atteindre la destination, le robot doit contourner davantage d'obstacles. Une fois encore, l'algorithme retourne une solution qui semble efficace.

5.3.8 Synthèse de la méthode

La méthode de planification inspirée de la mécanique des fluides qui alimente ce projet de recherche se résume en quelques étapes entre la réception des données brutes *LIDAR* jusqu'à l'obtention du chemin à suivre par le robot. L'étape préliminaire est la génération du modèle de terrain basé sur un maillage triangulaire. La synthèse de cette étape a été présentée à la section 4.8. Une fois le maillage construit, l'algorithme de planification de chemin peut s'exécuter. Dans un premier temps, il résout le potentiel de vitesse (équation de Poisson 5.6) sur le maillage du terrain en associant une source positive de fluide à la position du robot et un puits à la destination à atteindre. Ensuite, il calcule le champ de vitesse sur chaque triangle au moyen du gradient négatif du potentiel. L'étape suivante est le calcul des lignes de courant qui suivent la direction du

champ de vitesse. Les lignes de courant 2D sont par la suite projetées sur le terrain 3D. Les tracés infaisables ou dangereux sont exclus et un ensemble de chemins candidats est formé (*roadmap*). Finalement, le meilleur chemin satisfaisant les critères d'optimisation du tableau 3.1 est sélectionné. Le schéma présenté à la figure 5.16 résume ces étapes.

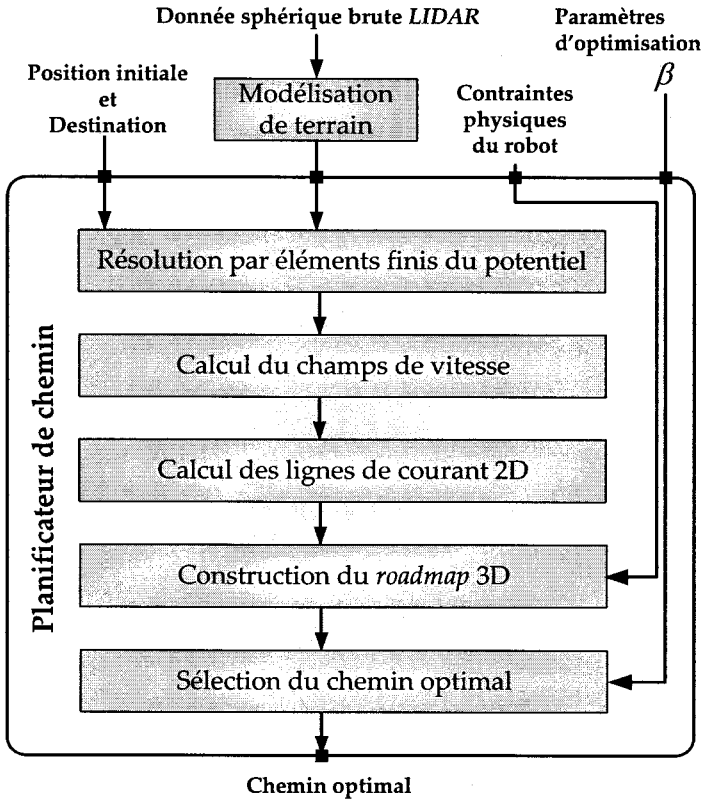


Figure 5.16 Diagramme de flux du calcul de planification de chemin au moyen de l'approche de mécanique de fluide

5.3.9 Conclusion sur l'approche par mécanique des fluides

La section 5.3 a présenté en détail l'algorithme de planification de chemin qui anime ce projet de recherche. Cette approche élégante basée sur les lois physiques régissant les écoulements potentiels de fluide permet d'obtenir des chemins qualifiés d'optimaux. Ceux-ci sont lisses et continus tout comme les lignes de courant d'un fluide s'écoulant de façon laminaire entre une source et un puits. Les changements d'orientation étant limités, le robot suit un chemin qui engendre un minimum d'erreur odométrique. Un autre avantage de la méthode est l'indépendance de la solution au conditionnement du maillage. En effet, contrairement aux approches de recherche de graphe, les chemins

générés par la méthode de fluide n'imposent pas de contrainte artificielle forçant le robot à suivre les cellules du modèle de terrain. Si la ligne droite est libre et s'avère le meilleur choix, l'algorithme est capable de retourner cette solution.

Par contre, la méthode possède quelques lacunes non négligeables. Le temps de calcul de la fonction potentielle par la méthode des éléments finis est de complexité $O(n^3)$. C'est donc dire que pour un maillage possédant beaucoup de noeuds, le calcul peut être long. Aussi, la méthode telle que présentée à la section 5.3, ne permet que l'usage d'un maillage 2.5D décrivant une fonction au sens mathématique du terme. En d'autres mots, pour chaque coordonnée (x, y) du modèle de terrain ne peut exister qu'un seul z. C'est pourquoi l'étape de traitement du maillage présenté à la section 4.4.1 extrait le domaine navigable du terrain et rejette les plafonds. Une autre faiblesse de l'approche vient de l'imposition des lois physiques qui force le robot à se déplacer comme un fluide. Il est possible que le véritable chemin optimal ne ressemble en rien à une ligne de courant.

La section suivante présente le calcul du corridor de sécurité faisant partie des objectifs présentés à la section 3.1. Par la suite, une méthode hybride de planification utilisant une recherche de graphe, un corridor de sécurité et un calcul fluidique est présentée.

5.4 Corridor de sécurité

Tel qu'expliqué à la section 3.1, les objectifs du projet comprennent la construction d'un corridor de sécurité autour du chemin. La figure 3.1 illustre ce concept. Celui-ci permet d'associer au chemin une certaine incertitude quant à la précision du positionnement du robot requise. Sans ce corridor, le chemin possède une épaisseur nulle et aucun robot n'est capable de suivre une ligne si mince en pratique. La construction d'un tel corridor est relativement simple. La largeur du corridor étant fixée, les cellules dont le centre est à l'intérieur de l'enveloppe convexe formée par le chemin élargi de la demi-largeur sont conservées. Il suffit ensuite de retirer les cellules jugées non sécuritaires en fonction des critères antérieurement définis à la section 5.3.5. Toutefois, l'évaluation systématique de tous les triangles peut s'avérer longue. Il est possible d'accélérer l'analyse des triangles au moyen d'une recherche structurée.

5.4.1 Construction du *kdTree*

Il existe de nombreuses façons de rechercher des cellules en fonction de la position de leur centre. La méthode triviale est une recherche de *force brute* qui analyse systématiquement toutes les cellules. Cette approche de complexité $O(n)$ peut s'avérer longue dans le cas d'un maillage contenant beaucoup de triangles. Pour accélérer la recherche, il est possible d'utiliser une structure de données ayant préalablement ordonné la coordonnée du centre de chaque cellule en fonction de leur position. Une des méthodes basées sur ce principe est la structure *quadTree*. Dans celle-ci, l'ensemble d'un maillage 2D est sous-divisé en *boîtes* de dimensions égales et chaque centre de cellule est classé dans la boîte le supportant. Ensuite, lorsque la cellule dont le centre est le plus proche d'un point est recherchée, la recherche s'effectue uniquement dans la boîte contenant le point. Si cette boîte contient bien le centre du triangle recherché, la recherche est terminée. Autrement, la recherche est poursuivie dans les boîtes voisines. Dans le meilleur des cas, la complexité d'une telle recherche est de $O(n/p)$, où p est le nombre de boîtes. Le livre *Computer graphics : principles and practice* [Foley, 1995] présente la méthode de construction et d'utilisation des *quadTrees* et des *ocTrees* dans le cas d'un maillage volumique 3D.

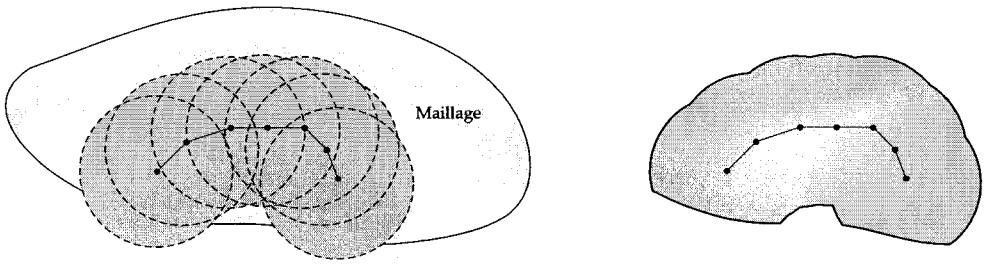
Plus récemment la structure *kdTree* est apparue. Celle-ci organise les cellules dans un ensemble de boîtes dont les dimensions sont variables. L'algorithme de recherche *kdTree* a été écrit avec une telle généralité qu'il s'utilise en toute dimension. L'article *Multidimensional divide-and-conquer* [Bentley, 1980] examine en détail la performance de cette méthode qui possède une complexité de recherche de $O(\log(n))$.

La première étape conduisant à l'obtention du corridor est la construction du *kdTree*. Pour ce faire, de nombreux codes libres de droit sont disponibles et plusieurs versions sont écrites pour *Matlab*™. Une fois la structure construite, l'étape subséquente est la recherche des cellules à proximité du chemin.

5.4.2 Sélection des cellules à proximité du chemin

Cette étape de calcul prend en entrée un maillage du terrain et un chemin et retourne uniquement les triangles à proximité du chemin sans se soucier de leur faisabilité pour le robot. Tel qu'expliqué précédemment, un chemin est formé d'une liste de points 3D appartenant au domaine maillé. Pour rechercher les cellules près du chemin, une stratégie simple consiste à se placer successivement sur chacun des points du chemin et de rechercher et de conserver les triangles situés dans un rayon d'une demi largeur

du corridor. La recherche de triangles *kdTree* est bien adaptée à ce type de recherche (en anglais *range search*). La figure 5.17 montre un schéma de cette approche.



(a) Recherche de cellules dans un rayon

(b) Conservation des cellules à l'intérieur des cercles

Figure 5.17 Schématisation de la démarche de recherche des cellules à proximité d'un chemin (ligne rouge)

Plus l'échantillonnage du chemin est fin (points rapprochés), plus le corridor est bien découpé. La figure 5.17 montre un schéma de la méthode, en pratique l'algorithme doit vérifier l'échantillonnage du chemin par rapport à la largeur du corridor et rééchantillonner au besoin.

5.4.3 Rejet des cellules non sécuritaires

L'étape finalisant la construction du corridor de sécurité est le rejet des cellules non sécuritaires à proximité du chemin. Pour ce faire, le centre de chaque triangle est analysé au moyen des critères présentés à la section 5.3.5. En chaque point d'évaluation, l'empreinte discrète du robot est projetée sur le corridor et la rugosité ainsi que la pente locale peuvent être calculées. Si au moins une des contraintes opérationnelles du robot est dépassée, la cellule est éliminée. Si les frontières du maillage initial n'ont pas été préalablement agrandies, il est nécessaire de vérifier qu'aucun des points de l'empreinte ne sort du corridor. Dans le cas contraire, la cellule fautive doit être rejetée.

