Fusion de données multicapteurs pour la construction incrémentale du modèle tridimensionnel texturé d'un environnement intérieur par un robot mobile

Ayman Zureiki

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *l'Université Toulouse III - Paul Sabatier*
Discipline ou spécialité : *Systèmes Automatiques*

Présentée et soutenue par *Ayman ZUREIKI*
Le *16 septembre 2008*

Titre : *Fusion de Données Multi-Capteurs pour la Construction Incrémentale du Modèle Tridimensionnel Texturé d'un Environnement Intérieur par un Robot Mobile*

JURY

Président :	*M. Patrice DALLE*
Rapporteurs :	*M. Etienne COLLE*
	M. Fawzi NASHASHIBI
Examinateur :	*M. David FILLIAT*
Co-Directeurs :	*M. Raja CHATILA*
	M. Michel DEVY

Ecole doctorale : *Systèmes (EDSYS)*
Unité de recherche : *Laboratoire d'Analyse et d'Architecture des Systèmes - LAAS-CNRS*
Directeur(s) de Thèse : *Raja CHATILA et Michel DEVY*

Remerciements

Les travaux présentés dans ce manuscrit ont été effectués au sein du Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) du Centre National de la Recherche Scientifique (CNRS) à Toulouse. Je tiens pour cela à remercier les directeurs successifs du LAAS, M. Malik Ghallab et M. Raja Chatila, pour m'avoir accueilli au sein de leur laboratoire.

Tout d'abord, j'adresse mes plus chaleureux remerciements à mes deux co-directeurs de thèse, M. Raja Chatila et M. Michel Devy, pour m'avoir proposé ce sujet passionnant, pour m'avoir fait confiance sur certaines pistes techniques et pour avoir orienté mon travail à travers leurs remarques très pertinentes, généralement sources de longs développements ultérieurs. En particulier, je remercie Raja de m'avoir initialement accueilli au groupe Robotique et Intelligence Artificielle (RIA) (aujourd'hui le pôle RIA), d'avoir suivi avec enthousiasme mon doctorat et pour les multiples éclairages qu'il m'a apportés dans le cadre de ma thèse. Et je remercie également Michel, responsable du groupe de recherche Robotique Action et Perception (RAP), d'avoir guidé avec un grand sérieux mon cheminement de doctorant, pour l'attention qu'il m'a toujours accordée, et pour m'avoir offert de nombreuses occasions d'approfondir mes connaissances en robotique.

Lire et juger une thèse n'est pas une tâche aisée à accomplir. Aussi, je tiens tout particulièrement à exprimer ma reconnaissance à tous les membres du jury de ma thèse.

Je tiens à exprimer toute ma gratitude à M. Etienne COLLE, professeur à l'Université d'Evry Val d'Essonne, et Directeur-adjoint du Laboratoire Informatique, Biologie Intégrative et Systèmes Complexes (IBISC), et à M. Fawzi NASHASHIBI, Ingénieur de recherche (HDR) à l'École des Mines de Paris, pour m'avoir fait l'honneur d'accepter d'être rapporteurs et dont les remarques positives et constructives m'ont encouragé à terminer ce travail plus sereinement.

Je remercie sincèrement M. Patrice DALLE, professeur à l'Université Paul Sabatier (Toulouse III), pour avoir présidé ce jury, et M. David FILLIAT maître de conférences à l'École Nationale Supérieure de Techniques Avancées, qui ont bien voulu, malgré un emploi du temps chargé, être examinateurs.

Il m'est très difficile de remercier toutes les personnes qui ont contribué de près ou de loin à ce travail. Je veux néanmoins exprimer toute ma gratitude à Aurélie Claudic et à Noureddine OUADAH pour le temps qu'ils ont passé pour chercher les coquilles et à améliorer la qualité linguistique de ce manuscrit.

