UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Génie électrique et génie informatique

ATLAS MULTI-COUCHES POUR ROBOT MOBILE

Mémoire de maîtrise ès sciences appliquées
Spécialité: génie électrique

Jean-Luc BEDWANI

Jury: François Michaud
Érick Dupuis
Dominic Létourneau
Michel Lauria

Sherbrooke (Québec) Canada Août 2009

«Ce que l'on conçoit bien s'énonce clairement
et les mots pour le dire arrivent aisément»

– N. Boileau

RÉSUMÉ

En cherchant à connaitre les origines de la vie sur Terre, on se demande également si elle peut se trouver ailleurs. Malgré le fait que Mars nous est voisine et qu'elle ne semble pas en démontrer la présence, on désire toutefois l'étudier et la comparer avec la Terre. Pour y arriver, l'Homme a recours à la technologie : les télescopes, satellites, sondes et robots servent à observer Mars. À chacune des expéditions d'observation de Mars, nous cherchons à en connaitre davantage et améliorons en conséquence la technologie utilisée. C'est pour cela que la prochaine génération de véhicules d'exploration planétaire nécessitera une plus grande autonomie de navigation.

De tels requis impliquent une gestion adéquate de données géoréférencées potentiellement volumineuses et denses, représentées sous la forme de cartes. Ce mémoire présente l'état des recherches faites sur un système de gestion de données utilisable dans un contexte d'exploration planétaire autonome. Pour valider un tel système, il a fallu utiliser, mais également constituer, un répertoire de cartes en trois dimensions ayant été géoréférencées. Afin d'obtenir un répertoire représentatif du contexte d'utilisation, il a également été nécessaire de développer un comportement d'exploration et de navigation sur une plateforme robotisée, capable d'acquérir les cartes requises. À l'aide de ce répertoire, il a été possible de vérifier les capacités, les performances ainsi que l'exactitude des opérations effectuées à l'aide du système de gestion proposé.

Un article intégré au mémoire présente la conception d'un tel système de gestion de données, ainsi qu'une solution permettant de gérer dynamiquement une variété de données, l'incertitude des relations spatiales entre deux cartes, de procurer un mécanisme de planification de chemins au travers des cartes, ainsi que la corrélation des cartes pour les opérations de localisation. Cet article présente également les résultats expérimentaux sur l'utilisation du système de gestion atlas par un véhicule d'exploration autonome.

En plus de démontrer la faisabilité et l'utilité d'un tel gestionnaire de données en navigation autonome, le système pourrait également être utilisé comme plateforme d'analyse afin de comparer les performances de différents algorithmes de recalage de surfaces.

Mots-clef : Atlas, Gestionnaire de données, Cartographie, Recalage, Robot, Navigation autonome.

REMERCIEMENTS

Avant toute chose, je tiens à remercier mes co-directeurs de recherche, M. François Michaud et M. Érick Dupuis, d'avoir accepté la supervision de cette maîtrise. Sans leurs lumières, leur soutien et leurs nombreuses révisions, ce mémoire et son article ne seraient point.

Cette expérience m'a également permis de connaitre de nombreuses personnes, agrémentant chaque instant vécu. Que ce soit un diner en compagnie des membres du Laborius, une pause café philosophique avec les docteurs Aghile, Rekleitis, Parsa et Salerno ou même une conversation de cubicule avec les autres étudiants des Hautes-Baies, tous ces moments ont permis de tisser les liens de nouvelles amitiés. Un grand merci à l'équipe des technologies spatiales et tout particulièrement à vous Pierre, Régeant, Tom et Sébastien sans qui je n'aurais pu en apprendre tant sur Éclipse, ses plug-ins et les caprices du P2AT. J'ai également eu la chance de partager de bons moments avec mon successeur à l'agence, David Gingras, à qui je lève mon chapeau pour avoir su prendre brillamment le relais sur les recherches.

A special thank to you Ioannis, for these numerous discussions on robotics, particules filters, martial arts, sci-fi and politics while waiting for a path planner to complete its work. "Do you want to execute the path ? (Y/n)... Calibrating, please wait 30 sec... Execute path... Look ! It found a rock !"

Finalement, je n'aurais pu réaliser mon rêve d'étudier la robotique sans le support de mon épouse Katherine. Merci beaucoup de ne pas m'en vouloir pour ces soirées tardives passées à expérimenter un nouvel algorithme ou à compléter un article à la dernière minute.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ACRONYMES

Acronyme	Définition
2.5D	Deux et Demi Dimensions
2D	Deux Dimensions
3D	Trois Dimensions
A*	A Étoile
ASC	Agence Spatiale Canadienne
CNRS	Centre National de la Recherche Scientifique
CPU	Central Processing Unit
CSA	Canadian Space Agency
D*	D Étoile
DARPA	Defense Advanced Research Project Agency
DEM	Digital Elevation Map
EKF	Extended Kalman Filter
EM	Elevation Map
EMF	Eclipse Modeling Framework
ESA	European Space Agency
GB	GigaByte
GHz	GigaHertz
GIS	Geographic Information Systems
GMF	Graphical Modeling Framework
GPS	Global Positioning System
GVG	Generalized Voronoi Graph
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
ITM	Irregular Triangular Mesh
IR	InfraRouge
JPL	Jet Propulsion Laboratory
LAAS	Laboratoire d'Analyse et d'Architecture des Systèmes
LIDAR	LIght Detection And Ranging
MER	Mars Exploration Rover
MSL	Mobile Science Laboratory
NASA	National Aeronautics and Space Administration
RAM	Random-Access Memory
SLAM	Simultaneous Localization And Mapping
SWT	Standard Widget Toolkit
SONAR	SOund NAvigation And Ranging
SUMMITT	System for Unifying Multiresolution Models and Integrating Three-dimensional Terrains
UAV	Unmanned Air Vehicle
UGV	Unmanned Ground Vehicle

Suite sur la prochaine page

Acronyme	**Définition**
UML	Unified Modeling Language
XML	Extensible Markup Language

CHAPITRE 1

Introduction

À l'aube du 21e siècle, la quête de connaissance de l'Homme l'a mené à explorer de nouvelles frontières. En décembre 2003 et janvier 2004, les véhicules d'exploration martienne (*Mars Exploration Rovers*, ou MERs) "Spirit" et "Opportunity" ont atterri sur Mars et ont entrepris, pour la première fois, une exploration semi-autonome d'une autre planète [Biesiadecki et coll., 2007]. Cet exploit se distingue des missions d'exploration robotique précédentes qui ne possédaient pas à leur bord de système d'autonomie (Lunokhod [Carrier, 1992], Viking [Klein et coll., 1976], Sojourner [Matijevic et Shirley, 1997]). Les prochaines missions d'exploration de Mars sont celles du Mars Mobile Science Laboratory (MSL) mené par la NASA pour 2011 [Volpe, 2006] et d'ExoMars de l'Agence spatiale européenne (ESA) prévue pour 2016 [Vago, 2004]. Ces deux missions requièrent la capacité de voyager jusqu'à un kilomètre par jour.

Étant donné le coût extrêmement élevé de ces missions, on s'attend de ces dernières un gain important de connaissances. Par conséquent, de grands efforts sont faits pour optimiser la durée de vie de ces robots, leurs habiletés, ainsi que leurs ressources. L'un des problèmes majeurs rencontrés par ces robots est celui de la faible qualité du lien de communication avec la Terre : le temps pour l'aller-retour varie entre 8 et 40 minutes, les fenêtres de communication ne durent qu'une heure et sont espacées de 12 heures, tout en ayant une bande passante très étroite [Pedersen et coll., 2003]. En considérant que les MERs nécessitent de nombreuses interactions avec leur opérateur sur Terre avant d'accomplir une opération (typiquement, trois fenêtres de communication sont requises pour atteindre une roche, une fois que celle-ci a été identifiée par l'équipe de scientifiques), on comprend l'importance d'augmenter l'autonomie des robots planétaires. Donner la capacité aux robots d'exploration planétaire de se déplacer de manière autonome sur un terrain partiellement connu ou inconnu permettra une utilisation plus efficace de leur durée de vie utile. Lors des interruptions de communication avec la Terre, le véhicule pourra se déplacer de manière autonome vers un ou plusieurs sites, tandis que lors des fenêtres de communication, il maximisera l'utilisation de la bande passante pour retourner des données scientifiques et recevoir des instructions de haut niveau.

La navigation autonome sur de longues distances par un robot nécessite la capacité de cartographier son environnement, de s'y localiser et d'y planifier des chemins afin d'atteindre son but. Lors de l'exécution de ces tâches, un robot planétaire autonome accumule et utilise d'importantes quantités de données provenant de ses capteurs, des résultats d'expériences ainsi que d'autres informations pertinentes. Ces informations sont habituellement reliées aux cartes qui représentent le monde. Au fur et à mesure que le nombre de cartes augmente, les opérations simples de planification deviennent de plus en plus complexes, puisque plusieurs choix de cartes peuvent être faits. Les données provenant de différents capteurs, tels que les capteurs de distance LIDAR, les images des caméras monoscopique et stéréoscopique, etc., produisent différentes cartes. En outre, pour un même capteur, les données recueillies à différents moments et à partir de différents endroits produisent des cartes de résolution et de fidélité variables. Des décisions doivent alors être prises sur lesquelles des cartes d'une région donnée sont adéquates pour une opération de planification. Qui plus est, la planification sur de longues distances exige également une décision sur le moment de changer la carte utilisée.

Les cartes d'un environnement sont également cruciales pour la localisation. La sélection d'une carte appropriée à partir de la base de données est alors influencée par la résolution, le temps et les multiples possibilités d'intégration de cartes.

Les robots d'exploration planétaires autonomes représentent leur environnement à l'aide de cartes métriques. Ce type de représentation permet des calculs précis et la modélisation nécessaire pour la navigation autonome. On s'attend également à ce que ces cartes métriques évoluent dans le temps, au fur et à mesure que des sections inexplorées du monde sont visitées. L'introduction de nouvelle information dans la carte peut être faite en fusionnant la nouvelle information dans la carte existante. Cependant, toute erreur sur le positionnement du robot corrompt la représentation connue du monde avec une information inexacte. Pour cette raison, il est préférable de garder de multiples cartes locales au lieu de mettre à jour une seule et unique carte globale. Cette approche a l'avantage de conserver ensemble les données ayant des relations spatiales fortes, et de relier souplement les données ayant une relation spatiale plus faible. Avec cette approche, l'incertitude associée à l'origine d'une carte peut toujours être réduite plus tard puisqu'elle n'est jamais fondue avec les données [Grisetti et coll., 2007].

Cette représentation de l'environnement doit être segmentée en de multiples cartes locales, et leur nombre augmente au fur et à mesure que le robot explore de nouveaux terrains.

Une gestion efficace de ces cartes locales est alors nécessaire afin de maintenir l'uniformité du monde ainsi divisé.

Après l'étude de différents systèmes de gestion des données géo-référencées utilisés par des robots autonomes, l'on dénote une sous-exploitation du potentiel de ces données. Ces dernières sont habituellement utilisées au moment de leur acquisition et ne sont plus réutilisées par la suite. Nous posons donc l'hypothèse que si ces données sont entreposées dans un système adéquat, un robot peut alors en exploiter leur plein potentiel et, par le fait même, améliorer son comportement de navigation. La première contribution de cette recherche est de dresser une liste des caractéristiques qu'un tel système de gestion des données devrait considérer. La deuxième contribution est de démontrer la faisabilité d'un tel système par la présentation d'une implémentation logicielle du système Atlas. Cette dernière est soumise à certaines restrictions de conception imposée par l'Agence spatiale canadienne. La solution proposée doit être intégrée à leur plateforme robotisée mobile et doit être conçue par méta-programmation (programmation par modèles) à l'aide de l'environnement EMF d'Éclipse.

La solution présentée est capable de contrôler dynamiquement un grand ensemble de cartes tout en considérant les problèmes inhérents à l'opération de cartographie ainsi qu'à son utilisation. Les problèmes considérés sont : la large variété de contenu de données, l'incertitude dans les relations spatiales entre les cartes, la planification de trajectoires par de multiples cartes, et la corrélation des cartes dans les opérations de localisation.

Ce document présente les travaux relatifs aux systèmes de gestion de cartes et de ses domaines adjacents dans le chapitre 2. Un aperçu des défis entourant le problème de la gestion de cartes est exposé au chapitre 3. Le contexte expérimental ayant permis de valider l'approche proposée se retrouve au chapitre 4. Le chapitre 5 présente l'article scientifique *Multi-Layer Atlas System for Map Management* soumis à la conférence *International Conference on Field and Service Robots 2009*. Cet article porte sur la conception d'un tel système de gestion de données, ainsi qu'une solution permettant de gérer dynamiquement une variété de données, l'incertitude des relations spatiales entre deux cartes, de procurer un mécanisme de planification de chemins au travers des cartes, ainsi que la corrélation des cartes pour les opérations de localisation. Cet article présente également les résultats expérimentaux sur l'utilisation du système de gestion de données Atlas par un véhicule d'exploration autonome.

CHAPITRE 2

Cartographier un monde

La gestion des cartes d'un environnement par des robots autonomes est assurément à ses débuts et très peu de progrès ont été faits dans ce domaine. Ceci peut être attribué au fait que la technologie d'un tel gestionnaire n'est pas requise pour la majorité des systèmes robotiques.

Les techniques de base requises pour la navigation autonome sont maintenant connues, voir même maîtrisées pour les environnements statiques, et les robots en possèdent diverses implémentations fiables [Rekleitis et coll., 2007a, 2008a,b]. La technologie de gestion de cartes peut alors être explorée afin d'accroître le niveau d'autonomie des robots.

Ce chapitre présente une revue des recherches dans le domaine des gestionnaires de cartes. Puisque ce domaine de recherche est relativement jeune, une vue d'ensemble des domaines connexes est également présentée. Ces domaines ont été choisis en fonction de leurs impacts sur la façon dont les cartes sont crées ou utilisées et, ce faisant, sur la façon dont les cartes devraient être gérées. Ces champs complémentaires sont donc : l'exploration autonome planétaire et de terrains difficiles, la représentation de cartes en 3D (trois dimensions), le recalage (alignement de cartes, localisation, et SLAM), ainsi que les planificateurs de trajectoires.

2.1 Atlas et systèmes de gestion de cartes

Souvent, les systèmes de gestion de cartes se retrouvent dans la littérature sous différents noms (atlas, environnements de modélisation, bases de données) et parfois sont fusionnés dans les procédés de localisation et de cartographie. Une revue des systèmes de cartographie trouvés dans divers systèmes autonomes est donc présentée.

Bosse et coll. proposent un système de gestion de cartes basée sur l'étude du concept d'un atlas conventionnel [Bosse et coll., 2004]. Dans la solution proposée du *SLAM in large-scale cyclic environments*, les auteurs proposent l'utilisation d'un environnement d'atlas. Cet environnement est basé sur un système hybride de cartes métrique/topologique. Au plus haut niveau, un graphe est employé pour représenter le monde et où les sommets de graphe représentent les endroits, et les liens représentent la transformation entre ces

endroits. À un niveau plus bas, chaque endroit est décrit comme une carte métrique locale en 2D (deux dimensions). Cette organisation des cartes locales permet à l'incertitude d'être modélisée dans chaque cartes, et non pas en fonction d'une référence globale. Le modèle d'incertitude emploie une représentation gaussienne de l'erreur. Finalement, l'environnement proposé d'atlas est utilisé pour transformer l'incertitude d'états entre les cartes locales afin de créer de nouvelles cartes, de manipuler des situations de fermeture de boucle, et pour générer et évaluer des hypothèses concurrentes de la meilleure carte locale représentant l'état du système courant.

Un autre système similaire à l'atlas est proposé par [Lisien et coll., 2005]. Semblable à [Bosse et coll., 2004], une représentation hiérarchique est employée. Au plus haut niveau, une représentation topologique est calquée sur le graphe généralisé de Voronoi (GVG) de l'environnement cartographié. Cependant, ce sont les transitions du graphe qui sont représentées au plus bas niveau comme une collection de caractéristiques. Le GVG a l'avantage d'être incrusté à même l'environnement, et peut donc être facilement extrait lors de l'exploration. Qui plus est, l'ensemble des caractéristiques d'une carte représente l'environnement local depuis la perspective d'un nœud du GVG vers un nœud voisin. Chaque carte de transitions représente les discontinuités de l'environnement sous la forme d'une pose relative et d'une incertitude gaussienne. Le système proposé d'atlas a l'avantage d'être non demandant en mémoire et en calcul, mais est fortement sensible aux changements de l'environnement tout en étant restreint à une utilisation dans des petits espaces, riches en caractéristiques.