5.4.4 Exemple d'utilisation de l'extracteur de corridor

La figure 5.18 présente un exemple d'utilisation de l'algorithme d'extraction d'un corridor de sécurité. Le maillage placé en entrée est un maillage généré à partir de données réelles mesuré sur le banc d'essai de l'ASC. Le chemin utilisé vient de la méthode de recherche de graphe.

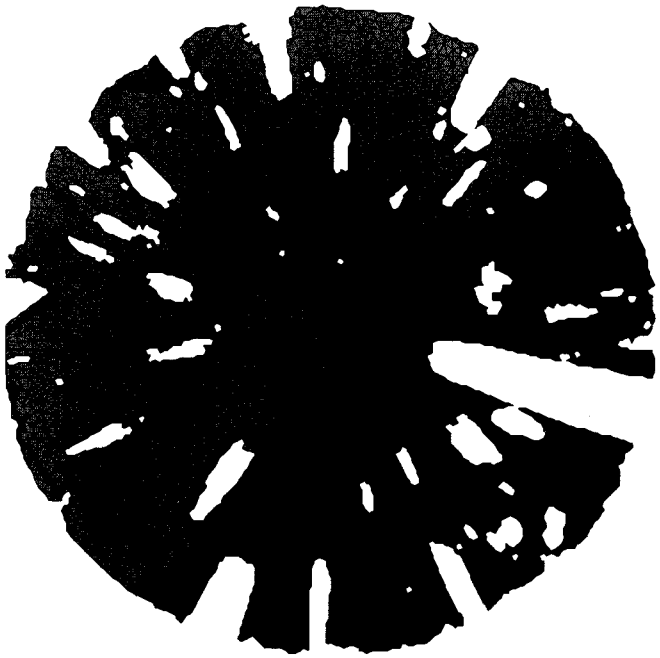


Figure 5.18 Exemple d'un corridor (surface verte) généré à partir d'un chemin obtenu de la méthode de recherche de graphe (ligne noire)

L'exemple de la figure 5.18 montre un corridor où les cellules à proximité des roches et des frontières ont été retirées.

5.4.5 Synthèse de la méthode

L'extracteur de corridor de sécurité prend en entrée un maillage de terrain et conserve les cellules à proximité d'un chemin. Afin d'accélérer la recherche de triangles, l'algorithme a préalablement construit une structure *kdTree*. Une fois le corridor découpé, les cellules non sécuritaires sont rejetées au moyen de l'approche expliquée à la section 5.3.5. Le résultat est finalement un corridor sous forme d'un maillage triangulaire où chaque cellule est sécuritaire pour le robot. Pour que cette sécurité soit valable, il faut que la dimension des cellules soit faible par rapport au rayon de l'empreinte du robot, car uniquement le centre des cellules est vérifié. La figure 5.19 présente un diagramme synthèse de la méthode.

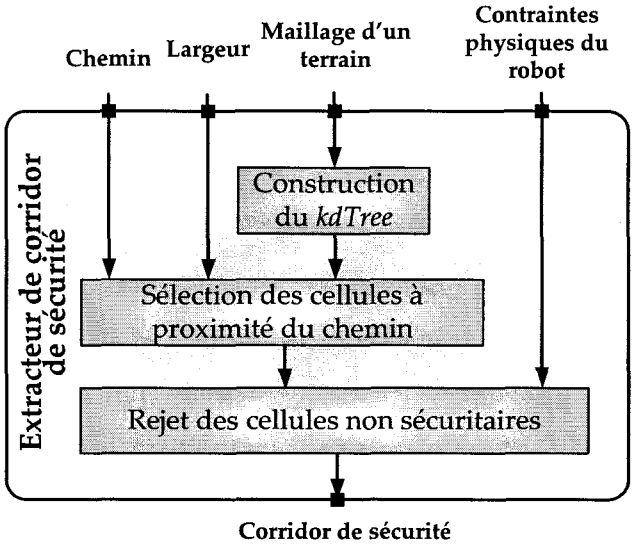


Figure 5.19 Schéma du calcul menant à l'extraction du corridor de sécurité

5.5 Méthode hybride

Les sections 5.2 et 5.3 ont présenté en détail les deux approches de planification de chemin animant ce projet de recherche. Il en est ressorti que la méthode basée sur une recherche de graphe A^* possède des avantages certains. Sa faible complexité de calcul permet d'obtenir une solution en peu de temps. De plus, l'usage d'une fonction coût apporte une flexibilité utile (p. ex., favoriser les chemins passant par les zones ensoleillées). Toutefois, les chemins résultants sont fort sensibles à la forme et au conditionnement du modèle de terrain. Aussi, le détail du tracé est irrégulier et *zigzague* trop pour être envoyé directement à un contrôleur de position. Le lissage de la courbe n'est pas une solution simple à ce problème, car rien de garantit que le nouveau chemin lissé est sécuritaire. La méthode inspirée des écoulements de fluide, quant à elle, génère des chemins lisses, continus et dont l'orientation change minimalement. Par contre, la méthode offre peu de flexibilité et sa complexité de calcul est élevée. De plus, l'approche présentée fait abstraction de la topographie du terrain pour le calcul des lignes de courant. Ces dernières sont projettées sur le terrains 3D pour ensuite être analysées. Rien ne garantit que cette méthode va toujours fonctionner. Par exemple, les lignes de courant calculées sur un maillage contenant une densité importante d'obstacles pourraient être toutes infaisables par le rover.

La méthode inspirée par la mécanique des fluides est fort différente de l'approche de référence $A^\star$, mais pas nécessairement meilleure. L'algorithme $A^\star$ est largement utilisé en pratique, car il possède des avantages. Ce qu'il manque à l'approche de graphe est une étape de *postcalcul* qui lisse le chemin résultant, mais d'une façon sécuritaire. C'est ici que la méthode d'écoulement fluidique peut s'avérer utile. Pour être utilisées comme fonction de lissage, il faut que les lignes de courant produites se concentrent autour du chemin obtenu de $A^\star$ et ainsi le chemin en *dents-de-scie* peut être remplacé par une ligne de courant. Afin de s'assurer que l'écoulement suit bien le chemin de $A^\star$, le calcul de fluide s'effectue sur le corridor de sécurité qui a été découpé autour du chemin. Le corridor possédant beaucoup moins de cellules que le modèle de terrain complet, la résolution par éléments finis est rapide tout comme le reste des calculs de mécanique des fluides. Puis, les lignes de courant étant contraintes à rester sur le corridor de sécurité, les chemins obtenus en sortie sont sécuritaires pour le robot.

La complexité de calcul de l'approche hybride s'obtient par l'analyse de la complexité de la recherche de graphe ($O(n^2)$, où n est le nombre de noeuds du modèle de terrain), de la construction du corridor (recherche *kdTree* $O(\log(n))$) et du calcul de potentiel sur le corridor ($O(m^3)$, où m est le nombre de noeuds du corridor). La complexité de la recherche *kdTree* étant faible par rapport à celle de la recherche de graphe, la complexité de l'approche hybride devient donc : $\max\left(O(m^3), O(n^2)\right)$.

Il s'agit donc ici d'utiliser la méthode $A^\star$ afin d'obtenir une estimation initiale de la solution et l'approche fluidique comme méthode complémentaire permettant de raffiner et d'améliorer la solution initiale. En combinant les deux méthodes, il émerge une approche hybride qui permet de concilier les avantages des deux sans toutefois les inconvénients. Effectivement, la flexibilité de la recherche de graphe est conservée tout comme la rapidité de calcul. Le résultat en sortie amène aussi tous les avantages pour un robot à suivre une ligne de courant. La figure suivante présente un diagramme synthèse de la méthode.

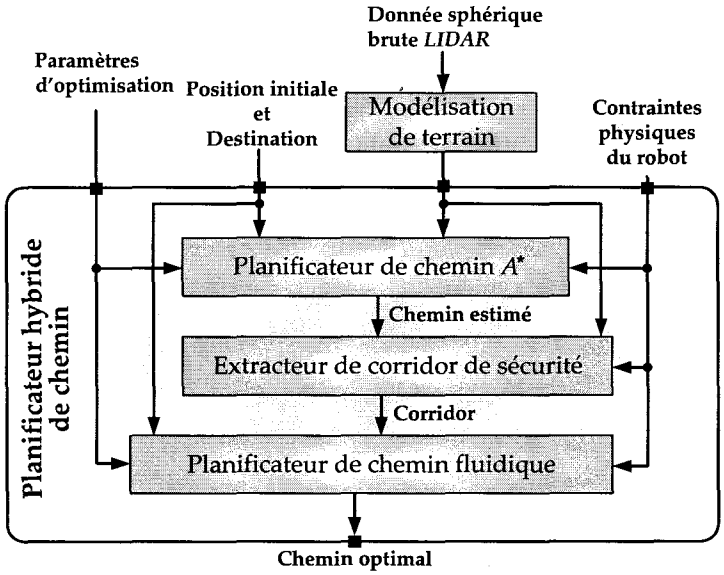


Figure 5.20 Diagramme synthèse de la méthode de planification de chemin hybride

5.5.1 Exemples d'utilisation

Les figures 5.21, 5.22, 5.23 et 5.24 présentent des exemples d'application de l'approche hybride sur des maillages obtenus du banc d'essai de l'ASC.