J'adresse également dans une plus large mesure mes remerciements à toutes les personnes, collègues permanents, post-doc, doctorants, ingénieurs, stagiaires de l'ex-groupe Robotique et Intelligence Artificielle (RIA), ou aujourd'hui des groupes RAP, RIS et GEPETTO, dont l'ambiance a donné une autre dimension à ce travail. Un merci tout particulier à Sara et Matthieu pour leurs conseils éclairés sur l'utilisation de GenoM et leur support technique concernant l'utilisation des robots. Et je tiens à remercier Maxime et Thomas pour l'aide et le support qu'ils m'ont accordé afin de démarrer à l'utilisation de Jafar. Petite dédicace aux collègues du bureau B69-3, ayant dû me supporter tout au long de cette thèse. Un petit coucou aux anciens, certains déjà partis pour de nouveaux horizons : Nicolas, David, Gauthier ... Je n'oublie pas ceux qui n'ont encore fini : Vincent, Mario, Dora-Luz, Noureddine, Back Van ... à qui je souhaite bon courage et bonne continuation. Je tiens aussi à remercier Jido, notre adorable robot, à qui cette thèse doit vraiment beaucoup. Mes remerciements vont enfin aux enseignants de l'Université Paul Sabatier, ainsi qu'à toutes les personnes qui m'ont apporté leur aide. Merci à tous pour votre accueil et soutien.

Merci également à tous mes amis qui malgré la distance ou leurs propres soucis ont toujours

pensé à moi, merci pour être venu en nombre, afin d'assister à la soutenance et/ou de participer à la préparation du pot de thèse.

Enfin, c'est aussi l'occasion d'adresser mes plus affectueux remerciements à mes parents et à toute ma famille pour leur précieux soutien et leurs encouragements et ce depuis ... une petite trentaine d'années maintenant. Merci à mes parents qui attendent depuis plusieurs années mon retour sans cesser de prier pour moi, et sans qui rien de tout cela n'aurait été possible. Merci aussi à tous les membres de ma belle-famille pour leur soutien rassurant. Finalement, je n'oublierai pas ma petite famille, mes enfants Alaa et Julie, et ma compagne Tharaa, pour m'avoir supporté pendant ces années chargées de travail. Faire une thèse, en ayant deux enfants devient un véritable défi qu'on ne peut pas relever sans avoir à son coté une merveilleuse et adorable âme. À toi cette thèse doit énormément. Merci encore Tharaa.

à maman et papa : source de tendresse inépuisable, l'origine
à mes frères et sœurs : soutien à vie, le présent
à mes enfants : espoir de la vie, l'avenir
à toi, mon âme sœur : hier, aujourd'hui et demain, l'amour

Ayman

إلى أمي وأبي
إلى أخوتي وأخواتي
إلى ولديّ علاء وجولي
إليك رفيقة دربي ثراء
إليكم جميعاً أهدي عملي المتواضع هذا

أيمن

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Chapitre 1

Introduction

1.1 Le Robot Mobile

Durant ces dernières années, le nombre de robots en activité est en pleine explosion, sous les effets combinés des progrès techniques, de la baisse des coûts et de l'émergence de nouveaux besoins socio-économiques (sécurité, défense, aide aux personnes âgées, transport, etc.). Des robots domestiques (comme par exemple les aspirateurs automatiques et les tondeuses automatiques) ont déjà envahi nos habitations et jardins. De plus, les robots de loisir sont devenus de plus en plus demandés. La croissance du nombre de robots dans notre environnement est remarquable. Ces petits engins pourraient en effet intégrer très vite notre quotidien grâce à la multitude d'applications qu'ils pourraient accomplir. Doté de nombreux capteurs, capable de se déplacer en évitant les obstacles et d'aller se recharger en énergie sans intervention humaine, le robot mobile présente surtout l'avantage d'être une plate-forme ouverte à toute technologie. Robot ménager, gardien d'appartement, aide aux personnes handicapées, explorateur des planètes lointaines, support de recherches pour les laboratoires ou simple jouet, tous les espoirs sont donc permis avec ce petit bijou de la robotique.

Les travaux de recherche se poursuivent afin de rendre ces robots de plus en plus autonomes et intelligents. Un robot autonome ne devra plus seulement se limiter au triptyque classique : percevoir, raisonner et agir. Il aura à intégrer des capacités fonctionnelles et décisionnelles adaptées pour l'apprentissage de lieux, d'objets, d'actions ..., pour l'interprétation de situations, pour l'intervention au sein de son environnement, pour la communication avec cet environnement devenu "intelligent", pour la coordination avec ses congénères robots et avec les humains partageant l'environnement, sans oublier la mise en oeuvre de mécanismes toujours plus évolués d'interaction avec l'homme (son maître, son tuteur.. !).