Un agencement hiérarchique des cartes est exploré différemment par [Kelly et coll., 2006] en maintenant trois niveaux de cartes selon des périodes décroissantes d'acquisition de données et des niveaux croissants du détail des données. Les cartes des niveaux inférieurs sont égocentriques au robot, tandis que la carte de plus haut niveau est une représentation globale du monde. Les trois cartes utilisées sont une grille volumétrique, une carte locale d'élévation et une carte globale de traversabilité. La création/mise à jour du contenu des cartes se propage de la carte la plus détaillée vers la carte globale. De plus, la carte globale se compose de canaux multiples. Par conséquent, des cartes de traversabilité de différentes sources et résolutions peuvent être présentées sur la même carte (par exemple, des données du robot, d'un véhicule aérien autonome (UAV), et de l'information satellite). L'arrangement hiérarchique de cartes a l'avantage de permettre le recalage de la couche globale de cartes sans affecter le contenu des cartes plus basses. Cependant, cela implique la nécessité de maintenir une carte 3D complète du monde en mémoire. De plus, ce système

fusionne les données lors du processus de mise à jour des cartes, pouvant avoir comme conséquence la corruption du contenu des cartes par l'introduction permanente de bruit.

Le Jet Propulsion Laboratory (JPL) présente leur suite d'outils de modélisation de terrains SimScape dans [Jain et coll., 2006]. Son but est de fournir une infrastructure commune pour représenter les données de modélisation de terrains provenant de multiples sources, et de les rendre disponibles aux applications de simulation. Cette suite d'outils supporte plusieurs représentations de la géométrie de terrain telles que les cartes d'élévation numérique 2.5D, des nuages de points, des maillages 3D et maillages triangulaires irréguliers 2.5D (ITM). Elle gère également la conversion entre les différentes représentations de modèles de terrain et la génération de modèles de terrain composites provenant des modèles hétérogènes.

En conclusion, le système de gestion de cartes employé par l'opérateur des MERs est présenté dans [Olson et coll., 2007]. SUMMITT (*System for Unifying Multi-resolution Models and Integrating Three-dimensional Terrains*) est une suite d'outils logiciels fournissant la modélisation de terrains pour l'opérateur des MERs. Cet environnement immersif manipule des images de résolutions et d'échelles différentes provenant d'une multitude de sources comme des images de descente, des images de surface de l'atterrisseur et du robot d'exploration, aussi bien que des images orbitales. Pour un site donné d'exploration, SUMMITT contrôle le processus de recalage d'images, la conversion des images stéréo en primitives volumétriques (*voxels*), et le stockage/fusionnement des *voxels* [Kaufman et coll., 1993] dans une structure d'*octree* [Jackins et Tanimoto, 1980]. Une fois que l'emplacement est modélisé dans un *octree*, SUMMITT est capable de générer une représentation polygonale du site (pour la visualisation et l'évitement de collisions) aussi bien qu'une carte d'élévation.

2.2 Exploration autonome de planète et de terrains difficiles

Il existe plusieurs systèmes robotiques réalisant divers niveaux de navigation autonome. Ils abordent tous le même problème en utilisant différentes techniques de détection et de locomotion. Il est toutefois possible d'y extraire quelques conditions générales afin de définir les caractéristiques composant le cœur de l'autonomie. En d'autres termes, pour réaliser une navigation autonome, les tâches secondaires suivantes doivent être résolues :

1. Recherche d'un objectif.
   La première tâche est le choix d'un objectif à atteindre. Par définition, la navigation sous-entend le concept de guidage d'un déplacement vers un endroit plus ou moins précis. Un robot doit donc être capable de reconnaître ces endroits et conséquemment de se situer dans l'environnement, définissant ainsi le concept de localisation. En interprétant les changements dans son environnement entre deux lieux adjacents ou lors de son déplacement, un robot peut alors reconnaître si son objectif de navigation est atteint ou non. La plupart du temps, le robot possède une connaissance a priori du monde lui permettant de combler ce préalable ; autrement, elle doit être créée par exploration.
2. Représentation de l'environnement immédiat.
   Pour naviguer adéquatement dans un environnement, un robot doit pouvoir le représenter. Cette représentation est également nécessaire à la localisation et à la planification de chemins. Différents capteurs sont disponibles pour s'acquitter de cette tâche, et leur choix a une répercussion sur la manière d'opérer la navigation. De tels capteurs incluent le SONAR, l'infrarouge et les scanneurs à laser, caméras stéréo. etc.
3. Estimation de la pose du robot.
   La pose d'un robot se définit selon deux paramètres, soit la position et l'orientation dans un espace en 3D. Cette pose s'exprime habituellement à l'aide de six paramètres $P = [x, y, z, r, p, \theta]$, où $[x, y, z]$, est la position cartésienne en 3D, et $[r, p, \theta]$, l'orientation du robot selon trois axes. Deux types de capteurs sont utilisés pour estimer la pose : proprioceptif, avec des capteurs qui mesurent la pose avec des mesures du robot, telles que les encodeurs odometriques, les centrales inertielles, etc. ; et extéroceptif, avec des capteurs qui mesurent la pose relativement à l'environnement, tel que le sonar, l'infrarouge, le radar, les scanneurs à laser, les GPS, et les caméras. Quand la pose du robot est estimée en utilisant uniquement des mesures successives entre les poses, l'erreur accumulée en raison du bruit des capteurs augmente sans limite. Quand le robot estime sa pose à partir d'une référence globale, l'incertitude devient limitée. Les systèmes autonomes emploient l'une de celles-ci ou une combinaison des deux pour estimer leur pose.
4. Planification du chemin.
   Pour atteindre son objectif, le robot doit tracer un chemin qui le mènera de sa pose courante à la pose de destination. Le processus est généralement divisé en deux étapes. En premier lieu, un chemin brut est tracé pour guider le robot vers l'objectif (planification globale). Ceci est accompli en utilisant la connaissance a priori générale

du monde, décrivant comment des endroits sont reliés. Dans une deuxième phase, l'environnement observé est employé pour tracer un chemin précis, libre d'obstacles, et qui suit le chemin brut (planification locale).

5. Exécution de la trajectoire.
   Pour terminer, le robot exécute la trajectoire déterminée. Cette opération consiste à utiliser les actionneurs du robot lui permettant de se déplacer, tout en essayant de reproduire le plus fidèlement possible le chemin pré-établi. Cependant, certains évènements extérieurs au robot peuvent survenir (glissement des roues, enlisement, collisions, etc.), compromettant alors le résultat escompté. Différentes techniques peuvent être utilisées pour réagir à ces évènements impromptus, tels que l'évitement d'obstacle dynamique ou l'utilisation de différents comportements.

2.2.1 Exploration planétaire

La recherche sur la navigation autonome est particulièrement utile pour le robot d'exploration planétaire. D'importants travaux du Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) sont présentés dans [Lacroix, 2006]. Il propose différents systèmes autonomes pour réaliser une exploration planétaire. Leur robot d'exploration Lama exploite un châssis à configuration flexible lui permettant de se déplacer sur des terrains difficiles. Le robot Dala est une plate-forme d'essais utilisée pour divers développements tels que la gestion de tâches et la re-planification en ligne. Le LAAS étudie également le secteur du véhicule aérien non piloté (UAV) avec le robot dirigeable Karma, et plus récemment avec le drone Lhassa. L'étude d'autonomie est orientée vers la détection visuelle basée sur la vision monoscopique et stéréoscopique, utilisée avec les capteurs classiques et panoramiques. Le laboratoire explore également l'odométrie 3D et visuelle pour gérer l'évaluation de pose de leurs systèmes. Du côté de la planification de chemins, différentes techniques telles que les champs de potentiels et les arcs élémentaires sont considérées.

En partie basée sur ce travail, la NASA a développé le dernier cri des robots d'exploration planétaire "Spirit" et "Opportunity" au *Jet Propulsion Laboratory* (JPL) [Maimone et coll., 2006]. Le système de navigation est capable d'exécuter un déplacement sur des dizaines de mètres par jour. Utilisant la vision stéréo, l'observation faite du terrain est utilisée pour créer le modèle en 3D de l'environnement à l'aide de voxels. Des systèmes similaires de modélisation de terrain et de localisation visuelle sont également développés en industrie par MDA [Barfoot et coll., 2006]. Les algorithmes d'estimation de position des MERs emploient la fusion d'odométrie de roue et visuelle, tandis que l'estimation de l'orientation est fournie par une IMU (*Inertial Measurement Unit*) de haute précision. Au

niveau de la planification de chemins, les opérateurs sur Terre fournissent la planification globale sous la forme de séquence de lieux à atteindre. Cependant, la planification locale peut être raffinée de façon autonome avec l'utilisation de projections multiples de chemins sur le modèle du terrain 3D.

Actuellement, l'Agence spatiale canadienne (ASC) étudie le domaine de la navigation à longue portée nécessaire à la prochaine génération de robots d'exploration planétaire [Rekleitis et coll., 2007a, 2008a,b]. Leur approche de l'autonomie requiert une connaissance a priori du monde sous la forme de cartes basse résolution produites en utilisant des images orbitales ou de l'information provenant des missions précédentes. L'observation de l'environnement est faite activement à l'aide d'un scanneur laser, et produit une représentation basée sur un maillage. Cela représente une innovation puisque jusqu'à présent, seuls des capteurs passifs ont été employés par les systèmes autonomes similaires. L'évaluation de la pose du robot est maintenue en fusionnant différents capteurs tels qu'une IMU, l'odométrie de roue et l'azimut (champ magnétique ou observation du soleil). Finalement, ils emploient un planificateur de chemins à deux niveaux (local et global) basé sur des algorithmes efficaces de recherche de graphe.

2.2.2 Navigation sur terrains difficiles

Un autre domaine de recherche sur les systèmes robotiques se concentre sur le développement de la navigation autonome. Fortement subventionné par les militaires américains, son but est l'étude de véhicules autonomes pouvant traverser des terrains difficiles. Le *Defense Advanced Research Project Agency* (DARPA) finance plusieurs projets dans ce domaine. L'agence est également connue pour son concours *DARPA Grand Challenge* qui a offert un prix d'un million de dollars US au gagnant d'une course de véhicule terrestre non pilotée (UGV) en 2004. Ce n'est qu'en 2005 que l'UGV Stanley de l'écurie Stanford remporta ce défi en 6 h 53, atteignant une vitesse moyenne de 30.90 km/h et gagnant un prix de deux millions de dollars [Thrun et coll., 2006].

Cette course a eu lieu dans le désert de Mojave. Les véhicules devaient suivre un parcours prédéfini. Des défis sous la forme de longs tunnels, de portes, d'étroits viaducs et de routes plus ardues parsemaient ce parcours. L'édition 2007 de cette course a poussé la navigation autonome des véhicules une étape plus loin en transportant ce défi dans un environnement urbain.

Un autre programme du DARPA qui a également connu un énorme succès en navigation autonome est le projet PerceptOR [Kelly et coll., 2006]. Ce véhicule autonome a été

développé par le Robotics Institute de l'université de Carnegie Mellon, et emploie un double système de robot basé sur un UGV et un véhicule aérien non piloté (UAV). Il est capable d'opérer sous le feuillage d'une forêt, dans les hautes herbes et les buissons, autour d'arbres jonchant le sol, dans un environnement montagnard, et sur plusieurs autres types de terrains.

2.3 Représentation en 3D

La majorité des travaux portant sur la représentation en 3D est orientée vers la création de modèles pour des environnements urbains ou des objets synthétiques. Comme exprimé par [Remondino et El-Hakim, 2006] dans sa revue du domaine, la modélisation de terrain en 3D peut être classée selon les catégories suivantes : modélisation par imagerie, modélisation par distance et une combinaison des deux. Les deux premières catégories sont couramment utilisées par des systèmes autonomes, tandis que la combinaison de l'image et de la modélisation par distance est plus rare.

2.3.1 Modélisation par images

La majeure partie du travail sur la modélisation en 3D se base sur l'imagerie. Dans sa revue sur la modélisation en 3D par l'imagerie planétaire, [Remondino et El-Hakim, 2006] présentent les différentes approches trouvées dans la littérature. Ils analysent ce processus depuis la conception d'un réseau de capteurs, passant par l'extraction de caractéristiques et les procédés de mesure, pour finalement présenter une série de techniques de reconstruction des surfaces à partir de nuages de points en 3D et de texturisation.

La modélisation par imagerie est également une approche populaire pour les robots autonomes [Gonzalez-Barbosa et Lacroix, 2005; Lacroix et coll., 2001; Olson et coll., 2007]. Il n'est pas étonnant de la retrouver sur ces systèmes puisqu'elle a l'avantage d'être économique, portable, polyvalente et consomme peu d'énergie. Typiquement, la région active de sa profondeur de champ est plus courte, mais a l'avantage de fournir des modèles réalistes sans coûts supplémentaires. D'ailleurs, cette technique s'est montrée efficace avec le succès des robots d'exploration planétaire "Spirit" et "Opportunity", qui ont traversé plus de 2.5 km utilisant la vision stéréo [Barfoot et coll., 2006; Olson et coll., 2007].

Habituellement, le processus de modélisation de surfaces en 3D à l'aide d'images comprend les étapes suivantes. Avant tout, il faut s'assurer d'utiliser une paire d'images ou une série d'images consécutives prise à intervalles de temps fixe. La première étape consiste à obtenir

l'endroit et l'orientation de la prise de vue. Malheureusement, cette information n'est pas toujours disponible et des estimations sont utilisées aux dépens d'une perte de qualité dans le modèle qui en résulte. La deuxième étape est celle de l'extraction de caractéristiques de chacune des images et de l'association de ces dernières avec celles d'autres images. Basée sur le modèle du réseau de capteurs, la paire de caractéristiques en 2D est convertie en un point en 3D. À ce stade du traitement, les modélisations par image et par distance se rejoignent puisqu'ils sont représentés par des nuages de points en 3D et peuvent dès lors être traitées de la même manière.

2.3.2 Modélisation par distance

Les robots autonomes utilisent une variété de capteurs de distance. Un capteur employé couramment est le SONAR, qui exploite les propriétés de la propagation du son dans l'air ou l'eau pour déterminer la distance des objets. La distance peut également être mesurée à l'aide de lumière infrarouge (IR) pulsée. La lumière voyageant dans l'espace est reflétée par la surface d'objets et revient finalement au capteur. L'intensité mesurée de la lumière est utilisée pour estimer la distance. Ces deux capteurs ont l'avantage d'être économiques, mais ils manquent de fiabilité et de précision. Plus récemment, le scanneur laser est devenu plus accessible aux systèmes robotique. Les LIDARs sont retrouvés sous forme de scanneurs de ligne en 2D ou de balayage 3D, procurant un capteur de distance beaucoup plus fiable.

La modélisation par distance de surfaces crée un nuage de points en 3D. Son nombre de points augmente rapidement en fonction de la surface modélisée, tout comme la complexité d'analyser ces données. C'est pour cette raison que la modélisation par distance tend à utiliser une représentation compressée de l'espace. Parmi différentes approches de compression de données utilisées à cette fin, les plus communes sont :

1. Carte d'élévation.

   Une façon de compresser un nuage de points en 3D est par la réduction de l'espace dimensionnel à 2.5D. Les cartes d'élévation (*Elevation Map*, ou EM) appliquent cette technique en discrétisant un volume dans une grille en 2D pour laquelle chaque cellule représente une valeur d'élévation. Typiquement, les surplombs ou les cavernes ne peuvent pas être représentés convenablement par les EMs, puisque plus d'une surface existe pour une position donnée en 2D.

   Les EMs classiques utilisent une structure de données *quadtree* pour implémenter leurs grilles 2D [Gowdy et coll., 1991]. Récemment, une nouvelle approche a été

proposée en combinant une EM avec des couches multiples [Pfaff et coll., 2007; Triebel et coll., 2006], permettant ainsi la modélisation de surplombs. Cependant, cette approche ne règle pas les problèmes reliés à l'utilisation de mémoire et de recherche de chemins dans la grille.