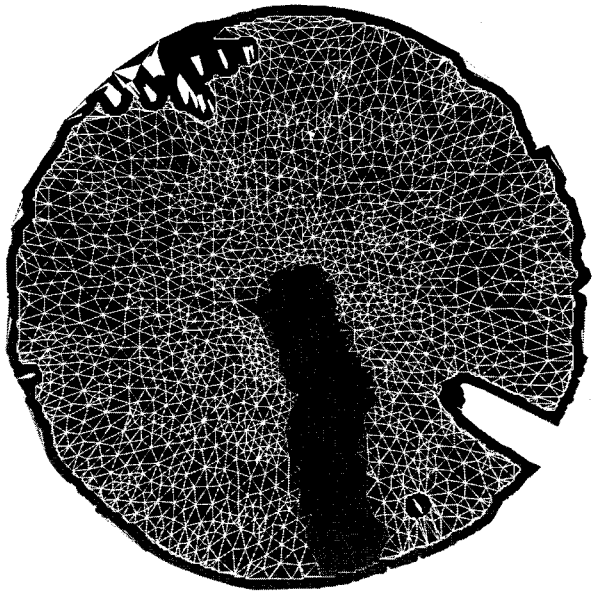


Figure 5.21 Exemple d'une planification de chemin au moyen de la méthode hybride sur un terrain plat et non accidenté (le maillage ayant servi au calcul de chemin $A^\star$ est en blanc, le corridor de sécurité est en vert, le chemin obtenu d'$A^\star$ est en noir et le chemin lissé par l'approche fluidique est en bleu)

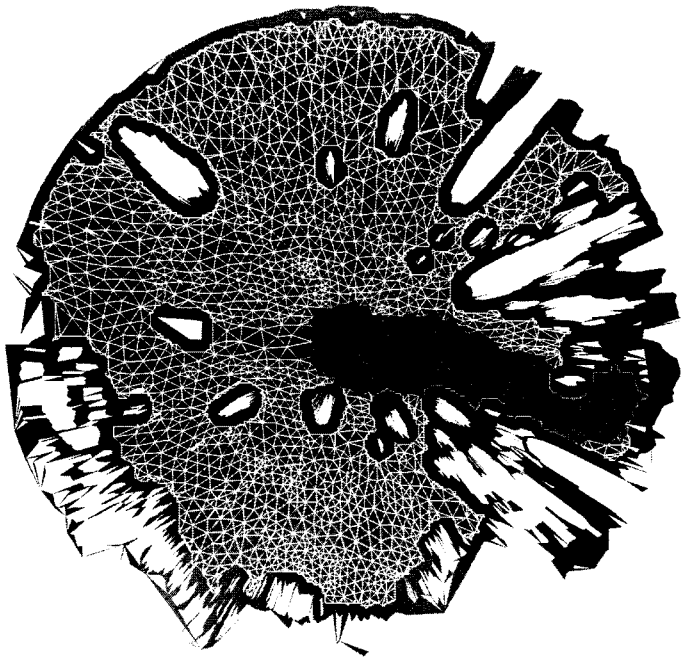


Figure 5.22 Exemple d'une planification de chemin au moyen de la méthode hybride sur un terrain plat possédant beaucoup d'obstacles

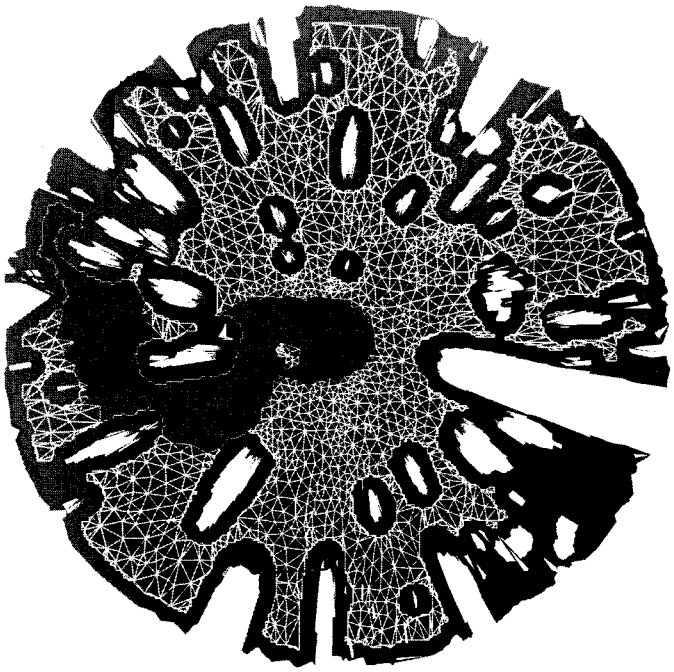


Figure 5.23 Exemple d'une seconde planification de chemin au moyen de la méthode hybride sur un terrain plat possédant beaucoup d'obstacles

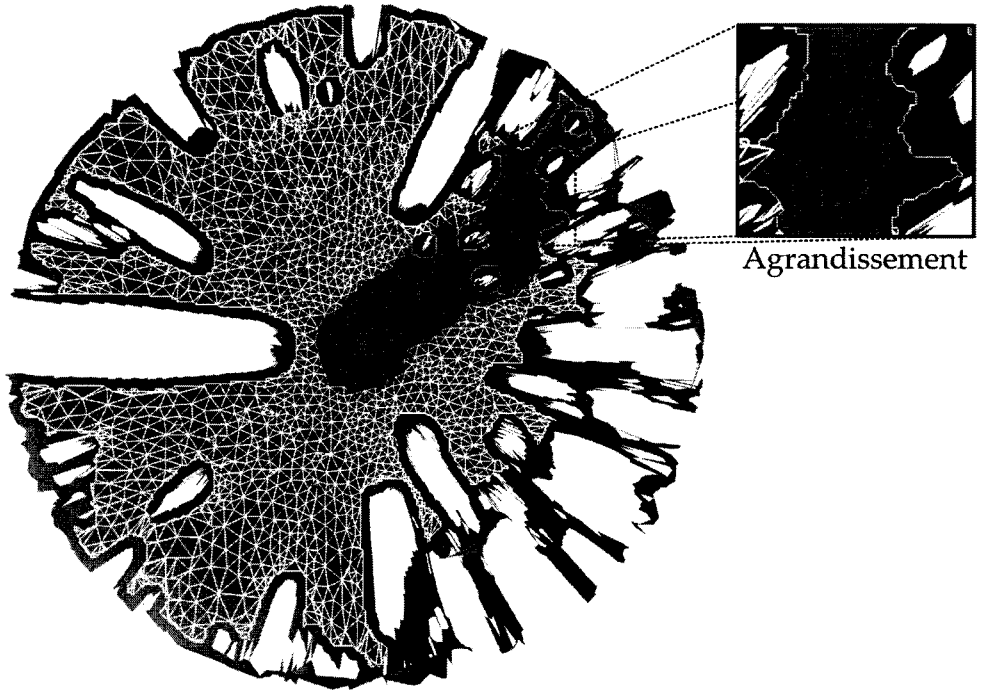


Figure 5.24 Exemple d'une troisième planification de chemin au moyen de la méthode hybride sur un terrain plat possédant beaucoup d'obstacles

Les quatre exemples précédents montrent bien comment deux méthodes différentes peuvent être unies pour former une meilleure méthode. En effet, les chemins obtenus de l'approche hybride (ligne bleue) semblent tout à fait acceptables pour un robot mobile, car ils sont lisses, continus, sécuritaires et optimaux. Aussi, lorsque le chemin obtenu d'$A^\star$ fait fausse route (p. ex., *zigzag* en terrain plat), la méthode fluidique corrige la situation. Par exemple, l'agrandissement à la figure 5.24 montre un chemin $A^\star$ qui *zigzague* inutilement et où la ligne de courant résultante n'a pas été influencée par ce mauvais tracé.

Aussi, l'approche hybride fournit de meilleurs chemins que $A^\star$, elle est plus rapide que la méthode purement fluidique et, de surcroît, fournit aussi un corridor de sécurité. Ce dernier peut s'avèrer utile pour d'autres applications (p. ex., déterminer la précision du positionnement requise en fonction de la largeur du corridor).

CHAPITRE 6

VALIDATION EXPÉRIMENTALE

Ce chapitre a pour but de confirmer expérimentalement l'atteinte des objectifs de projet présentés au chapitre 3.

6.1 Validation *hors-ligne*

Afin de valider l'ensemble des critères de performance à atteindre présentés à la section 3.2, les algorithmes sont mis à l'épreuve sur un grand nombre de situations réelles de planification de chemins. Pour ce faire, cette expérimentation utilise une banque de données contenant 284 nuages de points *LIDAR* mesurés sur le banc d'essai de l'ASC lors des campagnes de tests des années 2007, 2008 et 2009.

6.1.1 Protocole et implantation

Les opérations suivantes sont effectuées séquentiellement sur l'ensemble de la banque de données :

1. charger un nuage de points *LIDAR* de la banque de données ;
2. générer le maillage triangulaire du terrain au moyen de la méthode présentée au chapitre 4 ;
3. générer une destination aléatoire à proximité de la limite du maillage ;
4. confirmer la faisabilité de la destination et sélectionner une autre destination au besoin ;
5. planifier un chemin au moyen des 3 approches présentées au chapitre 5 ;
6. sauvegarder les résultats ;
7. recommencer à 1. jusqu'à ce que tous les nuages de points soient utilisés.

Cette expérience sauvegarde les résultats suivants :

- nombre de points dans le nuage de points *LIDAR* ;
- temps de calcul associé à la génération du maillage et à la planification de chemin ;
- longueur de chaque chemin ;
- changement de courbure moyen de chaque chemin ;
- changement positif d'altitude des chemins (énergie potentielle).

De plus, l'expérience s'assure que le chemin atteint bien la cible et fait le suivi des échecs. Une image illustrant les résultats est aussi sauvegardée pour chacun des cas afin de permettre une inspection visuelle des résultats. L'expérience s'effectue sur un ordinateur portable *Latitude* de *Dell*™ possédant un processeur *Intel Core*™ 2, de 2.0 GB de mémoire *RAM* et d'un système d'exploitation *Linux*. Les paramètres numériques utilisés pour les calculs sont présentés au tableau 6.3.

6.1.2 Résultats

Le tableau 6.1 présente les résultats obtenus *hors-ligne*.

Tableau 6.1 Résultats de l'expérience *hors-ligne* (les écarts-type sont entre parenthèses)

Modélisation de terrain			
Nb. moyen de points *LIDAR*	96k (25k)		
Temps de calcul moyen (s)	17.7 (2.9)		
Échec (%)	0.0		
Planification de chemin			
	Graphe	Fluide	Hybride
Temps de calcul moyen (s)	9.1 (1.9)	14.1 (2.3)	10.3 (2.4)
Échec (%)	3.1	2.8	4.4
Longueur moyenne des chemins (m)	10.0 (1.1)	7.1 (0.8)	6.9 (0.9)
Changement de courbure moyen des chemins (°)	56.4 (5.1)	4.1 (0.7)	4.8 (0.7)
Changement positif d'altitude des chemins (m)	0.33 (0.1)	0.51 (0.1)	0.30 (0.1)

Le tableau 6.1 montre que la recherche de graphe s'avère la plus rapide et l'approche fluidique la plus lente. Le temps total de calcul de la méthode hybride s'apparente à celle de la recherche de graphe. C'est donc dire que la complexité de calcul de ces deux méthodes doit être similaire. Lors de l'exécution des 284 expériences, il est survenu quelques échecs de l'ordre de 3 à 4 % des cas. L'analyse visuelle des résultats sauvegardés en image a révélé que dans la majorité des cas, l'échec provient du fait qu'aucun

chemin sécuritaire n'existe entre la position initiale et la destination. Par exemple, il n'existe aucun chemin possible entre deux secteurs déconnectés d'un maillage. La méthode hybride, présentant un nombre plus élevé d'échecs, semble moins robuste que les deux autres approches. Cela s'explique par le fait que l'approche hybride effectue davantage d'opérations augmentant ainsi le risque d'échec et c'est ce qui s'est produit expérimentalement. Toutefois, le nombre de cas réellement en échec demeure faible permettant quand même de considérer les trois approches comme robustes. L'analyse des chemins montre que les chemins suivant une ligne de courant sont plus courts et plus lisses. Cela s'explique par le fait que la recherche de graphe génère habituellement des chemins en *zigzag*. Le chemin obtenu par l'approche purement fluidique minimise moins l'énergie consommée par le robot, car le calcul des lignes de courant ne suppose a priori aucune connaissance de la topographie du terrain.