2. Maillage triangulaire irrégulier.

   Une autre manière de réduire les besoins en mémoire des nuages de points en 3D est en modélisant plusieurs points comme surface triangulaire. Les triangles sont construits de manière à représenter les données originales en tolérant une erreur maximale. Ainsi, des zones riches d'information sont modélisées par plusieurs petits triangles, tandis que les zones planes sont modélisées par quelques grands triangles. Le ITM (*Irregular Triangular Mesh*) accomplit ceci en enlevant les points faisant partie de surfaces planes, et ce faisant, laisse la plupart des points pour décrire les zones riches d'information. Cette représentation a l'avantage de pouvoir modéliser les structures concaves tout en conservant le contenu scientifique retrouvé dans les données topographiques.

   Les travaux précédents sur les ITMs ont exploré la décimation des nuages de points par l'approximation planaire et le fusionnement des surfaces [Hart et coll., 1968; Thrun et coll., 2003]. Cette approche statistique s'est avérée efficace pour des environnements urbains et de mines souterraines, bien qu'elle nécessite des ressources importantes en calculs. La décimation basée sur la tolérance d'erreur a également été utilisée par un banc d'essai robotique d'exploration planétaire [Bakambu et coll., 2006].

3. Grille volumétrique.

   Semblable à la carte d'élévation, la grille volumétrique numérise l'espace dans une grille en 3D. Les cellules de la grille contiennent un ensemble de points 3D ou une valeur décrivant la densité du volume. Cette représentation n'est pas nécessairement efficace en ce qui concerne l'utilisation de l'espace mémoire, mais elle reste simple. L'implémentation des grilles volumétriques utilise habituellement une structure *octree* pour réduire l'espace mémoire nécessaire à la représentation [Noborio et coll., 1988]. L'octree est une structure d'arbre où un nœud représente un point dans un cube et où trois axes orthogonaux se croisent. Chaque nœud d'arbre définit huit nœuds secondaires, représentant les huit cubes définis par les axes. Cette représentation est également utilisée pour fusionner des nuages de points et pour la représentation locale d'un environnement [Kelly et coll., 2006; Olson et coll., 2007].

2.4 Recalage (alignement de cartes, localisation, et SLAM)

L'estimation précise de la pose est un élément clé pour une navigation robuste. Avec le temps, l'estimation de la pose du robot accumule des erreurs de diverses sources. Afin de maintenir une localisation d'une précision acceptable, des techniques probabilistes doivent être utilisées pour abaisser le niveau d'incertitude accumulée. Une pratique commune consiste à échantillonner sans interruption l'environnement tout en se déplaçant et utiliser l'entrée sensorielle pour déterminer le déplacement du robot. La pose ainsi inférée des données sensorielles est alors combinée à la pose prévue du robot par des techniques probabilistes pour produire une estimation plus précise de la pose du robot au fil du temps. Cette approche est connue sous le titre de la localisation et cartographie simultanée (SLAM).

Cependant, la mesure d'un déplacement à l'aide d'une paire de modèles en 3D de l'environnement implique l'utilisation d'opérations de corrélation des surfaces. Différentes approches ont été recensées dans la littérature [Campbell et Flynn, 2001; Planitz et coll., 2005]. L'approche générale consiste à associer les caractéristiques de deux modèles de surfaces. Ces propriétés sont qualifiées d'intrinsèques quand elles sont uniquement liées à la surface elle-même (c.-à-d., distance de deux points sur la surface), et d'extrinsèques quand elles sont liées à l'espace dans lequel la surface se situe (c.-à-d., un plan tangent à un point de la surface). Les algorithmes de corrélation sont donc classifiés en deux catégories basées sur la nature intrinsèque ou extrinsèque des propriétés de surfaces qu'elle exploite. Les algorithmes intrinsèques sont généralement utilisés pour trouver une correspondance brute, puisqu'ils se basent seulement sur les propriétés de la surface. Les algorithmes extrinsèques sont fonction de leur espace environnant et, par conséquent, sont limités à des opérations de raffinement de la corrélation.

Dans le contexte d'applications robotiques, le raffinement de l'alignement des surfaces utilise presque uniquement l'algorithme *Iterative Closest Point* (ICP) [Besl et McKay, 1992; Chetverikov et coll., 2002; Rusinkiewicz et Levoy, 2001]. D'une autre part, pour obtenir un alignement brut des surfaces, différents algorithmes sont alors exploités. Bien souvent, plus d'un algorithme est employé pour obtenir la meilleure évaluation. Les algorithmes communément utilisés pour l'alignement brut incluent le *Spin-Image* [Johnson, 1997; Johnson et Hebert, 1999], le *Point Fingerprint* [Sun et coll., 2003], le *Harmonic Shape Image* [Zhang, 1999], et beaucoup d'autres.

2.4.1 Cartographie et localisation simultanée en 3D

Amplement étudié [Filliat et Meyer, 2003; Meyer et Filliat, 2003], le SLAM est considéré comme problème résolu pour de petits environnements statiques en 2D (trois degrés de liberté, soit x, y et l'orientation) [Durrant-Whyte et Bailey, 2006]. Cependant, il est encore loin d'être résolu dans un contexte de cartographie de grands environnements, de modélisation d'environnements complexes et dynamiques, de SLAM multivéhiculaire et de SLAM en 3D (six degrés de liberté).

Beaucoup d'approches au SLAM en 3D sont étudiées. Une avenue de solutions considère la création d'une représentation virtuelle du monde en 2D depuis les données en 3D. Avec cette représentation, il est possible d'employer les techniques de SLAM en 2D pour résoudre le problème [Brenneke et coll., 2003]. Une approche différente explore l'extraction des caractéristiques visuelles en 3D depuis une source de vision stéréo, et infère le déplacement en 3D à l'aide d'un filtre de Kalman étendu (EKF) [Jung et Lacroix, 2003; Lemaire et coll., 2005], où même d'un filtre particulaire Rao-Blackwellised [Sim et Little, 2006]. Cependant, des améliorations importantes au SLAM visuel sont réalisées en combinant l'odométrie en 3D (mesure d'inertie) avec un EKF [Davison et Kita, 2001; Lamon et Siegwart, 2004]. Plus récemment, [Pomerleau, 2008] propose une optimisation d'un algorithme de recalage des surfaces appelé Kd-ICP afin de permettre une utilisation en temps réel compatible au SLAM. Son approche tire avantage de la fusion d'information spatiale avec une représentation simplifiée des couleurs permettant alors de réduire l'influence de la luminosité sur le processus de recalage.

2.5 Planification de trajectoires

La planification de trajectoires implique de trouver un chemin permettant de se diriger d'un point A vers un point B, tout en évitant les obstacles. Habituellement, ce processus est divisé en deux étapes. L'environnement immédiat observé est utilisé pour la planification locale tandis que la connaissance plus brute a priori de l'environnement est employée pour la planification globale. Cette distinction est principalement due à deux facteurs : la variation de la résolution de l'environnement discrétisé et le besoin de réaction rapide aux changements de l'environnement entourant le robot. Pour ces mêmes raisons, différentes techniques sont utilisées.

Pour la planification locale, le choix des algorithmes dépend fortement de la représentation d'environnements disponibles. Une technique bien étudiée est celle des champs de

potentiels [Slack, 1993]. Il faut savoir que cette technique est extrêmement sensible aux minimums locaux dans lesquels le robot peut potentiellement se retrouver bloqué. Une autre approche possible est d'utiliser de multiples chemins prédéfinis [Maimone et coll., 2006]. Ces chemins sont projetés sur le modèle local de terrain et le chemin le plus approprié est choisi ; les robots d'exploration martienne "Spirit" et "Opportunity" emploient cette technique. De manière similaire, [Kelly et coll., 2006] génère une série de trajectoires basées sur la simulation de projections de la dynamique du véhicule.

D'une autre part, la planification globale est généralement résolue avec l'aide d'algorithmes de recherche basés sur les graphes. L'un des plus fréquemment utilisés est l'algorithme *Shortest-Path* de Dijkstra [Dijkstra, 1959]. Malgré qu'il est considéré comme lent, il a l'avantage de fournir une solution optimale. L'algorithme bien connu de A* est souvent employé pour son traitement rapide [Hähnel et coll., 2003]. Utiliser cet algorithme avec une fonction d'heuristique appropriée donne une solution optimale. Un autre algorithme appelé D* fournit une planification optimale et efficace de chemins pour les environnements partiellement connus [Stentz, 1994]. Reconnu pour être plus lent que A* à l'étape de planification, il a l'avantage d'être beaucoup plus efficace lors de la re-planification de chemins. Plus récemment, l'algorithme *Field D** [Ferguson et Stentz, 2005] améliore D* en utilisant une interpolation entre les nœuds et, par conséquent, produit une trajectoire plus douce. Il faut noter que le véhicule autonome PerceptOr emploie avec succès cette méthode pour la planification et la re-planification à long terme [Kelly et coll., 2006].

CHAPITRE 3

Défis de la gestion de cartes

Les robots d'exploration planétaires peuvent utiliser une variété de données géo-référencées : imagerie satellite ; images basées sur une vision monocopique, stéréoscopique, ou omnidirectionnelle ; données de distance (LIDAR) comme des nuages de points 2.5D ; ou même des données scientifiques telles que l'information minéralogique obtenue à partir des spectromètres. Tous ces types de données géo-référencées permettent de représenter différents aspects de l'environnement pour un lieu donné. Maintenir cette variété de données a de multiples implications pour un système de gestion : formats de données multiples, différences dans la résolution, la précision et l'incertitude, l'âge des différents éléments, etc.

Cette recherche prend en compte uniquement les données qui sont référencées dans l'espace. Par conséquent, il existe des relations spatiales reliant toutes les données accumulées. Une relation spatiale forte existe pour des données provenant d'un même capteur, affecté seulement par l'erreur de ce dernier, ou lorsque différentes lectures de capteurs sont acquises en même temps et au même endroit, affectées cette fois-ci par l'erreur d'alignement [Mirzaei et coll., 2007]. Cependant, quand deux ensembles de données ont été acquis en des temps et endroits différents, leur corrélation spatiale est beaucoup plus faible en raison des erreurs de localisation. Un système de gestion de cartes doit être capable de manipuler ces relations spatiales fortes/faibles entre les éléments contenus.

La planification de trajectoires sur plusieurs cartes apporte également sa part de défis. Dans le cas de missions planétaires exigeant la navigation autonome sur de grandes distances, le processus de planification de chemins est typiquement décomposé en deux processus distincts : la planification globale de chemins en utilisant une carte approximative du monde, et la planification locale de chemins, utilisant une représentation de l'environnement de grande précision, acquise lors du déplacement. Les défis surgissent quand la planification doit être faite au travers de plusieurs cartes. Dans ce cas-ci, des décisions doivent être prises concernant quelles cartes utiliser en se basant sur des facteurs tels que la zone couverte, la résolution, et les propriétés temporelles des données. Là où les cartes se recouvrent, le système doit être capable de manipuler les transitions d'une carte à l'autre aussi bien que les contradictions entre celles-ci. De plus, lors du déplacement,

de nouvelles cartes peuvent être ajoutées comme une mise à jour de la vue satellite, ou l'addition de cartes locales à la représentation du monde. L'intégration de la nouvelle information en direct peut améliorer la qualité globale du chemin par des opérations locales de planification.

De même, le processus de localisation des véhicules d'exploration impose également des restrictions à un système de gestion de cartes. Comme la planification de chemins, la localisation peut être décomposée en deux processus distincts. Dans la localisation globale, le robot emploie la connaissance de son environnement immédiat et essaye de s'y localiser relativement à une carte de la région (habituellement une carte à basse résolution couvrant une grande surface). Le deuxième processus, le recalage de cartes, est utilisé pour relier ensemble les cartes locales et pour y corriger l'erreur due à l'odométrie sur des acquisitions successives. Dans les deux cas, le processus de localisation tente de détecter des caractéristiques identifiées dans une carte en les associant à des caractéristiques de la deuxième carte. La position relative est alors inférée depuis ces associations. Malheureusement, puisque les deux cartes sont prises selon différentes perspectives, une caractéristique présente dans une carte pourrait être occluse ou absente dans la deuxième. Les cartes peuvent également avoir différentes résolutions, causant les caractéristiques se retrouvant dans les deux cartes d'être échantillonnées avec différentes précisions, rendant le processus d'association plus difficile. Tous ceux-ci pourraient mener à une association erronée des caractéristiques, causant une localisation relative incorrecte.

CHAPITRE 4

Contexte expérimental

Afin de valider le système de gestion de données qui est présenté au chapitre 5, il est nécessaire de définir un contexte expérimental dans lequel ce système est étudié. Ce dernier est conçu pour une utilisation robotisée dans un processus d'exploration planétaire autonome, avec un robot dont les capacités sont décrites à la section 4.1. Le comportement du robot est programmé dans un environnement logiciel qui est décrit à la section 4.2. Il est également primordial de donner un aperçu de l'environnement physique dans lequel ce robot évolue telle que le présente la section 4.3, puisque l'interaction de l'environnement influence directement la lecture de ces capteurs et conséquemment, son comportement de navigation. Finalement, un aperçu du processus de navigation est présenté à la section 4.4, permettant de dresser un portrait complet du contexte expérimental.

4.1 Plateforme robotisée

Le robot utilisé est une plateforme P2-AT de la compagnie Pioneer qui a été modifiée par l'Agence spatiale canadienne (ASC) afin de créer leur plateforme mobile robotisée. La base robotisée du P2-AT utilise des encodeurs optiques au niveau de ses roues pour déterminer son déplacement selon le plan cartésien XY. L'orientation du robot autour de l'axe Z est donnée par la lecture d'une boussole électronique (TCM2 d'ActivMedia Robotics). Une série de capteurs de proximité SONAR et infrarouge sont également disposés en périphérie du robot, mais n'ont pas été utilisés lors des expériences.

À cette base robotisée, l'ASC a ajouté un ordinateur Toughbook de Panasonic muni d'un processeur Intel Core 2 Duo de 1.60GHz et disposant de 3GB de RAM. Cet ordinateur utilise un système d'exploitation Ubuntu avec un noyau Linux 2.6. C'est ce dernier qui exécute le comportement de navigation et qui utilise le système de gestion de données étudié. Ont également été ajoutés une centrale inertielle (IMU 300CC-100 de Crossbow) à six degrés de liberté et un laser (LMS200 de SICK) monté sur une base pivotante permettant d'obtenir une représentation en 3D de l'environnement du robot sur 360 degrés. Le schéma de la figure 4.1 montre les différentes composantes de cette plateforme mobile robotisée. Cette plateforme dispose d'une odométrie en six dimensions, soit en position

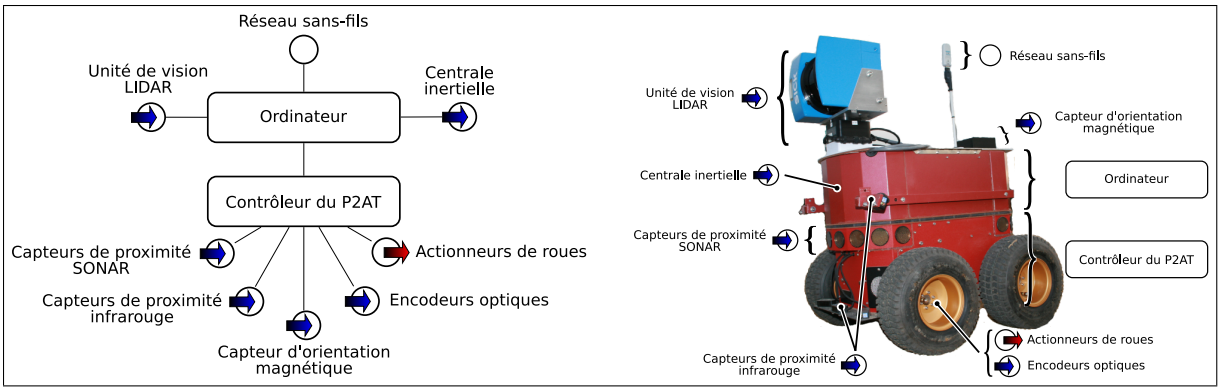


Figure 4.1 Schéma des composantes de la plateforme mobile robotisée de l'Agence spatiale canadienne.

(x, y et z) et en orientation (θ_x, θ_y et θ_z), ainsi qu'une représentation des surfaces aux alentours du robot dans un rayon d'environ quinze mètres.