6.2 Évaluation des méthodes

La présente section expose une comparaison entre les trois méthodes de planification de chemin traitées, soit : la recherche de graphe, l'approche par mécanique des fluides et la méthode hybride. Cette comparaison est effectuée en s'appuyant sur les critères de performance à atteindre présentés à la section 3.2. Dans un premier temps, les méthodes sont évaluées sur l'atteinte des critères liés aux chemins résultants (critères du tableau 3.1). Ensuite, c'est l'approche algorithmique qui est évaluée au moyen des critères antérieurement présentés permettant de qualifier un algorithme d'«*efficace*» (critères du tableau 3.2). Le tableau 6.2 présente la comparaison qualitative des approches de planification de chemin.

Tableau 6.2 Synthèse de l'atteinte des objectifs de projets des trois méthodes à l'étude (les critères non atteints sont en gras et la définition des critères a été présentée à la section 3.2)

	Méthode de planification de chemin		
Critères sur le résultat	Graphe	Fluide	Hybride
Atteinte de la destination	oui	oui	oui
Longueur minimisée	**non**	oui	oui
Énergie minimisée	oui	moyen	oui
Sécurité	oui	oui	oui
Rugosité minimisée du tracé	**non**	oui	oui
Respect des contraintes holonomes	**non**	oui	oui
Critères sur l'algorithme			
Complexité minimisée	oui	**non**	oui
Robustesse	oui	oui	oui
Contrainte artificielle minimisée	**non**	oui	oui
Complétude	oui	oui	oui
Capable d'utiliser un maillage 3D	oui	**non**	oui
Insensibilité de la solution au maillage	**non**	oui	oui
Flexibilité de la solution	oui	**non**	oui

La décision d'atteinte ou d'échec des critères exposés au tableau 6.2 vient des résultats expérimentaux et aussi de l'analyse théorique des méthodes. Toutes les approches ont expérimentalement atteint les destinations choisies lorsqu'elles étaient atteignables. À cause des fréquents changements de direction, A^* ne permet pas de minimiser la longueur des chemins. Les trois méthodes semblent minimiser l'énergie. Toutefois, l'approche fluidique a montré expérimentalement des chemins plus énergivores que les deux autres. Théoriquement, les trois approches prennent bien en compte les contraintes physiques du robot afin de générer des chemins sécuritaires. Les méthodes fluidique et hybride minimisent bien la rugosité des tracés et par le fait même, respectent mieux les contraintes holonomes du robot que la recherche de graphe. Le calcul par éléments finis sur la totalité du maillage rend l'approche par mécanique des fluides plus coûteux en ressource de calcul que les deux autres. Toutes les méthodes ont montré expérimentalement une bonne robustesse. Contrairement aux approches fluidique et hybride qui utilisent un champ de potentiel, la recherche de graphe impose comme contrainte artificielle d'optimisation de passer par le centre des cellules. Tous les algorithmes ont expérimentalement trouvé des chemins lorsqu'il en existe et retourné un message d'erreur dans le cas contraire, ce qui permet de confirmer l'atteinte de la complétude. Bien que l'approche fluidique telle que développée dans le projet soit incapable d'utiliser un maillage 3D (p. ex., une grotte avec un plafond et un plancher), l'approche hybride l'est, car le corridor de sécurité est découpé de telle sorte qu'il est toujours 2.5D (p. ex.,

plancher uniquement). La méthode de graphe s'avère fort sensible au conditionnement des maillages à cause des contraintes à passer par le centre des cellules. Un maillage déformé amènera donc par la recherche de graphe un chemin déformé. Enfin, la méthode fluidique ne permet pas une grande flexibilité, car elle n'utilise pas une fonction coût aussi flexible que la recherche de graphe. La méthode hybride est considérée flexible, car elle suit le tracé d'$A^\star$ qui lui peut utiliser beaucoup de critères d'optimisation.

Le tableau 6.2 montre bien qu'une recherche de graphe ou un calcul de mécanique des fluides ne permettent pas à eux seuls d'atteindre les objectifs de ce projet de recherche. Chacun d'eux possède des avantages et des faiblesses. L'utilisation d'une approche hybride permet de faire ressortir uniquement les avantages des deux premières méthodes. Toutefois, la méthode hybride, étant plus compliquée, peut s'avérer moins robuste. C'est donc la méthode hybride qui s'avère un algorithme «*efficace*» capable de générer des chemins «*optimaux*» pour un robot mobile explorateur de planète utilisant un système de vision *LIDAR*.

6.3 Implantation sur banc d'essai

Tel qu'expliqué à la section 3.3, ce projet de recherche vise une implantation des algorithmes développés sur un banc d'essai de l'ASC. Cette étape a pour but de confirmer la possibilité d'utilisation des algorithmes dans un contexte expérimental. Le banc d'essai est composé d'un robot mobile se déplaçant sur terrain extérieur simulant un sol martien. Les sections 6.3.1 et 6.3.2 présentent en détail le banc d'essai. Enfin, la section 6.3.3 présente quelques expériences réalisées *en-ligne* à l'ASC.

6.3.1 Plate-forme robotisée

Le robot utilisé est un *Pioneer P2-AT* de marque *ActivMedia*™. Celui-ci utilise des encodeurs optiques au niveau des moteurs afin d'estimer le déplacement dans le plan horizontal. Une centrale inertielle *IMU 300CC-100* de *Crossbow*™ permet de connaître l'accélération du véhicule de même que la vitesse angulaire. Afin d'obtenir une odométrie complète du robot (6 degrés de liberté), un observateur d'états utilise l'information des capteurs et retourne la position et l'orientation 3D filtrées. Le système de vision est un capteur *LIDAR LMS200 de SICK*™ installé sur une unité rotative permettant une mesure 3D de l'environnement du robot.

Sur le plan informatique, le *rover* possède un ordinateur portable *Toughbook* de *Panasonic*™ équipé d'un processeur *Intel Core*™ 2, de 3.0 GB de mémoire *RAM* et d'un système d'exploitation *Linux*. Une machine à états finis implantée dans l'environnement de programmation *Eclipse* prend les décisions de haut niveau. Ces algorithmes sont programmés en langage *JAVA*™.

Les algorithmes proposés par ce projet sont développés dans l'environnement *Matlab*™ et déployés vers *Eclipse* au moyen de l'outil *Builder-Ja* de *Mathworks*™. Cet outil convertit un code *Matlab*™ en une archive *JAVA*™ (*JAR-file*) directement intégrable à *Eclipse*. La figure 6.1 montre la plate-forme robotisée sur le terrain d'apparence martienne.

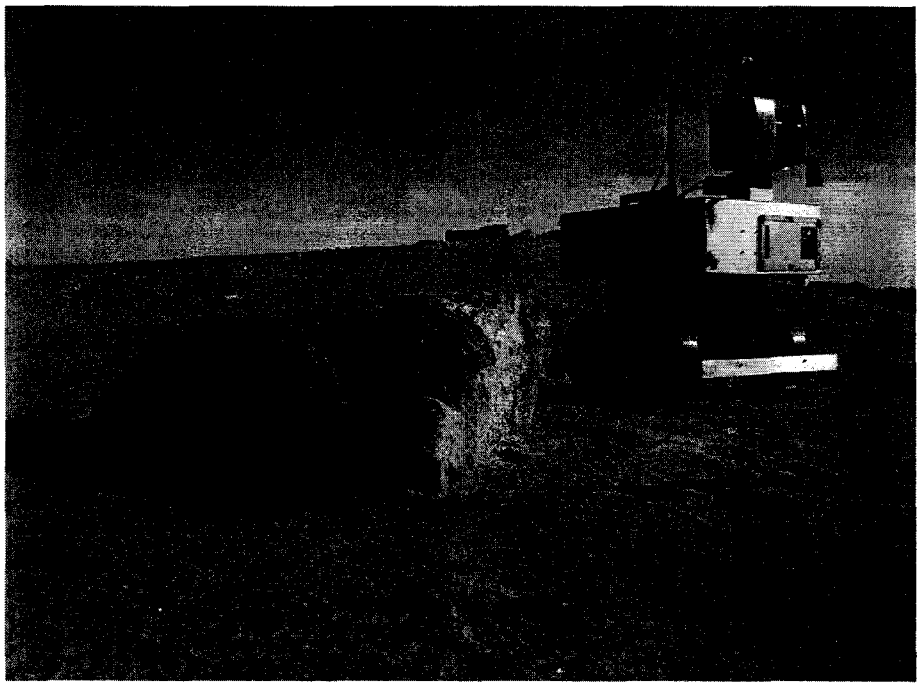


Figure 6.1 P2-AT modifié de l'ASC (photo prise en septembre 2009)

6.3.2 Environnement de test

L'environnement dans lequel le *rover* est utilisé reproduit un sol martien. Ce dernier est situé à l'extérieur du siège social de l'ASC à St-Hubert. Il occupe un espace de 30 m par 60 m et affiche des variations d'altitude de l'ordre de 3 m. La figure 6.2 montre la carte d'élévation du terrain de Mars.

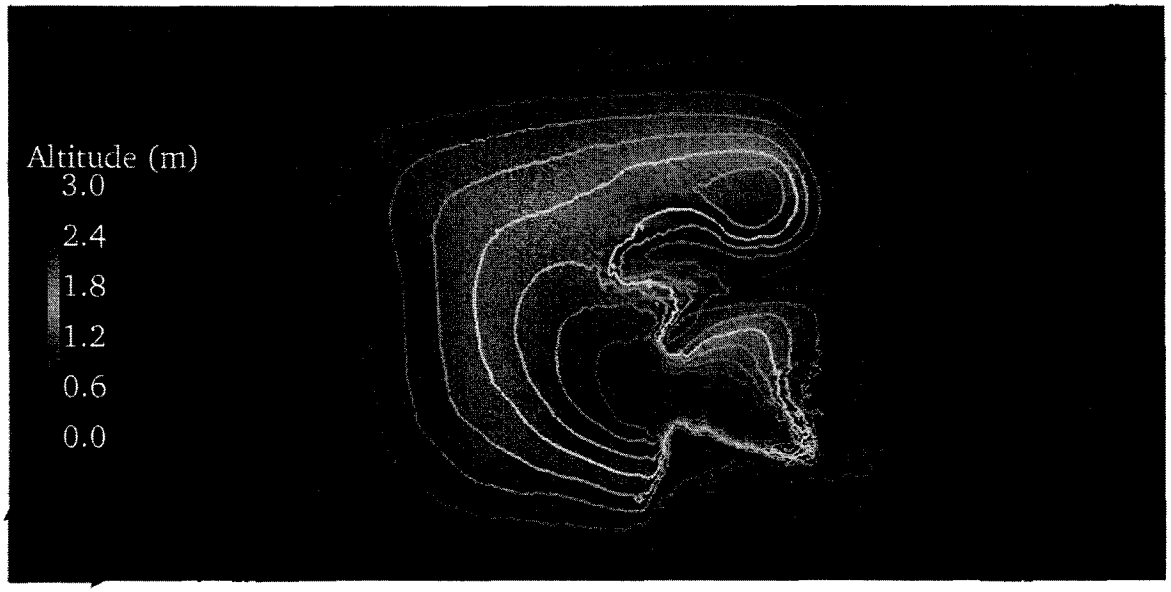
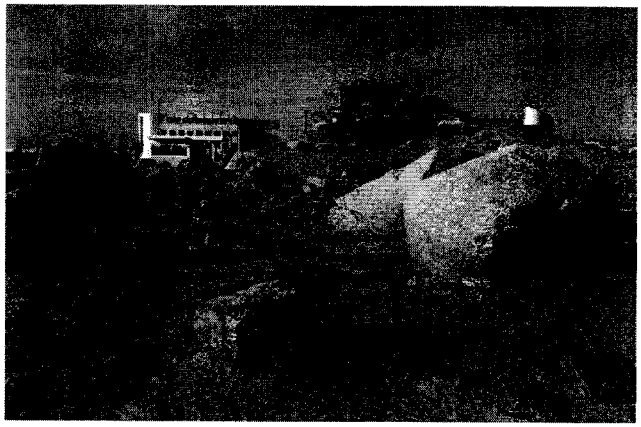


Figure 6.2 Carte d'élévation du terrain de Mars (les lignes représentent les courbes de niveau)

La figure 6.3 montre des photographies du terrain. On y voit des roches de différentes dimensions, un cratère, une montagne, une falaise, une plaine et une colline.

(a) Cratères et colline



(b) Roches et falaise

(c) Plaine et colline

Figure 6.3 Reproduction d'un terrain martien à l'ASC (photos prises en 2005, courtoisie de l'ASC)

6.3.3 Expérience *en-ligne*

Cette section présente trois expériences réalisées sur le banc d'essai décrit précédemment. Le but de cette section est de démontrer la possibilité d'implantation des algorithmes sur un robot mobile.

Présentation de l'expérience

Les paramètres numériques utilisés lors de ces expériences sont présentés au tableau 6.3.

Tableau 6.3 Paramètres utilisés *en-ligne*

Paramètres liés à la modélisation de terrain	
Rayon maximal du maillage	7.0 m
Résolution du rééchantillonnage	7.5 cm
Limite composante Z vecteurs normaux	0.35
Nombre de triangles cibles	8000
Paramètres liés à la modélisation du robot	
Rayon du *rover*	35.0 cm
Résolution de l'empreinte	2.0 cm
Limite opérationnelle de pente	15.0°
Limite opérationnelle de rugosité	6.0 cm

Pour plus d'information sur la nature des paramètres du tableau 6.3, le lecteur intéressé peut consulter le chapitre 4 pour les paramètres liés au modèle de terrain et le chapitre 5 pour les paramètres reliés à la modélisation du robot. Les expériences présentées à la section 6.3.3 suivent toutes le même protocole :

1. placer le robot à une position et une orientation connues sur le terrain ;
2. entrer cette configuration initiale dans l'interface humain/machine (GUI) ;
3. sélectionner une destination à atteindre dans le GUI ;
4. envoyer la commande.

Initialement, la position à atteindre est trop loin pour être directement vue par le *LIDAR*. Le robot doit exécuter de façon autonome une série d'actions le menant à sa destination. Cela comprend d'abord une planification de chemin global entre la position initiale et la destination. Pour ce faire, le robot utilise une recherche de graphe sur un maillage triangulaire à basse résolution (1 m) du terrain. Cette carte, enregistrée préalablement dans le robot, peut provenir de missions antérieures, de données obtenues au moyen d'un satellite ou de données recueillies lors de l'atterrissage du robot. Puis,

le robot mesure son environnement au moyen de son *LIDAR* et génère un maillage du terrain conformément à l'approche présentée au chapitre 4. Vient ensuite la recherche d'une destination sécuritaire à l'intérieur du maillage (destination locale). Pour ce faire, le *rover* débute sa recherche de destination locale au niveau de l'intersection entre le chemin global et la frontière du maillage. Par la suite la planification de chemin local (approche fluidique ou hybride selon l'expérience) est effectuée. Le robot exécute ensuite le chemin en boucle ouverte (sans vérifier en cours de route sa position) et une fois arrivé à sa destination locale, il se localise visuellement. Si la destination globale est atteinte, l'expérience est terminée, sinon le robot reprend les opérations locales. Pour bien comprendre le processus, la figure 6.4 montre le diagramme de flux de l'expérience.

À la figure 6.4, les étapes développées au sein de ce projet sont représentées en gras.

Résultats

Cette sous-section expose trois expériences réalisées sur le banc d'essai de l'ASC. Dans la première, le robot débute à la position (9.0 m, 1.0 m)[1] et cherche à atteindre (2.0 m, 28.0 m). Le planificateur de chemins utilisé pour ce cas est l'approche purement fluidique. La figure 6.5 montre les résultats obtenus sur le terrain.

À la figure 6.5, les maillages locaux sont représentés en beige, les chemins locaux fluidiques sont en bleu et le chemin global est en rouge. Le chemin global étant calculé au moyen d'une recherche de graphe affiche une allure en *dents-de-scie*. La position initiale est illustrée par un point rouge et la destination globale par un point jaune. Les discontinuités entre chaque chemin local viennent des erreurs odométriques qui, une fois corrigées par l'étape de localisation, engendrent un saut de position. Sur ce sol sablonneux, les roues du robot glissent et l'observateur d'états accumule des erreurs de positionnement. Le processus de localisation permet à chaque étape de réduire cette erreur, mais généralement pas de l'éliminer.

Les deux expériences qui suivent utilisent le générateur de chemin basé sur l'approche hybride. La figure 6.6 présente les résultats d'une ascension autonome de la montagne.

L'expérience 2 montre bien l'intérêt d'utiliser le planificateur de chemin global. Celui-ci amène le robot sur un chemin globalement plus long, mais où la pente est minimisée. Sans ce planificateur, le robot aurait possiblement attaqué de front la montagne. Une fois rendu au pied, il ferait face à une pente trop élevée et chercherait une alternative d'échappement (p. ex., rebrousser chemin, zigzaguer pour monter, etc.). Une prise de

[1] Voir figure 6.2 pour connaître le repère utilisé.

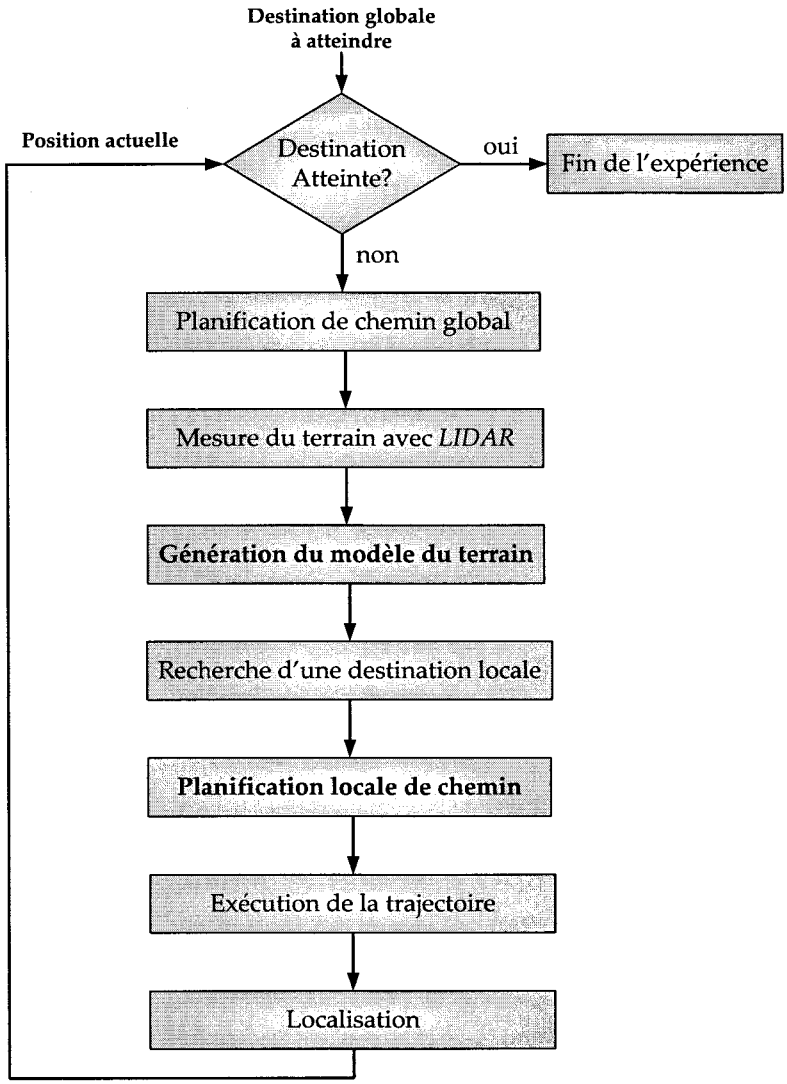


Figure 6.4 Diagramme de flux d'une expérience de navigation autonome

décision à l'échelle globale permet d'éviter les culs-de-sac, ce que ne permet pas une planification exclusivement locale. La figure 6.7 présente les résultats de l'expérience 3.