4.2 Environnement logiciel

Le contexte logiciel sélectionné pour le développement et l'exécution de l'Atlas est celui d'Éclipse 3.4 couplé d'une machine virtuelle Java 1.6.0 de Sun Microsystems. Le choix d'utiliser Éclipse et Java découle du préalable voulant que le développement de l'Atlas fasse partie intégrale de la plateforme de robotique mobile de l'Agence spatiale canadienne.

Le logiciel de la plateforme de robotique mobile est conçu de manière modulaire Il préconise une approche multi-plateforme et exploite l'environnement de modélisation d'Éclipse (EMF). Il est constitué de plus d'une centaine de plugiciel, définissant une hiérarchie de concepts relatifs à la robotique mobile. Ces concepts couvrent les domaines des capteurs, des actuateurs, des mathématiques, de la navigation, des algorithmes de vision et bien d'autres. C'est dans cet ensemble d'utilitaires fortement intégrés que le système de gestion d'Atlas est développé.

4.3 Environnement expérimental

La plateforme mobile robotisée est destinée à évoluer dans un environnement naturel tel que le terrain d'émulation martienne de l'ASC. Ce dernier est un terrain de 30 mètres par 60 mètres délimité par une bordure de béton. Le terrain est principalement constitué de sable et de roches et tente d'émuler différentes surfaces pouvant être retrouvées sur la planète Mars. On y retrouve entre autres une plaine déserte, une dune, une falaise, une

(a) Cratères, dune et cavernes.

(b) Plaine rocheuse et falaise.

(c) Plaine déserte, dune et falaise.

Figure 4.2 Photos présentant les différentes surfaces du terrain d'émulation martienne de l'Agence spatiale canadienne. Celles-ci furent prisent en 2004, une fois sa construction achevée.

caverne à trois entrées, deux cratères et une plaine rocheuse tel que présenté à la figure 4.2. Notons que les éléments constituants ce terrain sont à l'échelle du robot, de manière à émuler adéquatement le contexte martien.

Afin de supporter le processus de navigation, il existe trois modèles de l'élévation de ce terrain aux résolutions de 1 mètre, 0.5 mètre et 0.2 mètre. Ces modèles ont été pris il y a plus de 4 ans, après la fabrication du terrain et n'ont pas été refaits depuis. Ces modèles ne correspondent donc pas à l'état actuel du terrain qui a subi l'effet d'érosion dû au vent, à la pluie et aux multiples cycles de gel/dégel. Cette différence entre modèle et réalité est voulue puisqu'elle représente un retard temporel significatif que l'on retrouve dans un contexte réel d'exploration planétaire. Cet aspect temporel ainsi que la faible résolution de ces modèles ne leur permettent pas d'être utilisées pour planifier un déplacement sécuritaire de la plateforme mobile robotisée. Ils peuvent cependant représenter une connaissance a priori du terrain provenant d'images orbitales ou de missions précédentes.

4.4 Processus de navigation autonome

Le processus de navigation autonome présenté est décrit en détail dans [Rekleitis et coll., 2008b]. L'objectif de ce processus est d'exécuter un déplacement robotisé vers une destination non visible depuis le point de départ, sans nulle autre intervention humaine que la requête initiale. Nous considérons que le robot a une connaissance a priori générale du terrain et qu'il connaît sa position au moment de la requête de navigation vers la destination à atteindre. La figure 4.3 présente un schéma décrivant les étapes de ce processus.

En premier lieu, le robot utilise la carte globale de l'environnement pour planifier un chemin approximatif qui le guidera de sa position actuelle vers sa destination. Le robot

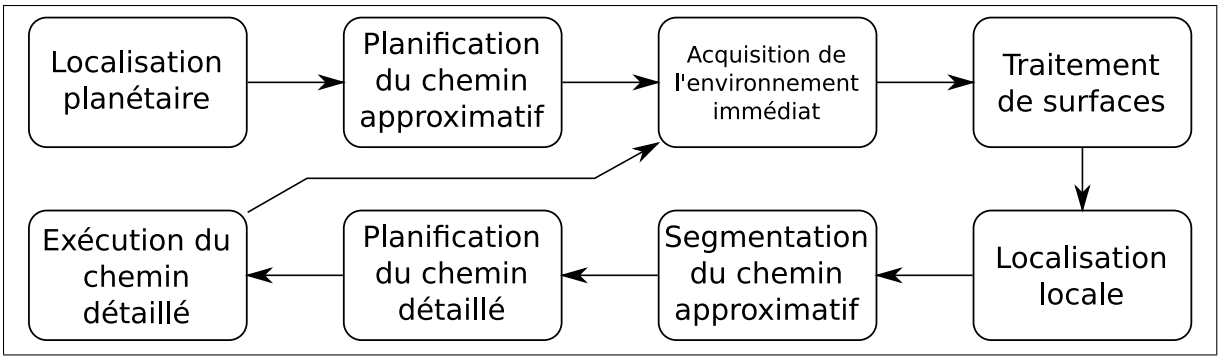
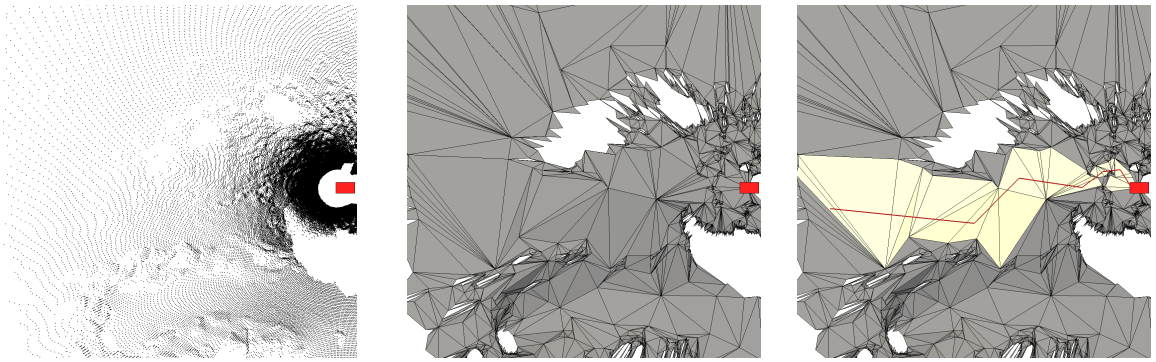


Figure 4.3 Schéma du processus de navigation autonome. Lors des expérimentations, l'étape de *Localisation planétaire* est exécutée par l'opérateur du robot. L'étape de *Localisation locale* est également facultative ; au détriment d'augmenter l'erreur en position sur la destination atteinte.

obtient également une représentation détaillée de son environnement immédiat à l'aide du LIDAR. Puis, le chemin approximatif est segmenté en utilisant la représentation locale détaillée pour déterminer quelle portion de ce chemin est présentement visible. Une planification optimale est alors calculée à même la carte détaillée de l'environnement pour tracer un chemin sécuritaire et faisable pour le robot. Une fois exécuté, le robot obtient une nouvelle représentation détaillée de l'environnement lui permettant de procéder à une nouvelle itération de navigation suivant le chemin approximatif restant, jusqu'à l'atteinte de sa destination. Au stade actuel, l'estimation de la pose du robot provenant de la combinaison de l'IMU et de l'odométrie des roues du P2-AT permet une navigation sécuritaire de la plateforme sur des segments de 10 mètres.

L'utilisation de maillages triangulaires irréguliers (ITM) se retrouve au cœur de ce processus pour la représentation de surfaces. Le capteur LIDAR produit un nuage de points en 3D. Ces points sont alors maillés par triangulation de Delauney dans un espace de coordonnées polaires, puis décimées en combinant plusieurs triangles coplanaires en un seul triangle. La figure 4.4 présente un exemple de ce procédé de traitement des surfaces. Notons la présence de grands triangles définissant les régions planes, ainsi qu'une densité accrue de triangles dans les zones présentant des obstacles. Une explication plus détaillée de l'utilisation des ITMs pour la représentation des surfaces est présentée par les articles [Rekleitis et coll., 2007a,b].

L'utilisation des ITMs pour décrire les surfaces gagne en intérêt au moment de planifier un chemin. En fait, les triangles du maillage représentent des cellules. Lors de la traversée du terrain, le robot se déplace d'une cellule à l'autre en passant par ses côtés communs. La représentation par ITM peut alors être transposée en une structure de graphe où un



(a) Données LIDAR brutes. (b) ITM après décimation. (c) Planification détaillé.

Figure 4.4 Processus de traitement des données visuelles utilisé lors de la navigation autonome. Le robot est représenté par le rectangle rouge, (a) les points noirs représentent les données brutes provenant du capteur LIDAR, (b-c) les triangles gris foncés constituent le maillage triangulaire irrégulier (ITM), (c) les triangles jaune pâle montrent les cellules formant le chemin détaillé et la ligne rouge représente le chemin planifié depuis ces cellules.

noeud représente une cellule et une transition représente un côté commun à deux cellules triangulaires. Le problème de recherche de chemins se transpose alors en un problème de recherche de graphe. L'algorithme de recherche de graphe utilisé est le *Shortest-Path* de Dijkstra [Dijkstra, 1959] dont une implémentation Java est disponible dans la librairie *jgrapht*. L'un des principaux avantages de l'utilisation d'un graphe pour la recherche de chemins est qu'elle ne reste pas prise dans des minimaux locaux ; s'il existe un chemin possible entre deux endroits, la recherche de graphe le trouvera. Qui plus est, pour une fonction de coût donnée, l'algorithme *Shortest-Path* de Dijkstra retourne toujours le chemin ayant le coût le plus faible, donnant ainsi une solution optimale. La fonction de coût utilisée prend en considération plusieurs facteurs tels que la pente et l'irrégularité du terrain se trouvant sous le robot, considéré comme un disque de rayon r. La figure 4.4 présente un exemple de solution pour une recherche de chemins dans un ITM.

CHAPITRE 5

Atlas multi-couches pour la gestion de cartes

5.1 Avant-propos

Auteurs et affiliations :

JL. Bedwani : étudiant à la maîtrise, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

I. Rekleitis : professeur adjoint, Université Mcgill, *Faculty of Science, School of Computer Science.*

E. Dupuis : gestionnaire, Agence spatiale canadienne, Technologies spatiales.

F. Michaud : professeur, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Date de soumission : 26 février 2009

Revue : *International Conference on Field and Service Robots*

Titre français : Atlas multi-couches pour la gestion de cartes

Note : L'article a été modifié de sa forme originale à la demande des membres du jury

Résumé français :

La prochaine génération de véhicules d'exploration planétaire nécessitera une plus grande autonomie de navigation. De tel requis impliquent la gestion de données géo-référencées potentiellement volumineuses et denses représentées sous la forme de cartes. Cet article présente la conception d'un système de gestion de données dans un contexte d'exploration planétaire autonome. Les auteurs présentent également une solution permettant de gérer dynamiquement une variété de données, l'incertitude des relations spatiales entre deux cartes, de procurer un mécanisme de planification de chemins au travers des cartes, ainsi que la corrélation des cartes pour les opérations de localisation. Cet article présente également les résultats expérimentaux sur l'utilisation du système de gestion Atlas par un véhicule d'exploration autonome.

Multi-Layer Atlas System for Map Management

Abstract : *Next generation planetary rovers will require greater autonomous navigation capabilities. Such requirements imply the management of potentially large and rich geo-referenced data sets stored in the form of maps. This paper presents the design of a data management system that can be used in the implementation of autonomous rover navigation schemes for planetary rovers. It also outlines an approach that dynamically manages a variety of data content and the uncertainty of the spatial relationship between two maps, and provides basic path planning through maps, and correlation of maps in localization operations. Experimental results on the usage of our atlas management system by a rover performing autonomous navigation operations are also presented.*

5.2 Introduction

In December 2003 and January 2004, the Mars Exploration Rovers (MERs) "Spirit" and "Opportunity" landed on Mars and conducted, for the first time, semi-autonomous exploration of another planet [Biesiadecki et coll., 2007]. This was a major step in robotics as all previous robotic missions (Lunokhod, Viking, Sojourner) had not implemented any on-board autonomy. The next missions planned to Mars are NASA's Mars Science Laboratory (MSL) in 2011 [Volpe, 2006] and the European Space Agency's (ESA) ExoMars in 2016 [Vago, 2004]. Both MSL and ExoMars have set requirements to travel up to one kilometer per day.

Given the extremely high cost of these missions, it is expected that each will provide a high return of knowledge for the investment. Therefore, great efforts are made by the scientific teams to make efficient use of these robots lifetime, capabilities, and resources. One of the main problems faced by these robotic probes is the poor quality of the communication link with Earth : the round trip communication delays range between 8 and 40 minutes ; the communication windows last approximately one hour and are spaced 12 hours apart. Finally, the bandwidth is very narrow [Pedersen et coll., 2003]. Considering that MER require intense human interaction with operators on Earth before accomplishing an operation, three communication windows are typically required to reach a rock once it has been identified by the scientific team. This is one of the main incentives for increasing the navigation autonomy of planetary robots. Providing planetary exploration robots with the capability to navigate in an unknown or partially known terrain in a fully autonomous

way would provide more efficient operations. While disconnected from Earth, the rover must be capable of traveling autonomously towards one or more locations using high-level commands. During the communication windows it will maximize the bandwidth usage for scientific data.

Autonomous long-range navigation for a rover implies the ability to map the environment, localize itself and plan trajectories to reach a goal. An autonomous planetary robot ends-up storing and handling huge amounts of sensor data, experimental results and other relevant information. This information is typically attached to maps describing the world. As the number of maps increases, simple planning operations become more complex because different maps combinations can be used to plan the path to the goal. Data from different sensors, such as LIDAR range finders, images from monoscopic and stereoscopic cameras, etc., produce different maps. Moreover, data from the same sensor collected at different times and from different locations produce maps of varying resolution and fidelity.

Autonomous planetary exploration robots represent their environment using metric maps. This type of representation provides precise calculations and modeling required for autonomous navigation. These metric maps are expected to evolve over time, as unexplored sections of the world are visited. The introduction of new information into the map can be done by fusing it with the old map. However, any errors in a robot location will corrupt the known world representation with erroneous information. For this reason, it is preferable to keep multiple local maps instead of updating one large map. This approach has the advantage of keeping data with strong spatial relationships together, and loosely connecting data with weaker spatial relationships. With this approach, the uncertainty associated with a map's origin can always be reduced later since it is never fused with the data [Grisetti et coll., 2007].

Unfortunately, this requires the segmentation of the environment representation in multiple local maps, whose number increases as the robot discovers new terrain. An efficient management system of these local maps is therefore required to maintain world consistency.

The objective of our work is to design a data management system that can be used in the development of autonomous navigation schemes for planetary rovers. The presented solution is capable of dynamically managing a large set of maps while considering the problems related to cartographic operations and uses. The issues considered are : the broad variety of data content, uncertainty in the spatial relationship between maps, trajectory planning through multiple maps, and the correlation of maps in localization operations.

Related work on map management systems is presented in Section 5.3. Section 5.4 proposes a structure for the map management system. Experimental results are presented in Section 5.5, and a critique of the implementation and suggestions for future improvements are given in Section 5.6.

5.3 Mapping a World

Often, map management systems appear in the literature under different names : atlases, modeling environments, databases. Sometimes such systems are merged into the localization and mapping process.

Bose et al. [Bosse et coll., 2004] proposed a map management system based on the concept of a conventional atlas for large-scale cyclic environments. Their framework is based on a hybrid, hierarchical metric/topological map system. At a higher level, a graph is used to represent the world, with graph vertices representing locations, and edges representing the transformation between locations. At a lower level, each location is described as a local 2D metric map. Uncertainty can therefore be modeled in each map, and not with respect to a global reference frame. The uncertainty model is based on a Gaussian representation of the error. The proposed atlas framework is used to transform state uncertainty between local maps, create new maps, handle loop closing situations, and to generate and evaluate competing hypothesis of the local map for representing the current system state.

A similar atlas-like system was proposed by Lisien et al. [Lisien et coll., 2005]. At the higher level, a topological representation is traced on the Generalized Voronoi Graph (GVG) of the mapped environment. However, the graph edges are represented at the lower level as a collection of features. The GVG has the advantage to be embedded in the environment, and can be easily extracted from it while exploring. Moreover, the collection of features of an edge-map represents the local area from the perspective of one GVG node, towards a neighboring node. Each edge-map represents discontinuities in the environment as a relative pose and a Gaussian uncertainty. Such a representation has the advantage of being scalable in memory and computation, but is highly sensitive to changes in the environment and is restrained to non-wide, feature-rich, indoor spaces.