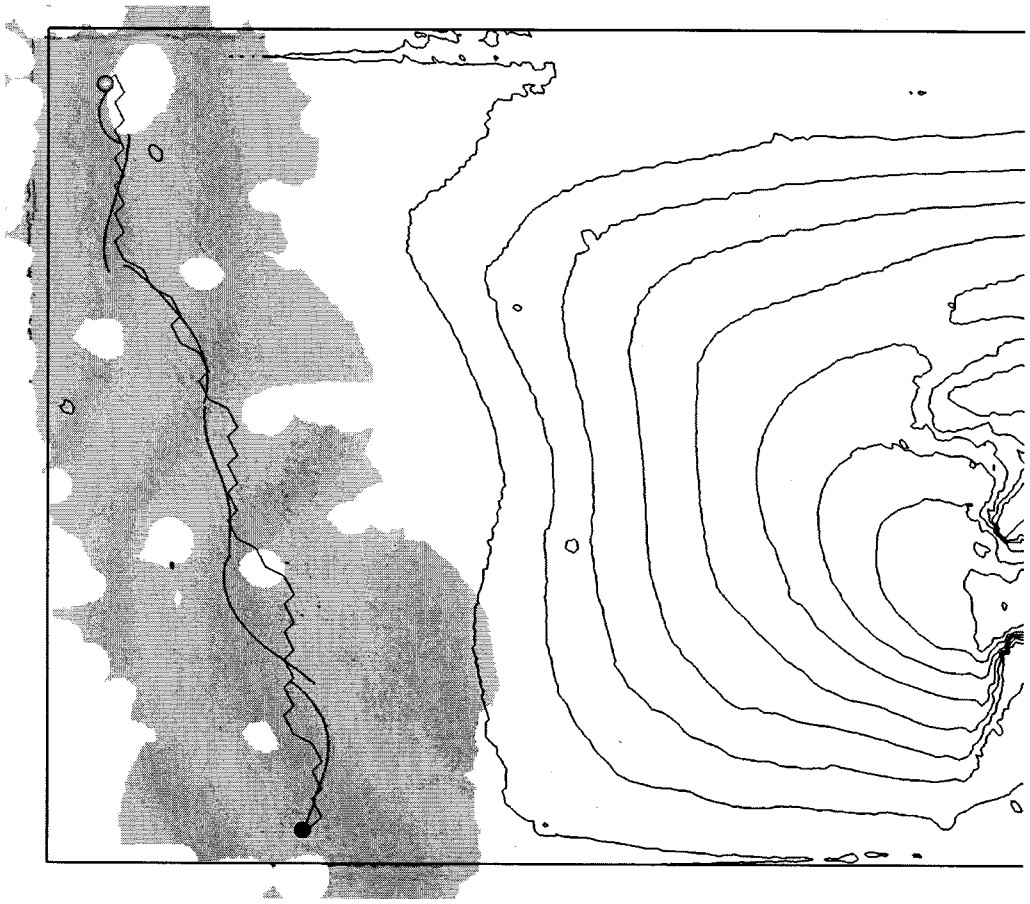


Figure 6.5 Résultats de l'expérience 1 obtenus au moyen de la méthode fluidique

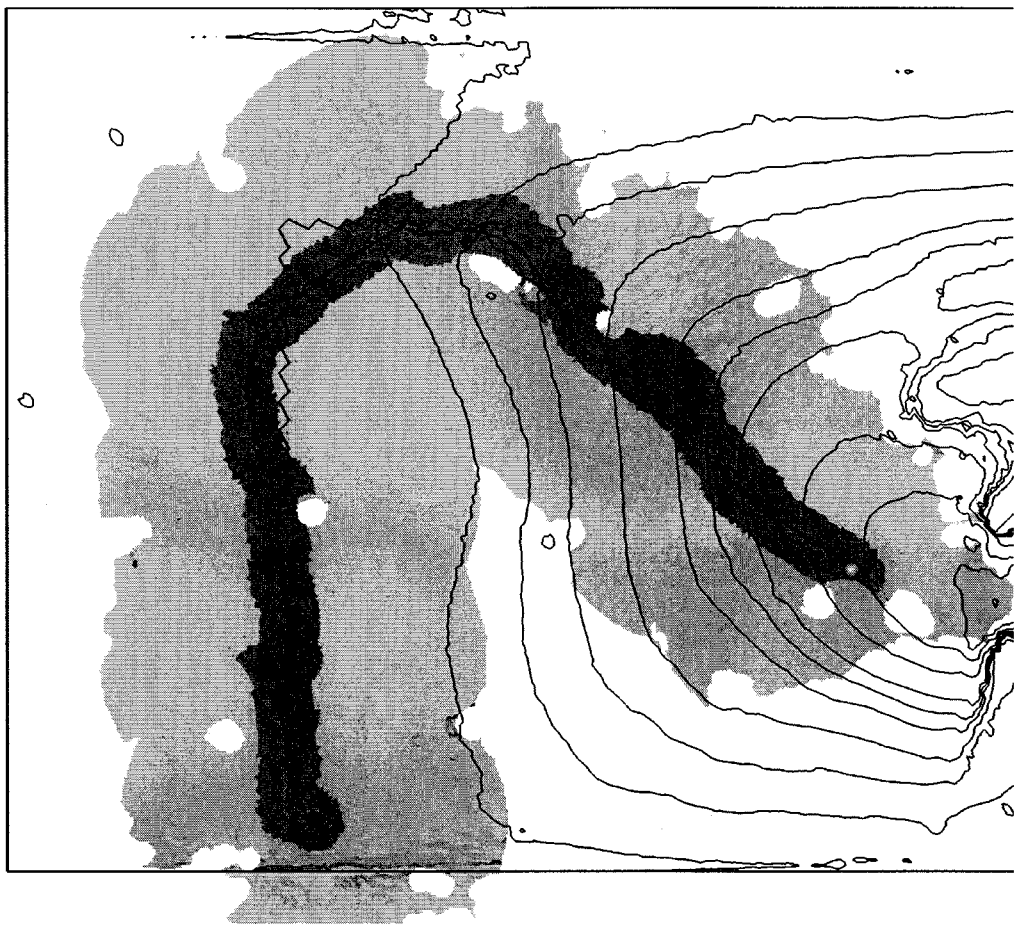


Figure 6.6 Résultats de la montée autonome de la montagne. Les chemins locaux calculés au moyen d'$A^\star$ sont en noir, les corridors de sécurité sont en vert, les maillages locaux sont représentés en beige, les chemins locaux fluidiques sont en bleu et le chemin global est en rouge.

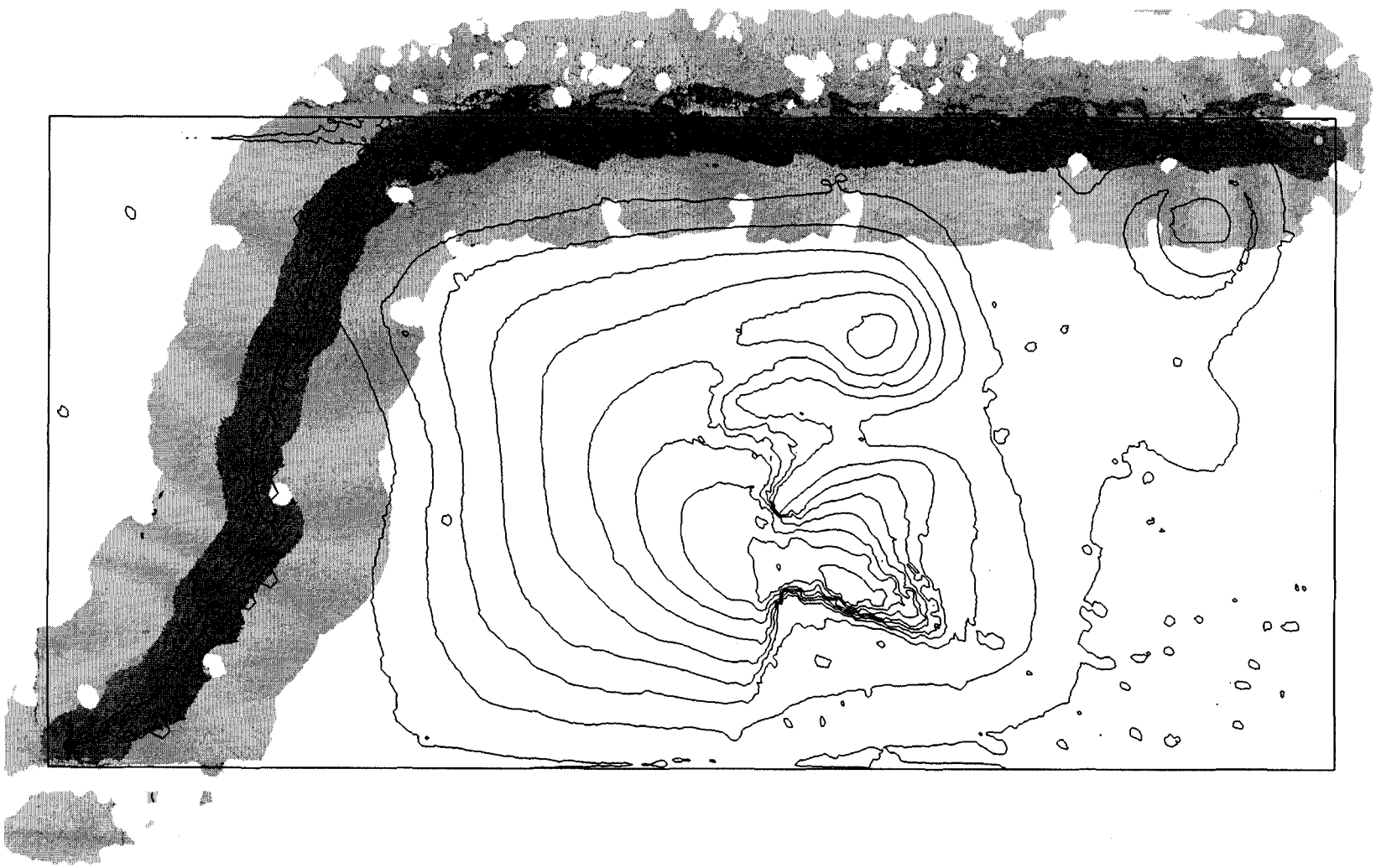


Figure 6.7 Résultats de la traversée autonome du terrain martien. Les chemins locaux calculés au moyen d'$A^\star$ sont en noir, les corridors de sécurité sont en vert, les maillages locaux sont représentés en beige, les chemins locaux fluidiques sont en bleu et le chemin global est en rouge.

La figure 6.7 montre une traversée complète du terrain martien. Le robot à débuté à (1.0 m, 1.0 m) et avait pour destination (59.0 m, 29.0 m). Il a atteint sa destination en évitant la montagne. Le tableau 6.4 présente des résultats quantitatifs obtenus pour l'ensemble des expériences.

Tableau 6.4 Résultats quantitatifs des expériences sur banc d'essai

Exp.	Algorithme	Distance parcourue (m)	Err. de positionnement finale (%)
1	Fluide	32.9	1.8
2	Hybride	50.4	non mesurée
3	Hybride	83.0	1.5

L'objectif d'implantation des algorithmes sur un vrai robot explorateur est donc atteint tout comme l'ensemble des objectifs de projet.

CONCLUSION

En somme, ce projet de recherche a étudié la question de recherche suivante : « Comment un robot peut-il, de façon autonome, générer un chemin sécuritaire et optimal entre une position initiale et une destination connue dans une représentation irrégulière de son environnement » ? L'algorithme conçu prend en entrée un maillage, une position initiale et finale, ainsi que les contraintes mécaniques d'un robot et retourne un chemin ainsi qu'un corridor de sécurité. Le corridor sert à quantifier dans quelle mesure le robot peut dériver de son chemin sans compromettre sa sécurité. L'intérêt de développer un tel algorithme vient du fait que dans un proche avenir, les robots mobiles explorateurs de planète délaisseront possiblement leur système de vision passif pour des systèmes actifs de plus longue portée (*LIDAR*). Ce type de capteur retourne une quantité importante de données qui doivent être généralement compressées. Bien souvent, cette compression amène une représentation irrégulière de l'environnement du robot. À l'heure actuelle, il ne semble pas exister d'algorithme ayant véritablement démontré son efficacité sur un maillage irrégulier. C'est donc dire que pour répondre aux besoins de l'exploration spatiale future, ce projet de recherche est d'une grande pertinence. En plus, les algorithmes de planification de mouvement ont aussi leurs applications sur Terre, en robotique certes, mais aussi dans les domaines des jeux vidéo, de l'animation numérique, de la conception assistée par ordinateur et même en chimie computationnelle.