Hierarchical arrangements of maps were explored differently in Kelly et al. [Kelly et coll., 2006], where three levels of maps are maintained, providing decreasing periods of data accumulation and increasing levels of details. The lower level maps are centered on the robot, while the higher-level map is a global representation of the world. The three maps used are

a volumetric grid, a local elevation map and a global traversability map. The creation/update of the map contents flows from the highest detailed map towards the global map. Moreover, the global map is composed of multiple channels. Therefore, traversability maps from different sensors and resolutions can be presented on the same map : for example, data from the robot, from a remote unmanned air vehicle (UAV), and from available satellite information can be presented in one map. The hierarchical map arrangement has the advantage of permitting relocalization of the global map channels without affecting the lower-level maps. It also bounds the memory requirement of maintaining a complete 3D map of the world. On the other hand, this system fuses the data during the update process of the maps, which may result in corruption of their content via the permanent introduction of noise.

The SimScape terrain modeling toolkit [Jain et coll., 2006] from the Jet Propulsion Laboratory (JPL) provides a common infrastructure to represent terrain model data from multiple data sources and make them available to simulation applications. This toolkit supports multiple representations of the terrain geometry such as 2.5D digital elevation maps, point clouds, 3D meshes, and 2.5D irregular triangular mesh (ITM). It also provides transformations between different terrain model representation, and generation of composite terrain models from heterogeneous models. Despite the many capabilities of this toolkit, it is oriented toward simulation applications and does not provide any management capabilities of map registration error.

Finally, the System for Unifying Multiresolution Models and Integrating Three-dimensional Terrains (SUMMITT) is a suite of software tools providing terrain modeling funtionality for the MERs operators [Olson et coll., 2007]. This immersive environment handles images of multiple resolutions and scales available from a variety of sources : descent images, surface images from the lander and the rover, as well as orbital images. For a given exploration site, SUMMITT manages the image registration process, the conversion of stereo images to volumetric primitives (voxels), and the storage/merging of the voxels as an octree structure. Once the site is modeled in the octree, SUMMITT is capable of producing polygonal representation of the site (for visualization and collision avoidance) as well as an elevation map. Unfortunately, this system is not dynamic regarding the evolution of its data content. Once registered, the data is merged and therefore, cannot be changed unless the complete database is re-registered and re-built. Finally, the intent for this suite of tools is to be used by human operators and is not integrated to be used by an autonomous robotic system.

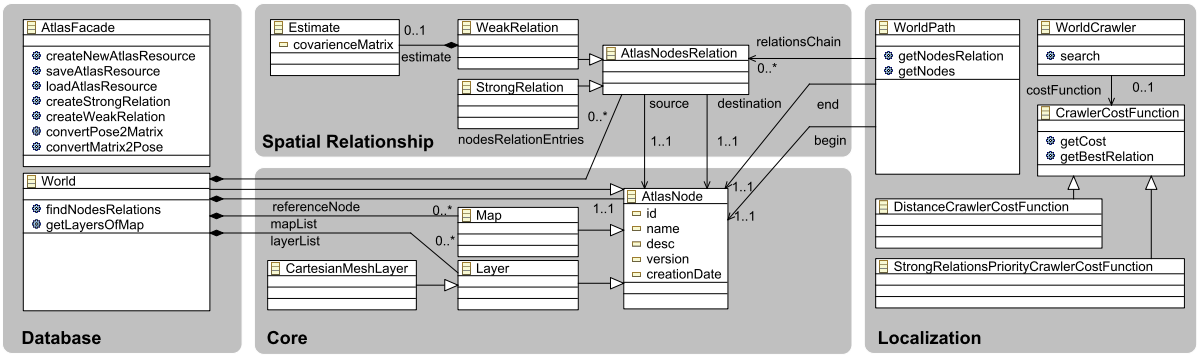


Figure 5.1 UML diagram of our multi-layer atlas management system.

5.4 Atlas Structure

Our atlas framework supports the typical mapping, localization and planning processes performed by a planetary mobile robot. The key idea is to provide a generic infrastructure to manage maps from multiple sources. Changes in the map processing algorithms are transparent to the atlas implementation. In the same way, using maps from different sensor sources or types becomes transparent to the planning algorithm.

The current implementation of the atlas is divided into four key components : the *Core* component defines the atomic elements of the system ; the *Database* component is used to store and access the different data within the system ; the *Spatial Relationship* component incorporates the concept of weak/strong geographic relation between elements of the atlas ; and the requirement to retrieve the location (transformation) of one element with respect to any other elements through a network of relative relations is handled by the *Localization* component. Figure 5.1 presents a UML diagram of the atlas structure.

Planetary rovers can use a broad variety of geo-referenced data : satellite imagery ; images based on monoscopic, stereoscopic, or omni-directional cameras ; ranging data (LIDAR) such as 2.5D point clouds ; or even scientific data such as mineralogical information obtained from spectrometers. All these geo-referenced data sets (LAYERs) represent a specific area of the environment (WORLD) and express different information. Usually, many of these representations are available for a given region (MAP). Supporting a variety of data has multiple implications for a management system : multiple data formats, differences in resolution, precision and uncertainty, the temporal properties of the data set, etc.

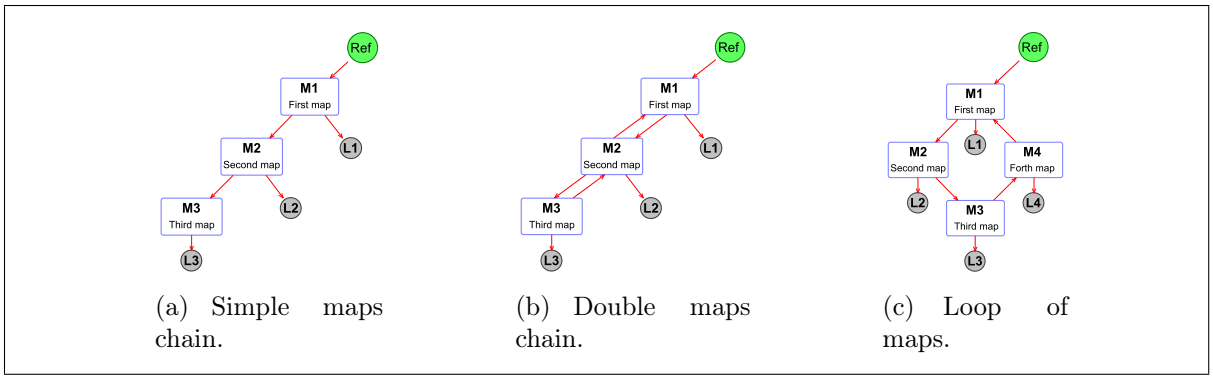
The *Core* component defines the atomic elements of the Atlas system. They represents the different data sets that form the current knowledge of the environment. This com-

ponent is composed of three items : AtlasNode, Map and Layer. A fourth item called World from the *Database* component can also be considered as part of this one. World shares common purpose with Map and Layer, but also acts as a database container. The AtlasNode is the basic element of the atlas system and it is between two of these elements that relative geographical relations can be specified. This element contains key information that defines an AtlasNode, such as its creation time, a name, a description, etc. Three different types of AtlasNode can be used in the atlas : World, Map and Layer. These three types of AtlasNode embed a hierarchy of data association. At the highest level, the World represents the global reference frame to which Maps can be registered to. At the next level, Map items are used to describe a local environment. Finally, a particular environment can be expressed in many ways and this is handled at the third level, using different Layer items.

The *Database* component handles storage and access operations of the atlas system, and it is implemented as two objects called World and AtlasFacade. The AtlasFacade is an utility class that standardizes operations of the management system. The World is the actual database that can access, add or remove different elements of the *Core* and the *Spatial Relationship* components. It also inherits from the AtlasNode element of the *Core* component. This inheritance makes it possible to create relations between any other AtlasNode items with respect to the World origin. Furthermore, the World element also manages the translation of the Atlas content to XML based files, for saving and restoring the database.

A working hypothesis for our atlas system is to consider only data that are spatially referenced. Therefore, a spatial relationship exists between all data in the set. A strong spatial relationship exists for data resulting from a single sensor reading. Such a relationship is affected only by the sensing error. Moreover, when different sensor readings are acquired at the same time and location, their spatial relationship is affected only by alignment error [Mirzaei et coll., 2007]. However, when two data sets are acquired at different times from different locations, their spatial correlation is much weaker because of localization errors.

The *Spatial Relationship* component handles the concept of strong/weak spatial relationship linking the different data of the atlas. It is composed of AtlasNodesRelation, StrongRelation and WeakRelation items. The AtlasNodesRelation embeds a relative transformation required to pass from one AtlasNode reference to another one. StrongRelation represents relations without major errors, like the ones of a sensor



(a) Simple maps chain.

(b) Double maps chain.

(c) Loop of maps.

Figure 5.2 Visualization of different maps configuration topology. The green circle tagged "Ref" represents the WORLD reference element, blue rounded rectangles represent MAP elements, while grayed circles represent LAYER elements. These elements are connected by red arrows corresponding to an ATLASNODESRELATION element.

fixed on the robot body frame. WEAKRELATION represents transformations with greater localization uncertainty, and is represented as a transformation augmented by a covariance matrix.

For example, a given MAP can be linked through strong relations to a number of LAYER items. This same MAP can also be linked to other MAP items using weaker relations. Finally, some MAP items might also be referenced to a global coordinate system, by adding a relation between MAP and the WORLD element.

The *Localization* component of the atlas system provides the functionality of converting the relative relations referenced above into simple transformations. It is used to compute the pose (position and orientation) of any ATLASNODEs with respect to an arbitrary reference frame.

The WORLDCRAWLER item transfers the atlas content in an augmented graph structure that allows it to use cost-based graph search algorithms to find appropriate chains of transformations between every two ATLASNODEs. Each implementation of a CRAWLER-COSTFUNCTION item represents an algorithm that assigns a cost for any segment of the graph. The cost algorithm can use many parameters, such as distance, creation time of nodes, weakness/strength of the relation, as well as a combination of multiple relations between two nodes. The search result of the WORLDCRAWLER item is provided as a WORLDPATH item. It represents a chain of relations between the two ATLASNODEs of the search, and is capable of providing the transformation between any two ATLASNODEs in the chain.

Figure 5.3 The Mars emulation terrain with our modified P2AT rover.

5.5 Experimental Results

The implementation and test of the atlas structure was done using the *Eclipse Modeling Framework* [Steinberg et coll., 2008] (EMF). An atlas database was created using this framework to verify that each element could be added and accessed appropriately. As presented by Fig. 5.2, a simple database viewer was generated using the *Graphical Modeling Framework* [Gronback, 2009] (GMF) in order to view and manipulate such EMF data structures. The complete implementation was deployed on Canada Space Agency's mobility platform shown in Figure 5.3.

In order to validate the usability of the atlas system, a set of use cases were defined : loading an existing atlas, creating a new atlas with a navigation scheme, updating the atlas with visual odometry, etc. These use cases were performed using the 2.5D laser range data and 3D odometry information collected during previous robotic experiments [Rekleitis et coll., 2008a]. The objective of the off-line experiments was to ensure that a large set of geo-referenced data (89 scans) can be easily handled by this system and to verify that the registration of the different data sets are preserved when using relative instead of global registration. The use of relative registration implies additional computational effort in order to obtain a map pose. This effort is therefore quantified in terms of time, while requesting the pose of different maps with respect to other maps or the atlas reference node. Finally, the system was tested on its information retrieval capabilities. A set of random points was generated, and for each random point the atlas was queried to return all the scans that contain data near the selected point. The atlas query consists of retrieving all the data sets origins in the proper frame of reference, sort them by distance to the specified

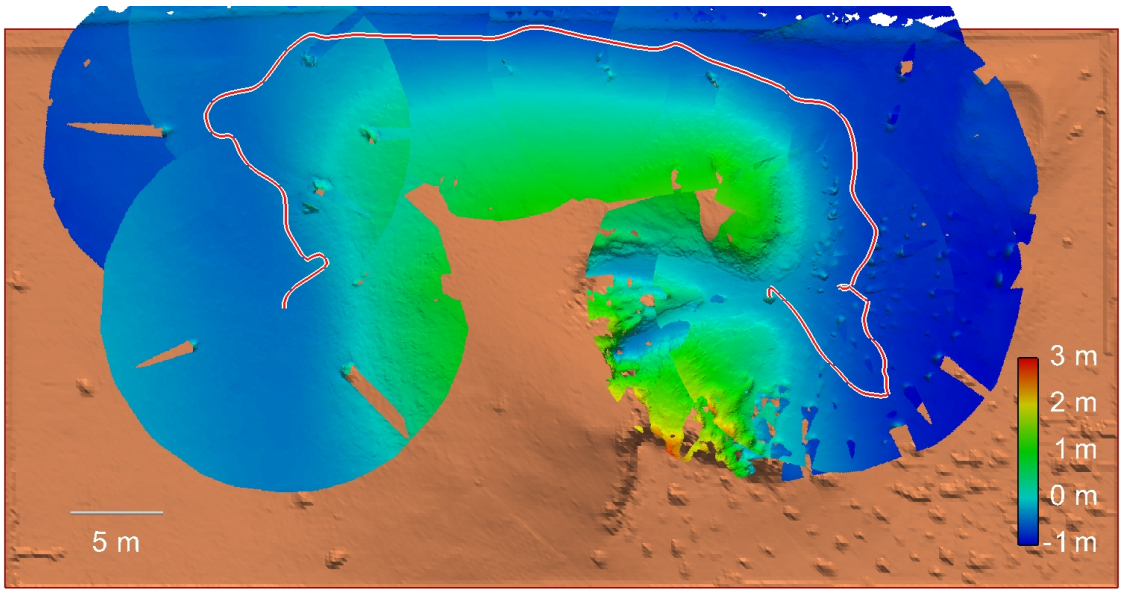
random point, and retrieving the corresponding scans data for maps within a 15 meters range. Over 15 trials the query time was on average 78 sec with a standard deviation of 1.3 sec. However, if only the first match is required, this query returns within few seconds.

Experimental validation of the atlas system was performed on a Pioneer P2AT platform enhanced with an IMU, a 360° fov LIDAR, and a Celeron-M class processing unit (1.5 GHz, 1 GB of RAM). They were run on the Canadian Space Agency's Mars emulation terrain, providing a variety of landscapes on a terrain of 60 meters by 30 meters [Rekleitis et coll., 2008b]. The goal is to perform an autonomous navigation [Rekleitis et coll., 2007a, 2008a] by employing the atlas system to store and register maps (using 3D odometry for the pose estimation) while traversing unknown terrain. When visual matching for two neighbors maps is available (using a variant of the ICP algorithm [Chetverikov et coll., 2002] called Kd-ICP [Pomerleau, 2008]), the atlas updates its content by adding a relation between these two maps, thus creating a double linked chain of maps as shown in Fig. 5.2b.

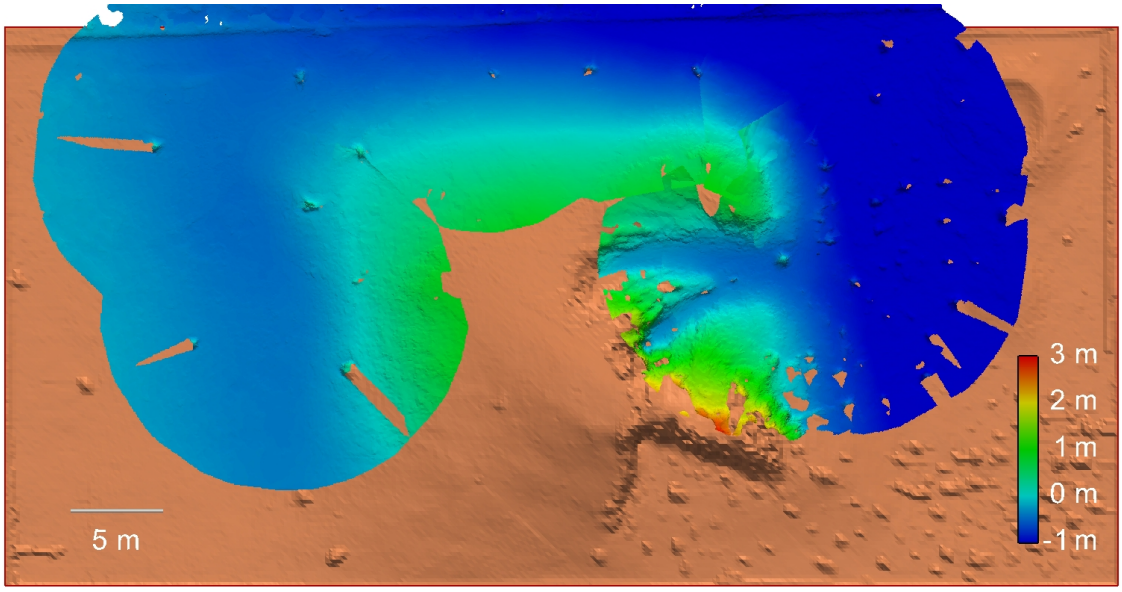
Figure 5.4 presents an illustrative example of the atlas framework in practice. The rover started at a known location and traversed to a final destination beyond its sensing horizon. During the experiment, there was no operator intervention after the selection of the final destination. The atlas was used to store the different scans and to also provide updated spatial relationships when the Kd-ICP algorithm was employed. Figure 5.4a presents the raw data from the nine scans that were acquired during the experiment by using the robot odometry as registration information. As can be seen by the discrepancies in the shades of colors, the scans do not intersect properly on a common surface. Therefore, planning a transition from one scan to the next is a difficult process. Figure 5.4b presents the same nine scan extracted from the atlas system while using the Kd-ICP corrected registration. They were transformed to be in the same coordinate frame using all available information. The different surfaces are well aligned and transitioning from one data set to the next is trivial.

5.6 Conclusion and Future Works

This work demonstrates the feasibility of a data management system suitable for robotic planetary exploration. The main features of our approach are the capability to dynamically manage a variety of data formats, the handling of uncertainty in the spatial relationship between the maps, the capability to provides series of maps linking two locations in planning operations, and the functionality to correlate maps in localization operations.



(a) Initial registration of the consecutives scans and the robot odometry path.



(b) Updated registration using ICP correction.

Figure 5.4 Atlas management system used to present an elevation map composed of nine scans collected during an autonomous navigation experiment. Scans registration is done using robot odometry and corrected using visual correlation with Kd-ICP algorithm.

This management tool opens new opportunities in the development of autonomous navigation schemes. In the first place, maps are never fused by the system, thus permitting registration of the maps at any time. The system makes it possible for two given maps to have multiple mutual spatial relationships such as an estimation of their relative pose from the wheel odometry and visual registration. The atlas system provides the mechanism to select/merge these multiple estimation relations in order to get the best estimation possible. Thanks to the modularity of the system, it is possible to provide different cost functions in order to obtain more accurate use of these multiple relations. Finally, by using relative relations instead of global relations, the atlas system automatically propagates to neighboring maps any improvements/changes deriving from the updated relation between two maps.

Experimental results of the atlas management system showed its capability to provide the required information in a timely fashion. In addition, the robust handling of the uncertainty relations between sub-maps led to more accurate and efficient navigation behaviour.

The atlas system introduced in this paper has several interesting additional features which are scheduled for future implementations. One example is the capability to automatically convert data of one type to another. Another improvement will be to use a binary database instead of a XML one to minimize the usage of the communication link of a planetary rover.

Future robotic experiments using the atlas system are scheduled for human supervised autonomous exploration experiments. The atlas system will provide a common framework to store, use, exchange, and visualize a broad variety of geo-referenced data (thermal, visual, and 2.5D range data) to both human and robotic clients.

Acknowledgment

We would like to thank Régent l'Archevêque and Pierre Allard for their many suggestions during the design and implementation of the atlas management system, Tom Lamarche and Sébastien Gemme for their support in providing a reliable LIDAR system and tools, David Gingras and Alessio Salerno for their support during field testings. Finally, thanks to François Pomerleau who kindly provided us with a reliable, efficient and robust implementation of the Kd-ICP algorithm. F. Michaud holds the Canada Research Chair in Mobile Robotics and Autonomous Intelligent Systems.

CHAPITRE 6

Résultats supplémentaires

L'article présenté au chapitre 5 mentionne l'utilisation de tests unitaires lors de la validation de l'Atlas (voir section 5.5). Considérant l'espace restreint autorisé pour la publication, ceux-ci ont volontairement été retirés, laissant place à un seul exemple intégrant toutes les facettes du système de gestion. Ce chapitre pallie ce manque en présentant ces divers tests. En premier lieu, les modules sont validés à l'aide des tests de base tels que décrits aux sections 6.1 et 6.2, puis l'interaction de ces modules et les fonctionnalités évoluées de l'Atlas sont couvertes par les tests d'intégration décrits à la section 6.3. La capacité d'extension du système est décrite à la section 6.4. La section 6.5 présente une comparaison des caractéristiques de l'Atlas avec celles des systèmes similaires. Pour terminer, une analyse des résultats présentés dans ce chapitre est donnée à la section 6.6.

La majorité des tests de base font usage de données synthétiques définissant des cas types d'utilisation ou d'organisation des données dans le système de gestion. Ces tests vérifient que les opérations de base sont fonctionnelles et réagissent comme prévu. Les tests plus complexes se basent sur des données réelles provenant d'expériences robotisées effectuées lors de la période de juin à novembre 2007 à l'aide de la plateforme robotisée décrite à la section 4.1 et du terrain d'émulation martienne présenté à la section 4.3. Cette campagne d'expériences a permis de tester l'utilisation de l'Atlas hors ligne avec des données en contexte réalistes et de constituer une banque d'images LIDAR en 3D accompagnées des données d'odométrie de la plateforme robotisée. En tout, 38 expériences, dont sept navigation semi-autonome (où les sites visités sont choisis par l'opérateur) et 15 complètement autonomes (où seul le site final est déterminé par l'opérateur). Au total, 108 sites ont été visités pour lesquels des données sont disponibles aux fins de validation du système de gestion.

6.1 Modules *Core*, *Database* et *Spatial Relationship*

6.1.1 Création d'un atlas

Une partie des données expérimentales de 2007 sont utilisées pour créer une base de données d'Atlas sur disque. Celles-ci proviennent des 22 expériences de navigation robotisée

comptant de un à dix sites chacune, pour un total de 97. Ce test n'intègre que l'information de structure (nom, date, description et relations) sans y inclure les données des cartes (ITM). Ce test vérifie que tous les champs de la base de données sont assignables et que le processus d'écriture sur disque fonctionne.

6.1.2 Accès d'un atlas

En conjonction avec le test unitaire précédent, ce test s'assure qu'un atlas sur disque et les données qu'il renferme sont accessibles en lecture. L'opération consiste à lire la base de données générée par le test de création d'un atlas et de s'assurer que son contenu et sa structure correspondent aux données utilisées lors de sa création.

6.2 Module de *Localisation*

6.2.1 Recherche simple

Ce test se base sur un atlas synthétique formé d'un nœud de référence et de deux cartes (donc trois nœuds dans l'Atlas) où seulement les deux cartes sont relativement reliées. Il n'existe donc pas de relations entre le nœud de référence et les cartes. Puisque l'implémentation logicielle du module de localisation transpose la structure d'atlas vers celle d'un graphe, il est important de vérifier son comportement de recherche pour les différents cas d'utilisation. Le test vérifie donc qu'une recherche de graphe depuis le nœud de référence vers les nœuds de cartes donne bien une solution vide (pas de chemins), et que la recherche entre les deux nœuds de cartes est exacte.

6.2.2 Recherche directionelle

Un atlas synthétique formé d'un nœud de référence et de deux cartes (donc trois nœuds dans l'Atlas) disposées en boucle est utilisé. On vérifie que la recherche entre deux nœuds d'une boucle est valide. La réponse idéale de cette recherche correspond au segment le plus court de la boucle. Les cas suivants sont vérifiés : recherche relative (carte à carte) dans les sens direct et inverse de la relation ainsi que la recherche globale de cartes (référence vers carte). Ce test vérifie donc que la séquence de nœuds composant la réponse est exacte.

6.2.3 Recherche séquentielle

Une expérience de navigation réelle comprenant dix sites visités séquentiellement est importée dans le système sans y inclure les données des cartes (ITM). Après interrogation de l'Atlas pour la séquence de nœuds reliant la référence au dernier site visité, le test vérifie que la séquence est respectée et que la pose globale de chaque noeud reste similaire à la valeur entrée (tolérance d'une erreur ne dépassant pas 1×10^{-10} m et 1×10^{-10} degrés). Cette erreur provient de la décomposition de la pose globale des cartes en relations séquentielles des cartes. La restitution de la pose ainsi décomposée se fait par la multiplication des matrices homogènes représentant chacune des relations. L'utilisation de points flottants dans ces matrices incorpore une erreur de numérisation. Ce test vérifie que cette erreur demeure négligeable dans le système.

6.3 Intégration de l'Atlas

6.3.1 Calcul de la pose globale

À l'aide d'un atlas contenant un nœud de référence, 97 nœuds de sites et 97 nœuds de données sous forme de maillages triangulaires irréguliers, ce test vérifie la capacité de l'Atlas à reconstituer la pose globale de tous ses nœuds. Ce test diffère de celui de recherche séquentielle dans le nombre de nœuds vérifiés (soit 195) et dans la complexité de structure d'atlas utilisé (ajout de nœuds de données et de leur relation au nœud de site). Pour chaque nœud de l'Atlas, la pose globale (depuis le nœud de référence vers un nœud de carte ou de données) est calculée et comparée à la valeur utilisée lors de sa création. Notons que la même tolérance d'erreur que le test de recherche séquentielle est utilisée lors de la validation.

6.3.2 Importation d'expériences de navigation

Similaire au test de création d'un atlas, nous créons une base de données d'atlas pour les 22 expériences de navigation, tout en y incluant les données de surface sous la forme de maillage triangulaire irrégulier. Les Atlas ainsi générés sont utilisables par les tests de recherche de cartes et d'ajout de relations via Kd-ICP, comme décrit ci-dessous. Ce test construit, sauvegarde et vérifie que tous les champs de la base de données (incluant les maillages décrivant les surfaces) sont assignés. Le test vérifie également que les processus d'écriture sur disque fonctionnent tant bien pour la structure des Atlas que pour leurs

maillages qui sont préservés dans des fichiers distincts. Ce test permet donc de reproduire hors ligne les opérations faites sur l'Atlas en direct, lors des expériences de navigation.

6.3.3 Recherche de cartes

Utilisant la même base de données que le test de calcul de la pose globale, le test vérifie l'exactitude et la performance de l'opération de recherche de cartes contenant des données de surface à un point donné dans le plan horizontal. Le test est exécuté pour 15 points répartis uniformément dans l'espace expérimental et préalablement vérifiés pour leur présence sur les cartes. La recherche consiste à calculer la pose globale de chaque nœud de données de l'Atlas, à les trier par ordre croissant selon leur distance euclidienne au point recherché et d'extraire l'information de surface pour toute carte à moins de quinze mètres. Ces cartes sont alors vérifiées pour la présence de surface au point recherché. Les résultats de ce test sont présentés dans l'article à la section 5.5.

6.3.4 Ajout de relations via Kd-ICP

Afin de démontrer l'aspect dynamique et la simplicité d'ajout de fonctionnalités à l'Atlas, ce test intègre adéquatement l'algorithme de corrélation de surfaces Kd-ICP [Pomerleau, 2008] permettant d'ajouter une nouvelle relation entre deux cartes adjacentes. Le choix du corrélateur de surface Kd-ICP a été déterminé selon trois critères : son accès à une implémentation logicielle existante, sa possibilité d'intégration avec Java et sa tolérance à l'erreur d'alignement des surfaces. Il est également important de noter que l'on ne cherche pas à quantifier l'efficacité ou l'exactitude du corrélateur. L'on cherche à démontrer la capacité de l'Atlas à évoluer dynamiquement à mesure que de nouvelles informations sont disponibles, ainsi qu'à démonter la simplicité d'extension du système de base par l'ajout d'une nouvelle fonctionnalité.

Le test consiste à trouver chacune des relations de carte à carte existantes dans l'Atlas, y extraire les données de surface et y appliquer le corrélateur. La nouvelle relation provenant du corrélateur est alors ajoutée à l'Atlas sous la forme d'une seconde relation reliant ces deux cartes. L'Atlas ainsi modifié peut être interrogé afin de créer une carte composite (voir figure 5.4) de tout le trajet en tenant compte ou non de la mise à jour apportée par la corrélation. Cette démonstration d'importance révèle la force de ce système de gestion qui ne fusionne pas les données de surface tout en permettant leur évolution sans compromettre leur intégrité.

6.4 Extension du système de gestion de données

Le système de gestion de données proposé est développé en langage Java et s'exécute dans un environnement Éclipse, tel que décrit à la section 4.2. Ce dernier se base sur la technologie de plugiciels ("Plug-ins") facilitant l'extension de logiciels. C'est cette technologie qui permet à l'Atlas d'être étendu sans avoir à modifier son module de base. Les sections 6.4.1, 6.4.2 et 6.4.3 présentent trois types d'extensions de l'Atlas permettant de supporter une multitude de couches de données, différents types de relations spatiales, ainsi que différentes méthodes d'interprétation des relations. Les directives émises dans cette section décrivent la ligne directrice permettant d'étendre les fonctionnalités de base de l'Atlas sans toutefois être un didacticiel des outils d'Éclipse, ni même d'EMF.

6.4.1 Ajout de type de données

La première extension pouvant être ajoutée au gestionnaire de données est celle du support de nouveaux types de données. Pour ce faire, il suffit de créer un nouveau plugiciel qui étend l'atlas de base en lui indiquant comment gérer ce nouveau type de données. Pour illustrer cette fonctionnalité, nous allons étendre l'atlas de base pour supporter une couche de données provenant d'une caméra (ex. : image thermique). La procédure est la suivante :

1. Création d'un plugiciel

   À l'aide d'un environnement Éclipse adéquatement configuré, l'on crée un nouveau plugiciel à l'aide du créateur automatisé de plugiciel que l'on accède par le menu **File→New→Other→Plug-in Project**. L'on entre *ca.gc.space.mrt.atlas.image* en tant que nom du plugiciel et l'on clique sur le bouton "Next". Les paramètres prédéfinis sont utilisés et l'on clique sur le bouton "Finish". Cette opération crée dans l'espace de travail d'Éclipse un nouveau plugiciel nommé *ca.gc.space.mrt.atlas.image* et ses fichiers associés. L'opération ouvre automatiquement le fichier de configuration de ce plugiciel.

2. Configuration des dépendances

   Dans le plugiciel que l'on vient de créer, l'on édite son fichier de configuration qui se retrouve à l'emplacement suivant : *Plugiciel/META-INF/MANIFEST.MF*. Sous l'onglet "Dependencies" l'on ajoute les plugiciels *ca.gc.space.mrt.atlas.core* et *org.eclipse.swt* au titre des "Required Plug-ins". Cette opération permet de spécifier que notre plugiciel dépend de celui de l'atlas comprenant le module "Core" que l'on veut étendre, ainsi que de la librairie SWT ("Standard Widget Toolkit") pro-

curant un support générique pour les images. Sous l'onglet "Runtime" l'on ajoute *ca.gc.space.mrt.atlas.image* au titre des "Exported Packages" spécifiant ainsi le nom des "Packages" qui seront procurés par ce plugiciel lors de son utilisation. Finalement, on sauvegarde le fichier de configuration en sélectionnant le menu **File→Save**.