L'étude de la littérature scientifique a révélé que les approches de génération de chemin basées sur une recherche de graphe sont largement utilisées. Une étude exhaustive de cette méthode présentée à la section 5.2, a démontré que ce type de méthode, bien que peu coûteuse en ressource de calcul, n'est pas adapté à une planification sur un maillage irrégulier. L'imposition de contraintes artificielles, tel que de passer par le centre des cellules, amène une forte sensibilité du résultat au conditionnement du maillage. Parallèlement à cette analyse, le candidat a développé une approche de génération de chemin basée sur un calcul d'écoulement de fluide non visqueux. L'intérêt de cette méthode vient de l'intuitivité, l'insensibilité au maillage et l'optimalité de la solution retournée. Toutefois, la phase de validation expérimentale présentée à la section 6.1 a démontré que cette méthode fluidique ne permet pas d'atteindre tous les critères de performance fixés au chapitre 3.

L'innovation de ce projet réside dans le développement d'une approche hybride. Celle-ci utilise une estimation initiale de chemin obtenue au moyen d'une recherche de graphe (A^*) et du corridor de sécurité pour générer un chemin formant une ligne de courant s'écoulant sur le corridor. C'est au moyen de cette méthode que les objectifs de projet peuvent être atteints. En effet, le calcul d'écoulement qui est lent sur le maillage complet du terrain devient rapide sur le corridor de sécurité. Le manque de flexibilité quant aux solutions possibles d'un calcul de fluidique disparaît lorsque le chemin d'A^* est utilisé comme solution initiale. Enfin, bien que l'approche fluidique telle que développée dans le projet soit incapable d'utiliser directement un maillage 3D (p. ex., une grotte avec un plafond et un plancher), l'approche hybride l'est, car le corridor de sécurité est découpé de telle sorte qu'il est toujours 2.5D (p. ex., plancher uniquement).

Finalement, les extrants de l'algorithme développé (chemin et corridor) pourraient ultérieurement déborder le cadre d'une application de guidage et être aussi utiles au contrôle de position des *rovers*. En effet, les algorithmes de contrôle pourraient utiliser le corridor de sécurité pour moduler la vitesse du robot. Par exemple, si à une position du chemin le corridor est plus étroit, le contrôleur pourrait abaisser l'amplitude de la vitesse, car des obstacles sont possiblement à proximité et inversement lorsque le corridor est pleine largeur.

Un autre aspect intéressant qui pourrait être étudié est le calcul d'écoulement de fluide directement en coordonnées sphériques qui forment le repère du nuage de points brut. L'approche développée dans ce projet transforme plutôt le maillage brut dans un repère cartésien et calcule la fonction de potentiel au moyen de la méthode des éléments finis. En travaillant directement sur le nuage de points sphérique, la lourde étape de maillage est éliminée. De plus, les nuages de points retournés par un *LIDAR* sont toujours 2.5D et généralement à résolution fixe ouvrant ainsi la porte à un calcul d'écoulement plus simple à implanter et moins coûteux qui pourrait être basée sur la méthode des différences finies généralisée.

ANNEXE A
Mise en équation des écoulements potentiels

Les explications qui suivent sont tirées du livre *Fluid mechanics* [White, 2004]. La vitesse en tout point d'un écoulement est la variable d'état à calculer. L'écoulement est supposé stationnaire et incompressible. La viscosité du fluide est de plus négligée. Ces simplifications permettent de garantir l'absence de tourbillon. Dès lors, l'écoulement est dit irrotationnel ce qui mathématiquement s'exprime ainsi :

$$\vec{\nabla} \times \vec{V} = \vec{0}. \tag{A.1}$$

Cet état de vorticité nulle permet de qualifier le champ de vitesse de *conservatif*. Tout champ vectoriel conservatif peut s'écrire comme le gradient d'une fonction scalaire :

$$\vec{V} = -\vec{\nabla}\phi, \tag{A.2}$$

où ϕ est une fonction scalaire qui s'appelle le *potentiel vitesse*. En 2D, les composantes de vitesse peuvent donc s'écrire ainsi :

$$V_x = -\frac{\partial \phi}{\partial x}, \tag{A.3}$$

$$V_y = -\frac{\partial \phi}{\partial y}. \tag{A.4}$$

La relation permettant de relier entres elles les composantes de vitesse est la première loi de la thermodynamique qui stipule que la masse se conserve dans un système. Dans le cas de la mécanique des fluides, cette relation est généralement présenté ainsi :

$$\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot \left(\rho \vec{V}\right) = 0, \tag{A.5}$$

avec ρ la densité du fluide. Comme le fluide est supposé incompressible, ρ est constant. L'équation A.5 se réduit donc à :

$$\vec{\nabla} \cdot \vec{V} = 0, \tag{A.6}$$

ou plus simplement en développant le terme de divergence :

$$\frac{\partial}{\partial x} V_x + \frac{\partial}{\partial y} V_y = 0. \tag{A.7}$$

Ensuite, il est possible de remplacer les équations A.3 et A.4 dans l'équation précédente A.7 pour obtenir la formulation suivante :

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0, \tag{A.8}$$

ou de façon plus compacte :

$$\nabla^2 \phi = 0. \tag{A.9}$$

L'équation A.9 est bien connue en mathématique et porte le nom d'*équation de Laplace*. Cette équation est à la base de la théorie des potentiels et des livres entiers y sont dédiés (voir [J.L., Doob, 1984]). Toute fonction qui vérifie l'équation de Laplace est dite *harmonique*. L'auteur du livre *Elements of partial differential equations* [Drábek et Holubová, 2007] explique qu'en ajoutant une fonction scalaire $q(x, y)$ au côté droit de l'équation A.9, celle-ci permet de prendre en compte les apports externes de fluide tels que les sources et les puits. Cette nouvelle équation elliptique non homogène s'appelle *équation de Poisson* et s'exprime ainsi :

$$-\nabla^2 \phi = q(x, y). \tag{A.10}$$

ANNEXE B

Calcul du gradient du potentiel à l'aide du logiciel *Maple*™

L'équation du potentiel par interpolation de Lagrange (eq. 5.9)

```
> phi_e:= Sum(phi[i]*N[i],i=1..3);
```

$$phi\_e := \sum_{i=1}^{3} \phi_i N_i$$

Définition des fonctions de base (eq. 5.11)

```
> N[1]:= a1*x+b1*y+c1:
> N[2]:= a2*x+b2*y+c2:
> N[3]:= a3*x+b3*y+c3:
```

Assignation des valeurs nodales de potentiel dans le tableau phi[]

```
> phi[1]:= phi1:
> phi[2]:= phi2:
> phi[3]:= phi3:
```

Système d'équations à résoudre pour le noeud 1

```
> eq11:= a1*x1+b1*y1+c1= 1:
> eq21:= a1*x2+b1*y2+c1= 0:
> eq31:= a1*x3+b1*y3+c1= 0:
```

Système d'équations à résoudre pour le noeud 2

```
> eq12:=a2*x1+b2*y1+c2=0:
> eq22:=a2*x2+b2*y2+c2=1:
> eq32:=a2*x3+b2*y3+c2=0:
```

Système d'équations à résoudre pour le noeud 3

```
> eq13:=a3*x1+b3*y1+c3=0:
> eq23:=a3*x2+b3*y2+c3=0:
> eq33:=a3*x3+b3*y3+c3=1:
```

Résolution des systèmes d'équations

```
> sol1:=solve({eq11,eq21,eq31},[a1,b1,c1]):
> assign(sol1);
> sol2:=solve({eq12,eq22,eq32},[a2,b2,c2]):
```

```
> assign(sol2);
> sol3:=solve({eq13,eq23,eq33},[a3,b3,c3]):
> assign(sol3);
```

Evaluation de la somme

```
> phi_e:=value(phi_e):
```

Calcul du gradient de phi_e (eq. 5.8)

```
> Grad_phi_e:=simplify(Gradient(phi_e,[x,y]));
```

$$Grad\_phi\_e := \frac{-\phi 1\, y3 + \phi 1\, y2 - \phi 2\, y1 + \phi 2\, y3 + \phi 3\, y1 - \phi 3\, y2}{x3\, y1 - x3\, y2 - x2\, y1 - x1\, y3 + x1\, y2 + x2\, y3}\, ex -$$
$$\frac{-\phi 1\, x3 + \phi 1\, x2 + \phi 2\, x3 - \phi 2\, x1 + \phi 3\, x1 - \phi 3\, x2}{x3\, y1 - x3\, y2 - x2\, y1 - x1\, y3 + x1\, y2 + x2\, y3}\, ey$$

Les coordonnées des sommets 1 à 3 sont (xi, yi, zi) et les potentiels nodaux associés sont ϕi, avec i = 1 à 3. Dans l'environnement de *Maple*, la base vectorielle engendrant le domaine cartésien 2D est ex et ey.

LISTE DES RÉFÉRENCES

Astola, J. et Kuosmanen, P. (1997). *Fundamentals of Nonlinear Digital Filtering*. CRC Press, 276 p.

Aurenhammer, F. (1991). Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Computing Surveys*, volume 23, numéro 3, p. 345–405.

Béland, S., Dupuis, E., Tremblay, I., Dunlop, J., Reedman, T. et Fulford, P. (2000). Space robotics in Canada – From dexterous operations on ISS to planetary exploration robotics. Dans *51st International Astronautical Congress*.

Belyaev, A. et Ohtake, Y. (2003). A comparison of mesh smoothing methods. Dans *Israel-Korea Bi-National Conference on Geometric Modeling and Computer Graphics*. volume 2.

Bentley, J. (1980). Multidimensional divide-and-conquer. *Communications of the ACM*, volume 23, numéro 4, p. 214–229.

Berliner, H. (1979). The B* Tree Search Algorithm – A Best-First Proof Procedure. *Artificial Intelligence*, volume 12, p. 23–40.

Björck, A. (1996). *Numerical methods for least squares problems*. Society for Industrial and Applied Mathematics (SIAM), 408 p.

Carrier, W. (1992). Soviet rover systems. Dans *Space Programs and Technologies Conference (AIAA)*. p. 1487–1496.

Carsten, J., Rankin, A., Ferguson, D. et Stentz, A. (2007). Global Path Planning on Board the Mars Exploration Rovers. *IEEE Aerospace Conference*, p. 1–11.