3. Création d'un modèle EMF

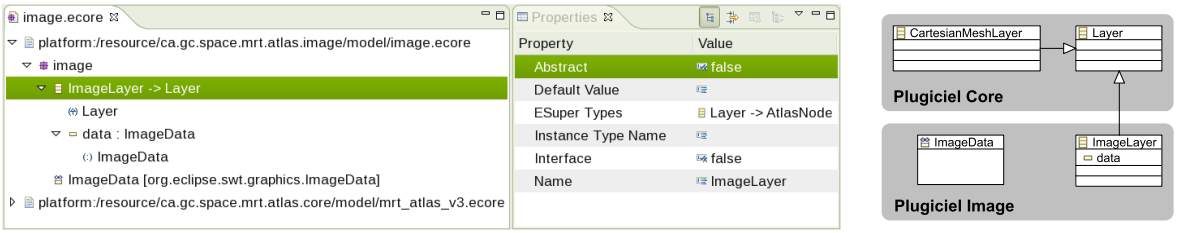
   Il faut maintenant créer une classe dans laquelle on implémente le support pour notre nouveau format de données. Au lieu de programmer notre classe, nous utilisons une technique de méta-programmation. Le principe consiste à définir un modèle de type UML de notre classe qui servira à en générer son code source. Nous définissons le modèle dans un fichier "Ecore" (similaire à UML) à l'aide de l'environnement de modélisation d'Éclipse appelé EMF. Pour ce faire, l'on génère un modèle "Ecore" vide en accédant au menu **File→New→Other→Ecore Model**. L'on spécifie le répertoire *Plugiciel/model/* où le modèle sera créé, ainsi que le nom du fichier modèle : *image.ecore*. Il ne reste plus qu'à cliquer sur le bouton "Next", puis sur le bouton "Finish". Cette opération ouvre automatiquement le fichier de modèle dans l'éditeur. Depuis la fenêtre "Properties" d'Éclipse, nous modifions les champs suivants pour notre modèle :

Name	:	image
NS Prefix	:	ca.gc.space.mrt.atlas.image
NS URI	:	http :///ca/gc/space/mrt/atlas/image/model/image.ecore

   Afin que notre modèle puisse accéder à la définition des éléments du modèle de l'Atlas, nous devons référencer le modèle de ce dernier. Pour ce faire, l'on accède au menu **Sample Ecore Editor→Load Resource...**, et l'on inscrit la ressource suivante : *platform :/resource/ca.gc.space.mrt.atlas.core/model/mrt_ atlas_ v3.ecore*.

4. Définition des éléments du modèle EMF

   Nous définissons maintenant les éléments nécessaires dans notre modèle "Ecore". Nous avons besoin de deux choses : une classe qui implémente le type LAYER de l'Atlas et un "wrapper" EMF pour la classe "ImageData" de la librairie SWT. L'utilisation d'un "wrapper" n'est pas requise si l'on utilise des types de données qui sont définit par un modèle EMF, tel que les éléments composant l'Atlas. Pour créer un "wrapper", nous accédons au menu **Sample Ecore Editor→New Child→EData Type** et nous inscrivons les données suivantes dans la fenêtre "Properties" d'Éclipse :



(a) Modèle Ecore. (b) Diagramme UML.

Figure 6.1 Représentation du plugiciel Image sous la forme de (a) modèle Ecore et de (b) diagramme UML.

Default Value	:	
Instance Type Name	:	org.eclipse.swt.graphics.ImageData
Name	:	ImageData
Serializable	:	true

Pour créer la classe qui implémente le type LAYER, nous sélectionnons le module image dans notre modèle. Par la suite, nous accédons au menu **Sample Ecore Editor→New Child→EClass Type** et inscrivons les données suivantes dans la fenêtre "Properties" d'Éclipse :

Abstract	:	false
Default Value	:	
ESuper Type	:	Layer->AtlasNode
Instance Type Name	:	
Interface	:	false
Name	:	ImageLayer

Nous ajoutons un attribut *data* de type IMAGEDATA à notre classe. À l'aide du menu **Sample Ecore Editor→New Child→EAttribute**, nous créons cet attribut pour lequel nous modifions les valeurs pré-définies dans la fenêtre "Properties" d'Éclipse par les suivantes :

EType	:	ImageData [org.eclipse.swt.graphics.ImageData]
Name	:	data

Sauvegardons notre modèle en accédant au menu **File→Save**. La figure 6.1 montre le contenu du fichier image.ecore tel qu'il devrait être, ainsi que son diagramme UML équivalent.

5. Génération du code source à partir du modèle EMF

   Finalement, nous générons le code source correspondant au modèle. Pour ce faire, nous créons un fichier de configuration pour le générateur de code à l'aide du menu **File→New→Other→EMF Model**. Nous spécifions le répertoire *Plugiciel/model/* où la configuration sera sauvegardée, ainsi que le nom du fichier : *image.genmodel*. Il ne reste plus qu'à cliquer sur le bouton "Next" pour accéder à la prochaine étape permettant de choisir l'importateur de modèles Ecore. Une fois sélectionné, nous poursuivons l'opération en choisissant le bouton "Next". Nous choisissons maintenant d'utiliser le modèle EMF que nous avons créé à l'étape précédente : *platform :/resource/ca.gc.space.mrt.atlas.image/model/image.ecore*, et nous cliquons les boutons "Load" suivis de "Next". À cette étape, nous spécifions que le groupe de classes ("Packages") à générer est *image*. Nous devons également donner la référence aux générateurs de code dont dépend l'atlas, soit :

Data3d	:	ca.gc.space.mrt.geometry.data3d
Ecore	:	org.eclipse.emf.ecore
Emf	:	ca.gc.space.java.emf
Geometrydata	:	ca.gc.space.mrt.geometry.data
Mrt_atlas_v3	:	ca.gc.space.mrt.atlas.world
Processors	:	ca.gc.space.mrt.common.processors

   Un utilisateur peut craindre de ne pas connaître les dépendances. Or il n'en est rien, car Éclipse informe l'usager des références manquantes. L'usager n'a qu'à choisir dans la liste celle qui correspond au message émis par Éclipse. Le fichier de configuration du générateur est créé une fois que l'on sélectionne le bouton "Finish". Cette opération ouvre automatiquement le fichier dans l'éditeur dans lequel on sélectionne l'élément suivant : Image→Image. Depuis la fenêtre "Properties" d'Éclipse, nous modifions le champ suivant pour notre générateur de code :

All→Base Package	:	ca.gc.space.mrt.atlas

   Il ne reste plus qu'à sauvegarder la configuration du générateur en sélectionnant le menu **File→Save**. Nous complétons l'opération en accédant au menu **Generator→**

Generate Model Code, qui aura pour effet de créer le code source dans les répertoires : *Plugiciel/src/ca.gc.space.mrt.atlas.**.

6. Implémentation logicielle du format de données

   La fonctionnalité permettant de supporter le format d'image est déjà implémentée par la librairie SWT et sa classe ImageData. Ce qui n'est pas implanté est sa sauvegarde sur disque et sa lecture depuis le disque. L'Atlas se sert de la capacité d'EMF à sérialiser les modèles "Ecore" pour s'acquitter de cette tâche. Du moment où nous utilisons les types pré-définis par EMF dans un modèle "Ecore", nous sommes assurés que la sérialisation est adéquatement implantée. Par contre, quand un modèle utilise un type externe à l'aide d'un "wrapper", il faut alors s'assurer de définir correctement la sérialisation de ce modèle.

   Le générateur de code que nous avons utilisé a permis de créer une série de classes Java qui composent notre plugiciel. L'une de ces classes se nomme ImageFactoryImpl et sert à différentes tâches dont la création des objets définis par le modèle ainsi que leur sérialisation. Pour supporter adéquatement la conversion du format ImageData vers et depuis une chaine de caractères String, nous n'avons qu'à modifier le code pré-généré. Le tableau 6.1 montre le code des deux fonctions devant être modifié pour permettre une sérialisation adéquate des données d'image. Il est important de noter l'ajout du mot *NOT* à la suite du code annoté Java *@generated* présent dans les entêtes de ces fonctions. Cette modification empêche le générateur de code d'écraser le contenu de ces fonctions lors d'utilisations subséquentes.

Cette procédure démontre clairement la simplicité avec laquelle nous pouvons étendre l'Atlas afin de supporter un nouveau type de donnée. Il suffit de six étapes qui se réalisent en une quinzaine de minutes. Le peu de code requis est celui nécessaire à la sérialisation des éléments externe à EMF, qui dans ce cas-ci représente une douzaine de lignes de code. À ce point-ci, nous disposons de tout ce dont il est nécessaire pour utiliser une couche de données d'images dans un atlas. Il suffira de créer un objet de type ImageLayer, de lui insérer les données de l'image, et d'ajouter l'objet dans l'Atlas. Tout comme nous le faisons pour des données de type CartesianMeshLayer, nous devons ajouter une relation spatiale permettant de relier notre objet de type ImageLayer à un autre nœud de l'Atlas, tel qu'un objet Map, un autre objet Layer, ou le nœud de référence globale.

```
/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated_NOT
 */
public ImageData createImageDataFromString(EDataType eDataType, String initialValue)
{
        // Extract the byte array from the string encoded version.
        byte[] data = Base64.decode(initialValue);

        // Creates an input stream to be read by the ImageLoader.
        ByteArrayInputStream inputStream = new ByteArrayInputStream(data);

        // Loads the ImageData from the input stream.
        ImageLoader loader = new ImageLoader();
        ImageData imageData = loader.load(inputStream)[0];

        return imageData;
}

/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated_NOT
 */
public String convertImageDataToString(EDataType eDataType, Object instanceValue)
{
        ImageData imageData = (ImageData) instanceValue;

        // Creates an ImageLoader that contains the ImageData.
        ImageLoader loader = new ImageLoader();
        loader.data = new ImageData[]{imageData};

        // Writes the content of the ImageLoader to an output stream.
        ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
        loader.save(outputStream, SWT.IMAGE_JPEG);

        // Encode the output stream byte array to a XML compatible string.
        String data = Base64.encode(outputStream.toByteArray());

        return data;
}
```

TABLEAU 6.1 Code source requis pour sérialiser un objet de type IMAGEDATA.

6.4.2 Ajout de type de relations

La procédure permettant d'ajouter le support d'un nouveau type de relations par l'Atlas est similaire à celui présenté à la section 6.4.1. Quelques différences sont à prévoir. La dépendance à la bibliothèque SWT n'est probablement pas nécessaire, mais il se peut qu'une autre bibliothèque le soit. Au niveau du modèle EMF, nous définissons une classe NewTypeRelation au lieu de ImageLayer qui hérite du super-type AtlasNodesRelation au lieu de Layer. Ce modèle n'a pas d'élément data, mais probablement un autre qui sera mieux adapté aux besoins du nouveau type de relations, par exemple *tenseur* ou *variance*. Cet élément peut utiliser un des types pré-définis par EMF tels que le EDouble, le EFloat, le EInt, ou tout autre type pour lequel on crée un "wrapper" similaire à l'exemple précédent de ImageData. L'usager n'a qu'à modifier la classe NewTypeRelationImpl pour y programmer les particularités de son implémentation, tel qu'un calcule mathématique basé sur son composant spécifique (ex. : *tenseur* ou *variance*).

6.4.3 Ajout d'un chercheur de relations

Le module "Localisation" de l'atlas propose un moteur de recherche basé sur le principe d'une recherche de graphe. L'extension de ce module se fait en proposant une nouvelle fonction du calcul de coût associée à la relation reliant deux noeuds. Différentes raisons peuvent pousser un utilisateur à utiliser différentes fonctions de coût. Une fonction peut vouloir favoriser ou exclure un type de relation (ex. : favoriser la localisation visuelle sur celle de l'odométrie, ou n'utiliser que la localisation odométrique de manière à enlever l'effet de tout repositionnement visuel), combiner plusieurs estimations (visuel et odométrique) en une approximation plus précise (ex. : filtre de Kalman, maximisation de tenseur), ou tout simplement optimiser un facteur en particulier tel que la distance, ou l'incertitude des relations. Les possibilités sont nombreuses et dépendent de l'objectif voulu.

En pratique, nous opérons de la mème manière qu'à la section 6.4.1. Les dépendances du plugiciel seront changées pour une dépendance au module "Localisation" qui se nomme *ca.gc.space.mrt.atlas.crawler*. Au niveau du modèle EMF, nous définissons une classe NewCostFunction au lieu de ImageLayer qui hérite du super-type CrawlerCostFunction au lieu de Layer. L'usager devra "overloader" les fonctions *getCost()* et *getBestRelation()* définis par la classe CrawlerCostFuction dans l'implémentation de sa classe NewCostFunctionImpl. Ces deux fonctions permettent respectivement de cal-

culer le coût du lien du graphe entre deux noeuds adjacents, et de retourner la meilleure des relations de l'Atlas reliant deux noeuds adjacents du graphe.

Pour utiliser la nouvelle fonction de coût, il suffit de créer une instance de WorldCrawler et de l'initialiser avec une instance de la classe NewCostFunction. Toutes les recherches effectuées à l'aide de cette instance du moteur de recherche utiliseront la fonction de coût spécifiée. Il est donc possible d'utiliser différentes instances du moteur de recherche pour effectuer des recherches basées sur différents critères (fonctions de coût). Le module "Localisation", tel que représenté à la figure 5.1, possède déjà deux implémentations : DistanceCrawlerCostFunction et StrongRelationsPriorityCrawlerCostFunction. Le tableau 6.2 présente le code source qui est utilisé pour l'implémentation de la classe DistanceCrawlerCostFunctionImpl. Ceci démontre la simplicité (11 lignes de code) avec laquelle il est possible d'étendre le moteur de recherche avec une nouvelle fonction de calcul du coût des relations.

6.5 Comparaison des systèmes de gestion de données

Le système de gestion de données proposé au chapitre 5 présente toutes les caractéristiques nécessaires à son utilisation dans un contexte d'exploration planétaire autonome, tel que défini au chapitre 1. Le tableau 6.3 nous permet de comparer ce système à ceux de la section 2.1. Plus un système contient les caractéristiques correspondant aux besoins en gestion de données, plus il sera adapté à la situation.

Les dix critères de comparaisons utilisés dans le tableau 6.3 sont définis ainsi :

Base de données : Le système présente la fonctionnalité d'une base de données, permettant de sauvegarder et accéder les données expérimentales.

Multi données : Le système est capable de gérer différents types de données expérimentales.

Sans pertes : Les données insérées dans le système ne subissent aucune altération qui soit irréversible ou de pertes/dégradation d'information.

Incertitude : L'incertitude reliée au positionnement et à l'orientation des données géoréférencées est maintenue dans le système.

Dynamique : Le gestionnaire est évolutif dans son fonctionnement. Il permet à son contenu d'être modifié ou augmenté, et adapte les requêtes qui lui sont faites en tenant compte de ces changements.

```
/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated NOT
 */
public double getCost(World world, AtlasNode from, AtlasNode to) {
        // Get the best relation from the atlas
        AtlasNodesRelation relation =
            this.getBestRelation(world.findNodesRelations(from, to));

        // Compute the cost
        return this.getRelationCost(relation);
}

/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated NOT
 */
public AtlasNodesRelation getBestRelation(EList<AtlasNodesRelation> relations) {
        Map<Double, AtlasNodesRelation> distanceToRelation =
            new HashMap<Double, AtlasNodesRelation>();
        double bestCost = Double.MAX_VALUE;

        // Iterate trough all the possible links
        for(AtlasNodesRelation relation : relations)
        {
                double cost = getRelationCost(relation);
                distanceToRelation.put(cost, relation);
                // Get the smaller distance from all of the links
                bestCost = Math.min(bestCost, cost);
        }

        /* Provides back the best relation found */
        return distanceToRelation.get(bestCost);
}

/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated NOT
 */
private double getRelationCost(AtlasNodesRelation relation) {
        Vector3d t = new Vector3d(relation.getTransform().getX(),
                                  relation.getTransform().getY(),
                                  relation.getTransform().getZ());

        // Get the distance of the link
        return t.length();
}
```

TABLEAU 6.2 Code source provenant de la classe DistanceCrawlerCostFunctionImpl et utilisé par le moteur de recherche WorldCrawler.

Système	Base de données	Multi données	Sans pertes	Incertitude	Dynamique	3D	Extension	Multi plateformes	Télé-opérateur	Robot autome
Atlas framework [Bosse et coll., 2004]	•	•	•	•	•					•
Hierarchical atlas [Lisien et coll., 2005]					•					•
PerceptOR [Kelly et coll., 2006]		•			•	•			•	•
SimScape [Jain et coll., 2006]	•	•	•			•	•		•	
SUMMITT [Olson et coll., 2007]	•	•	•			•			•	
Atlas multi-couches (voir chapitre 5)	•	•	•	•	•	•	•	•	•	•

TABLEAU 6.3 Comparaison des différents systèmes de gestion de données. La présence de la caractéristique dans le système est représentée par la marque •.

3D : Le référencement des données se fait en 3D.

Extension : Le système présente une interface ou un mécanisme permettant d'étendre ce dernier de manière à supporter de nouveaux types de données ou de nouvelles fonctionnalités.

Multi plateformes : Il est possible d'utiliser ce système sur différents processeurs ou systèmes d'exploitation sans recourir à une adaptation de son logiciel.

Télé-opérateur : Le système est utilisable par un usager ou un télé-opérateur pour visualiser ou accéder aux données expérimentales.