Cignoni, P., Montani, C. et Scopigno, R. (1998). A Comparison of Mesh Simplification Algorithms. *Computers and Graphics*, volume 22, numéro 1, p. 37–54.

Connolly, C., Burns, J. et Weiss, R. (1990). Path planning using laplace's equation. Dans *IEEE International Conference on Robotics and Automation (ICRA)*. volume 3. p. 2102–2106.

Daly, M., Choi, E., Michelangeli, D., Carswell, A. et Tremblay, I. (2004). The canadian met contribution to the 2007 phoenix mars mission. Dans *55th International Astronautical Congress*. p. 1–5.

De Berg, M., Cheong, O., Van Kreveld, M. et Overmars, M. (2008). *Computational Geometry : Algorithms and Applications*, 3e édition. Springer, 386 p.

de Lafontaine, J. et Gueye, O. (2004). Autonomous planetary landing using a lidar sensor : The navigation function. *Space Technology*, volume 24, numéro 1, p. 7–18.

Delaunay, B. (1934). Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, volume 7, p. 793–800.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, volume 1, numéro 1, p. 269–271.

Drábek, P. et Holubová, G. (2007). *Elements of partial differential equations*. Walter de Gruyter, 245 p.

Dudek, G. et Jenkin, M. (2000). *Computational Principles of Mobile Robotics*. Cambridge University Press, 280 p.

Dupuis, E., Allard, P., Bakambu, J., Lamarche, T. et W.-H., Zhu (2004). Towards autonomous long-range navigation. Dans *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*.

Foley, J. (1995). *Computer graphics : principles and practice*. Addison-Wesley Professional, 1175 p.

Forgave, J., Man, K. et Hoffman, A. (2006). *Overview of Mars Science Laboratory (MSL) Environmental Program* (Rapport technique). NASA, Jet Propulsion Laboratory.

Garland, M. et Heckbert, P. (1997). Surface simplification using quadric error metrics. Dans *Proceedings of the 24th Annual International Conference on Computer graphics and Interactive Techniques (SIGGRAPH)*. p. 209–216.

Ge, S.S. et Lewis, F.L. (2006). *Autonomous Mobile Robots : Sensing, Control, Decision-making, and Applications*. CRC Press, 709 p.

George, P. et Borouchaki, H. (1998). *Delaunay Triangulation and Meshing : Application to Finite Elements*. Kogan Page, 413 p.

Gold, C. (1978). The practical generation and use of geographic triangular element data structures. *Harvard Papers on Geographic Information Systems*.

Goldberg, S., Maimone, M. et Matthies, L. (2002). Stereo vision and rover navigation software for planetary exploration. *IEEE Aerospace Conference*, volume 5, p. 2025–2036.

Henriksen, L. et Krotkov, E. (1997). Natural terrain hazard detection with a laser rangefinder. *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, p. 968–973.

Howard, A. M. et Tunstel, E. W. (2006). *Intelligence for Space Robotics*. TSI Press, 425 p.

Huang, T., Yang, G. et Tang, G. (1979). A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, volume 27, numéro 1, p. 13–18.

Huebner, K. (2001). *The finite element method for engineers*. Wiley-Interscience, 720 p.

J.L., Doob (1984). *Classical potential theory and its probabilistic counterpart*. Springer, 846 p.

Kalra, N., Ferguson, D. et Stentz, A. (2006). Incremental reconstruction of generalized voronoi diagrams on grids. *Intelligent Autonomous Systems*.

Keymeulen, D. et Decuyper, J. (1994). The fluid dynamics applied to mobile robot motion : the stream field method. Dans *IEEE International Conference on Robotics and Automation (ICRA)*. volume 1. p. 378–385.

Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. Dans *IEEE International Conference on Robotics and Automation (ICRA)*. volume 2. p. 500–505.

Kim, J.-O. et Khosla, P. (1992). Real-time obstacle avoidance using harmonic potential functions. *IEEE Transactions on Robotics and Automation*, volume 8, numéro 3, p. 338–349.

Koren, Y. et Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. Dans *IEEE International Conference on Robotics and Automation (ICRA)*. volume 2. p. 1398–1404.

Krasner, M. S. (2002). *MSL Scenarios and Autonomy Requirements* (Rapport technique). NASA, Jet Propulsion Laboratory.

Kreith, F. et Goswami, D. Y. (1999). *The CRC Robotics Handbook of Mechanical Engineering*, 2e édition. CRC Press, 2688 p.

Kreyszig, E. (1993). *Advanced engineering mathematics*, 7e édition. Wiley New York, 1271 p.

Kring, D. (2006). *Lunar and Planetary Institute*. `http://www.lpi.usra.edu/science/kring/lunar_exploration/briefings/lunar_mobility_review.pdf` (page consultée le 20 février 2008).

Lacomme, P., Prins, C. et Sevaux, M. (2003). *Algorithmes de graphes*, 2e édition. Editions Eyrolles, 411 p.

Lange, C., Allard, P., Dupuis, E. et Gonthier, Y. (2004). Unified facility for the development, integration and testing of autonomous navigation schemes for planetary exploration. Dans *Romansy Conference*.

Latombe, J. C. (1991). *Robot Motion Planning*. Kluwer Academic Publishers, 651 p.

Laubach, S. et Burdick, J. (1999). An autonomous sensor-based path-planner for planetary microrovers. Dans *IEEE International Conference on Robotics and Automation (ICRA)*. volume 1. p. 347–354.

LaValle, S. (2006). *Planning algorithms*. Cambridge University Press, 826 p.

Li, Z. X. et Bui, T. D. (1998). Robot path planning using fluid model. *Journal of Intelligent and Robotic Systems : Theory and Applications*, volume 21, numéro 1, p. 29–50.

Liu, L., Crowe, T. G. et Bakambu, J. N. (2008). Efficient exploration algorithms for rough terrain modeling using triangular mesh maps. Dans *IEEE Conference on Robotics, Automation and Mechatronics*. p. 1206–1211.

Loudon, K. (2000). *Mastering algorithms with C*. O'reilly and associates, 540 p.

Lozano-Perez, T. (1983). Spatial planning : A configuration space approach. *IEEE Transactions on Computers*, volume 32, numéro 2, p. 108–120.

Lozano-Perez, T. et Wesley, M. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the Association for Computing Machinery*, volume 22, numéro 10, p. 560–570.

Lumelsky, V. J. et Stepanov, A. A. (1987). Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, volume 2, numéro 1, p. 403–430.

Maimone, M. W. (2007). Overview of the Mars Exploration Rovers' Autonomous Mobility and Vision Capabilities. *IEEE International Conference on Robotics and Automation (ICRA)*.

Matijevic, J. et Shirley, D. (1997). The mission and operation of the mars pathfinder microrover. *Control Engineering Practice*, volume 5, numéro 6, p. 827–835.

Mishkin, A. H., Morrison, J. C., Nguyen, T. T., Stone, H. W., Cooper, B. K. et Wilcox, B. H. (1998). Experiences with operations and autonomy of the mars pathfinder microrover. Dans *IEEE Aerospace Conference*. volume 2. p. 337–351.

M.S., Gatti (2003). *The Deep Space Network Large Array* (Rapport technique). NASA, 42–157 p.

Muirhead, B. K. (2004). Mars rovers, past and future. Dans *IEEE Aerospace Conference*. volume 1.

Murray, R. M. et Sastry, S. S. (1993). Nonholonomic motion planning : steering using sinusoids. *IEEE Transactions on Automatic Control*, volume 38, numéro 5, p. 700–716.

Nakamura, Y. et Mukherjee, R. (1991). Nonholonomic path planning of space robots via a bidirectional approach. *IEEE Transactions on Robotics and Automation*, volume 7, numéro 4, p. 500–514.

NASA (2008a). *Welcome to the NSSDC*. `http://nssdc.gsfc.nasa.gov` (page consultée le 10 février 2008).

NASA (2008b). *JPL Robotics : Home Page*. `http://www-robotics.jpl.nasa.gov` (page consultée le 16 février 2008).

Nilsson, N. J. (1969). A mobile automaton : An application of artificial intelligence techniques. Dans *International Conference on Artificial Intelligence*. p. 509–520.

O'Dunlaing, C. et Yap, C. K. (1985). A retroaction method for planning the motion of a disc. *IEEE Journal of Algorithms*, volume 6, numéro 1, p. 104–111.

Pitteway, N. L. K. (1973). Computer graphics research in an academic environment. Dans *Datafair Conference*. p. 471–478.

Pocchiola, M. et Vegter, G. (1993). The visibility complex. Dans *The ninth annual symposium on Computational geometry*. ACM New York, NY, USA, p. 328–337.

Rao, S. S. (2005). *The finite element method in engineering*, 4e édition. Butterworth-heinemann, 663 p.

Rekleitis, I., Bedwani, J.-L. et Dupuis, E. (2009). Autonomous planetary exploration using lidar data. Dans *IEEE International Conference on Robotics and Automation (ICRA)*.

Rekleitis, I., Bedwani, J.-L., Dupuis, E. et Allard, P. (2008a). Path planning for planetary exploration. Dans *Canadian Conference on Computer and Robot Vision (CRV)*. p. 61–68.

Rekleitis, I., Bedwani, J. L., Gingras, D. et Dupuis, E. (2008b). Experimental results for over-the-horizon planetary exploration using a lidar sensor. Dans *The Eleventh International Symposium on Experimental Robotics (ISER)*.

Sallaberger, C. et Force, S. P. T. (1997). Canadian space robotic activities. *Acta Astronautica*, volume 41, p. 239–246.

Schneider, P. J. et Eberly, D. H. (2003). *Geometric tools for computer graphics*. Morgan Kaufmann, 1009 p.

Siegwart, R. et Nourbakhsh, I. R. (2004). *Introduction to Autonomous Mobile Robots*, 2e édition. MIT Press, 321 p.

Spenko, M., Lagnemma, K. et Dubowsky, S. (2004). High speed hazard avoidance for mobile robots in rough terrain. Dans *SPIE Conference on Unmanned Ground Vehicle Technology*.

Volpe, R. et Khosla, P. (1990). Manipulator control with superquadric artificial potential functions : theory and experiments. *IEEE Transactions on Systems, Man and Cybernetics*, volume 20, numéro 6, p. 1423–1436.

Waydo, S. et Murray, R. (2003). Vehicle motion planning using stream functions. Dans *IEEE International Conference on Robotics and Automation (ICRA)*. volume 2.

Wein, R., van den Berg, J. et Halperin, D. (2007). The visibility – voronoi complex and its applications. *Computational Geometry : Theory and Applications*, volume 36, numéro 1, p. 66–87.

White, F. (2004). *Fluids mechanics*. WCB/McGraw-Hill,, 866 p.

Zienkiewicz, O. (1983). The generalized finite element method – state of the art and future directions. *Journal of applied mechanics*, volume 50, p. 1210–1217.

Zienkiewicz, O. C. et Taylor, R. L. (2000). *The finite element method – Fluid dynamics*. McGraw-Hill, 320 p.