Robot autonome : Le système est utilisable par un robot pour accéder ou interpréter les données.

En observant le tableau 6.3, nous réalisons que de nombreuses caractéristiques nécessaires au processus de navigation autonome d'un robot mobile planétaire sont manquantes chez la majorité des systèmes existants. Il est également important de noter que ces systèmes sont souvent conçus sans considérer les conceptes d'extension du système et de portabilité sur de multiple plateformes (robotiques et informatiques), les rendants inutilisables sur de nombreux systèmes robotiques.

6.6 Analyse des résultats

Les tests présentés aux sections 6.1, 6.2 et 6.3 ont tous été exécutés avec succès. L'ensemble de ces tests a permis de valider les différents aspects de l'Atlas depuis ses éléments

TABLEAU 6.4 Moyenne ± équart type du temps en millisecondes des différentes opérations de l'Atlas en utilisant différentes résolutions de maillages.

Opération \ Résolution (triangles)	10k (ms)	25k (ms)	50k (ms)
Lecture	306±80	733±146	1430±311
Écriture	304±157	1196±398	3318±1130
Relation	0.35±0.14	0.33±0.12	0.34±0.12
Recherche d'une carte	686±325	1520±730	2810±1280
Recherche toutes les cartes	9880±2600	26200±7020	53000±13400
Recalage par Kd-ICP	6630±4280	22500±17900	65400±56600

de base, en passant par une intégration graduelle de ses composantes, jusqu'à sa démonstration fonctionelle dans son ensemble. Le tableau 6.4 présente la compilation des performances de ces tests donnant un apperçu des capacités de l'Atlas. De plus, la modularité du système devant permettre son expansion sans processus complexes de conception et de programmation a même été démontrée avec succès à la section 6.4. Finalement, le système d'Atlas présente de nombreuses caractérisqiques et fonctions non présentes chez les autres systèmes existants, tel que démontré à la section 6.5.

CHAPITRE 7

Conclusion

Ce mémoire a présenté Atlas, un système de gestion de données géoréférencées adapté au domaine de l'exploration planétaire robotisée. Les caractéristiques de ce système sont : la capacité de gérer dynamiquement une variété de formats de données, la gestion de l'incertitude dans les relations spatiales reliant deux cartes, la capacité de procurer une série de cartes reliant deux lieux lors des opérations de planification de chemins, et la fonctionnalité de recaler des cartes lors des opérations de localisation. Cet outil de gestion ouvre la porte à de nouvelles opportunités dans le développement de scénarios de navigation autonome. En premier lieu, les cartes ne sont jamais fusionnées, permettant un recalage des cartes à tout moment. Le système permet également l'établissement de plusieurs relations spatiales entre deux cartes, telle une estimation de leur pose relative décrite par l'odométrie du robot et par une corrélation visuelle. L'atlas procure les mécanismes de sélection et de combinaison de ces multiples relations dans le but d'en retirer la meilleure estimation possible. Également, grâce à la modularité du système, il est possible de définir différentes fonctions de coût afin d'améliorer l'utilisation de ces relations multiples. Finalement, en utilisant des liens relatifs au lieu d'absolus, l'atlas propage automatiquement vers les cartes voisines tout changement/amélioration découlant de l'établissement d'une relation entre deux cartes.

Les résultats expérimentaux du système de gestion Atlas démontrent sa capacité de procurer les informations requises en un temps acceptable. De plus, la gestion adéquate de l'incertitude reliant les cartes tend à améliorer le comportement de navigation.

Le système Atlas est présentement utilisé par l'Agence spatiale canadienne afin de gérer les données géo-référencées de leur plateforme mobile robotisée. Cette dernière participera à l'expérience "Avatar Explore" lors de l'expédition 20 de la station spatiale internationale prévue pour la période de mai à octobre 2009. Cette expérience permettra d'étudier l'interaction homme-machine en utilisant différents scénarios d'exploration planétaire robotisée qui sera supervisée depuis l'espace par l'astronaute Robert Thirsk. Ce système procurera un environnement commun au robot et aux opérateurs afin d'emmagasiner, d'utiliser, d'échanger et de visualiser une grande variété de données (images thermiques, visuelles, nuages de points).

BIBLIOGRAPHIE

Bakambu, J.N., Gemme, S., Allard, P., Lamarche, T., Rekleitis, I. (2006) *3D reconstruction of environments for planetary rover autonomous exploration*, 44th AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics, Reno, Nevada, p. (accepted).

Barfoot, T., Se, S., Jasiobedzki, P. (2006) *Vision-based localization and terrain modeling for planetary rovers*, chapitre 4, Intelligence for Space Robotics, TSI Press, San Antonio, Texas, USA, p. 71–92.

Besl, P.J., McKay, N.D. (1992) *A method for registration of 3-D shapes*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, n° 2, p. 239–256.

Biesiadecki, J.J., Leger, P.C., Maimone, M.W. (2007) *Tradeoffs between directed and autonomous driving on the mars exploration rovers*, The International Journal of Robotics Research, vol. 26, n° 1, p. 91–104.

Bosse, M.C., Newman, P.M., Leonard, J.J., Teller, S. (2004) *SLAM in large-scale cyclic environments using the atlas framework*, International Journal of Robotics Research, vol. 23, n° 12, p. 1113–1139.

Brenneke, C., Wulf, O., Wagner, B. (2003) *Using 3D laser range data for SLAM in outdoor environments*, Intelligent Robots and Systems, vol. 1, n° 1, p. 188–193.

Campbell, R.J., Flynn, P.J. (2001) *A survey of free-form object representation and recognition techniques*, Computer Vision and Image Understranding, vol. 81, n° 2, p. 166–210.

Carrier, W.D. (1992) *Soviet rover systems*, Space Programs and Technologies Conference, American Institute of Aeronautics and Astronautics, p. 9.

Chetverikov, D., Svirko, D., Stepanov, D., Krsek, P. (2002) *The trimmed Iterative Closest Point algorithm*, Proceedings of the International Conference on Pattern Recognition, vol. 3, p. 545–548.

Davison, A.J., Kita, N. (2001) *3D simultaneous localisation and map-building using active vision for a robot moving on undulating terrain*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, IEEE Computer Society Press, Kauai, p. 384–391.

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

Durrant-Whyte, H., Bailey, T. (juin 2006) *Simultaneous localization and mapping : part I*, IEEE Robotics & Automation Magazine, vol. 13, n° 2, p. 99–110.

Ferguson, D., Stentz, A.T. (2005) *Field D* : An interpolation-based path planner and replanner*, Proceedings of the International Symposium on Robotics Research, Springer, p. 1926–1931.

Filliat, D., Meyer, J.A. (2003) *Map-based navigation in mobile robots : I. A review of localization strategies*, Cognitive Systems Research, vol. 4, n° 4, p. 243–282.

Gonzalez-Barbosa, J.J., Lacroix, S. (2005) *Fast dense panoramic stereovision*, Proceedings of the IEEE International Conference on Robotics and Automation, Barcelone, Espagne, p. 1210–1215.

Gowdy, J.W., Stentz, A., Hebert, M. (1991) *Hierarchical terrain representations for off-road navigation*, W.H. Chun, W.J. Wolfe (éditeurs), Proceedings of SPIE, vol. 1388, p. 131–140.

Grisetti, G., Grzonka, S., Stachniss, C., Pfaff, P., Burgard, W. (2007) *Efficient estimation of accurate maximum likelihood maps in 3D*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, p. 3472–3478.

Gronback, R.C. (2009) *Eclipse Modeling Project : A Domain-Specific Language (DSL) Toolkit*, Addison-Wesley Professional, 736 p.

Hähnel, D., Burgard, W., Thrun, S. (2003) *Learning compact 3D models of indoor and outdoor environments with a mobile robot*, Robotics and Autonomous Systems, vol. 44, n° 1, p. 15–27, 2003/7/31.

Hart, P.E., Nilsson, N.J., Raphael, B. (1968) *A formal basis for the heuristic determination of minimum cost paths*, IEEE Transactions on Systems, Science, and Cybernetics, vol. SSC-4, n° 2, p. 100–107.

Jackins, C.L., Tanimoto, S.L. (1980) *Oct-trees and their use in representing three-dimensional objects*, Computer Graphics and Image Processing, vol. 14, n° 3, p. 249–270.

Jain, A., Cameron, J., Lim, C., Guineau, J. (juillet 2006) *SimScape terrain modeling toolkit*, Proceedings of the Second IEEE International Conference on Space Mission Challenges for Information Technology, p. 8.

Johnson, A. (1997) *Spin-Images : A representation for 3-D surface matching*, Thèse de doctorat, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Johnson, A.E., Hebert, M. (1999) *Using spin images for efficient object recognition in cluttered 3D scenes*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, n° 4, p. 433–449.

Jung, I.K., Lacroix, S. (2003) *High resolution terrain mapping using low altitude aerial stereo imagery*, Proceedings of the 9th IEEE International Conference on Computer Vision, Nice (France) RAPPORT-LAAS No 03272, p. 946–951.

Kaufman, A., Cohen, D., Yagel, R. (1993) *Volume graphics*, Computer, vol. 26, n° 7, p. 51–64.

Kelly, A., Stentz, A., Amidi, O., Bode, M., Bradley, D., Diaz-Calderon, A., Happold, M., Herman, H., Mandelbaum, R., Pilarski, T., Rander, P., Thayer, S., Vallidis, N., Warner, R. (2006) *Toward reliable off road autonomous vehicles operating in challenging environments*, The International Journal of Robotics Research, vol. 25, n° 5-6, p. 449–483.

Klein, H.P., Lederberg, J., Rich, A., Oyama, V.I., Levin, G.V. (1976) *The Viking mission search for life on Mars*, Nature, vol. 262, p. 24–27.

Lacroix, S. (2006) *Navigation autonome en environnements extérieurs*, Thèse de doctorat, Institut National Polytechnique, Toulouse (France) Habilitation RAPPORT-LAAS No 06533.

Lacroix, S., Jung, I.K., Mallet, A. (2001) *Digital elevation map building with low altitude stereo imagery*, Proceedings of the 9th International Symposium on Intelligent Robotic Systems, Toulouse (France) RAPPORT-LAAS No 01100, p. 207–216.

Lamon, P., Siegwart, R. (2004) *Inertial and 3D-odometry fusion in rough terrain towards real 3D navigation*, Intelligent Robots and Systems, vol. 2, n° 2, p. 1716–1721.

Lemaire, T., Lacroix, S., Sola, J. (2005) *A practical 3D bearing-only SLAM algorithm*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton (Canada) RAPPORT-LAAS No 05324, p. 2757–2762.

Lisien, B., Morales, D., Silver, D., Kantor, G., Rekleitis, I.M., Choset, H. (2005) *The hierarchical atlas*, IEEE Transactions on Robotics, vol. 21, n° 3, p. 473–481.

Maimone, M., Biesiadecki, J., Tunstel, E., Cheng, Y., Leger, C. (2006) *Surface navigation and mobility intelligence on the Mars exploration rovers*, chapitre 3, Intelligence for Space Robotics, TSI Press, San Antonio, Texas, USA, p. 45–69.

Matijevic, J., Shirley, D. (1997) *The mission and operation of the Mars Pathfinder microrover*, Control Engineering Practice, vol. 5, n° 6, p. 827–835.

Meyer, J.A., Filliat, D. (2003) *Map-based navigation in mobile robots : II. A review of map-learning and path-planning strategies*, Cognitive Systems Research, vol. 4, n° 4, p. 283–317, 2003/12.

Mirzaei, F., Mourikis, A., Roumeliotis, S. (2007) *On the performance of multi-robot target tracking*, Proceedings of the IEEE International Conference on Robotics and Automation, p. 3482–3489.

Noborio, H., Fukuda, S., Arimoto, S. (1988) *Construction of the octree approximating a three-dimensional object by using multiple views*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 10, n° 6, p. 769–782.

Olson, C.F., Matthies, L.H., Wright, J.R., Li, R., Di, K. (2007) *Visual terrain mapping for Mars exploration*, Computer Vision and Image Understanding, vol. 105, n° 1, p. 73–85.

Pedersen, L., Sargent, R., Bualat, M., Kunz, C., Lee, S., Wright, A. (2003) *Single-cycle instrument deployment for Mars rovers*, Proceedings of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space, Nara, Japan.

Pfaff, P., Triebel, R., Burgard, W. (2007) *An efficient extension to elevation maps for outdoor terrain mapping and loop closing*, The International Journal of Robotics Research, vol. 26, n° 2, p. 217–230.

Planitz, B.M., Maeder, A.J., Williams, J.A. (2005) *The correspondence framework for 3D surface matching algorithms*, Computer Vision and Image Understranding, vol. 97, p. 347–384.

Pomerleau, F. (2008) *Registration algorithm optimized for simultaneous localization and mapping*, Mémoire de maîtrise, Sherbrooke University.

Rekleitis, I., Bedwani, J.L., Dupuis, E. (2007a) *Over-the-horizon, autonomous navigation for planetary exploration*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, p. 2248–2255.

Rekleitis, I., Bedwani, J.L., Dupuis, E. (2008a) *Experimental results for over-the-horizon planetary exploration using a lidar sensor*, Proceedings of the 11th International Symposium on Experimental Robotics, Athens, Greece, p. 65–77.

Rekleitis, I., Bedwani, J.L., Gemme, S., Lamarche, T., Dupuis, E. (2007b) *Terrain modelling for planetary exploration*, Proceedings of the Fourth Canadian Conference on Computer and Robot Vision, p. 243–249.

Rekleitis, I., Bedwani, J.L., Dupuis, E., Allard, P. (2008b) *Path planning for planetary exploration*, Proceedings of the Canadian Conference on Computer and Robot Vision, p. 61–68.

Remondino, F., El-Hakim, S.F. (2006) *Image-based 3D modelling : A review*, The Photogrammetric Record Journal, vol. 21, n° 115, p. 269–291.

Rusinkiewicz, S., Levoy, M. (2001) *Efficient variants of the ICP algorithm*, Proceedings of the Third International Conference on 3D Digital Imaging and Modeling, IEEE Computer Society, Los Alamitos, CA, USA, p. 145–152.

Sim, R., Little, J.J. (2006) *Autonomous vision-based exploration and mapping using hybrid maps and Rao-Blackwellised particle filters*, Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems, Beijing, p. 2082–2089.

Slack, M.G. (1993) *Navigation templates : Mediating qualitative guidance and quantitative control in mobile robots*, IEEE Transactions on Systems, Man and Cybernetics, vol. 23, n° 2, p. 452–466.

Steinberg, D., Budinsky, F., Paternostro, M., Merks, E. (2008) *EMF : Eclipse Modeling Framework, Second Edition*, Addison-Wesley Professional, 744 p.

Stentz, A. (1994) *Optimal and efficient path planning for partially-known environments*, Proceedings of the IEEE International Conference on Robotics and Automation, p. 3310–3317.

Sun, Y., Paik, J., Koschan, A., Page, D.L., Abidi, M.A. (2003) *Point fingerprint : A new 3-D object representation scheme*, IEEE Transactions on Systems, Man and Cybernetics, Part B, vol. 33, n° 4, p. 712–717.

Thrun, S., Hähnel, D., Ferguson, D., Montemerlo, M., Triebel, R., Burgard, W., Baker, C., Omohundro, Z., Thayer, S., Whittaker, W. (2003) *A system for volumetric robotic mapping of abandoned mines*, Proceedings of the IEEE International Conference on Robotics and Automation, vol. 3, p. 4270–4275.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., Mahoney, P. (2006) *Winning the DARPA Grand Challenge*, Journal of Field Robotics, vol. 23, n° 9, p. 661–692.

Triebel, R., Pfaff, P., Burgard, W. (2006) *Multi-level surface maps for outdoor terrain mapping and loop closing.*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Bejing, China, p. 2276–2282.

Vago, J. (2004) *Overview of ExoMars mission preparation*, Proceedings of the 8th ESA Workshop on Advanced Space Technologies for Robotics and Automation, Noordwijk, The Netherlands.

Volpe, R. (2006) *Rover functional autonomy development for the Mars mobile science laboratory*, Proceedings of the IEEE Aerospace Conference, vol. 2, Big Sky, MT, USA, p. 643–652.

Zhang, D. (1999) *Harmonic shape images : A 3D free–form surface representation and its applications in surface matching*, Thèse de doctorat, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.