1.2 Problématique Considérée

Le secret de la réussite d'un robot dans son fonctionnement réside dans son pouvoir à maintenir une relation étroite entre toutes les fonctionnalités qu'il intègre, et en particulier entre les trois composantes de base : la perception, la décision et l'action. Cette relation est rendue possible par les modèles que le robot apprend et met à jour tout au long de ses missions. Le robot ne doit pas seulement posséder un modèle de lui-même (capteurs, actionneurs), un modèle de son environnement (espaces libres et traversables, obstacles, etc.), et un modèle des actions qu'il peut exécuter (déplacements, etc.), mais il doit aussi préserver la cohérence entre tous ces modèles.

Un modèle de l'environnement est indispensable pour un robot mobile. La plupart des robots actuels "opérationnels" utilisent des capteurs 2D, en particulier des télémètres laser à balayage horizontal tels que les capteurs SICK : les données acquises sur un plan lui permettent de construire des cartes, de s'auto-localiser, et d'éviter les collisions. Ceci est justifié par le fait que la majorité des applications ont été développées sur des robots terrestres ou d'intérieur qui se déplacent dans un monde à deux dimensions, avec une forte hypothèse de sol plat. Bien que ces robots aient plus au moins donné satisfaction pour les applications visées, les nouvelles demandes sont plus exigeantes. En effet, pour ces nouvelles applications, comme le compagnon domestique, le robot devra affronter des situations pour lesquelles le modèle 2D de l'environnement n'est plus suffisant. À ce titre, citons le cas d'un robot pour l'aide aux personnes âgées ou handicapées. Un tel robot doit évoluer dans un espace tridimensionnel, et par exemple, il pourrait chercher un verre d'eau posé sur une table dans la salle de séjour. Une telle définition de la tâche oblige la connaissance du monde en 3D.

Dans ce mémoire, nous nous concentrons plus précisement sur le problème de la Modélisation tridimensionnelle d'un environnement intérieur, dans lequel un robot doit se déplacer. La construction d'un modèle 3D cohérent et interprétable est une fonction indispensable pour que le robot mobile exécute des tâches en mode autonome. Le choix de la représentation est essentiel : un modèle géométrique 2D est suffisant pour la navigation sur sol plat, avec des obstacles assimilables à des prismes droits. Mais un robot ne fait pas que se déplacer : la complexité et la richesse du modèle 3D posent plus de contraintes lors de la construction, mais offrent plus d'informations et de lisibilité, pour que notre robot exécute des tâches plus complexes (manipulation, action sur l'environnement, évitement d'obstacles mobiles et déformables, etc.).

La modélisation nécessite la mise en oeuvre d'un ensemble de trois fonctionnalités complémentaires :

- La perception 3D : les capteurs 3D embarqués sur notre robot mobile, permettent d'acquérir des données 3D sur la scène perçue depuis la position courante du robot.

- La construction d'une représentation 3D de l'environnement à partir de ces données 3D acquises dans une phase d'exploration : il faut exploiter une méthode capable de cartographier l'environnement et de localiser le robot pour maintenir la cohérence du modèle

- La gestion de cette représentation 3D au cours du temps, sachant que l'environnement peut être dynamique (en particulier présence d'humains et/ou d'autres robots) et évolutif (des objets peuvent être déplacés : chaises, objets posés sur les tables, etc.).

Dans notre travail, nous avons développé et évalué sur des démonstrateurs robotiques, les fonctions de perception 3D et de modélisation 3D. Notons que ces fonctions sont fortement liées entre elles : le choix du capteur influence la qualité des modèles obtenus, et vice versa.

La fonction **Localisation** intégrée sur tout robot mobile tente de trouver une réponse à la question de base "où suis-je ?". Cette fonction exploite une carte apprise ou donnée au préalable au robot : la localisation est relative à cette carte. Par ailleurs, la fonction **Cartographie** permet d'apprendre cette carte : elle est réalisée en utilisant des mesures relatives aux positions du robot mobile. Ainsi, pour pouvoir cartographier un environnement, il faut que les positions successives du robot soient bien connues. En conséquence, il faut gérer cette dépendance mutuelle entre la localisation et la modélisation de l'environnement. Ce problème de l'oeuf et de la poule (figure 1.1) est bien connu des roboticiens sous le nom de **Localisation et Cartographie simultanées** ou *Simultaneous Localization and Mapping* (**SLAM**) ; elle a aussi été nommée *Concurrent Mapping and Localization*. Dans la suite de ce mémoire, nous utiliserons principalement le terme de SLAM.

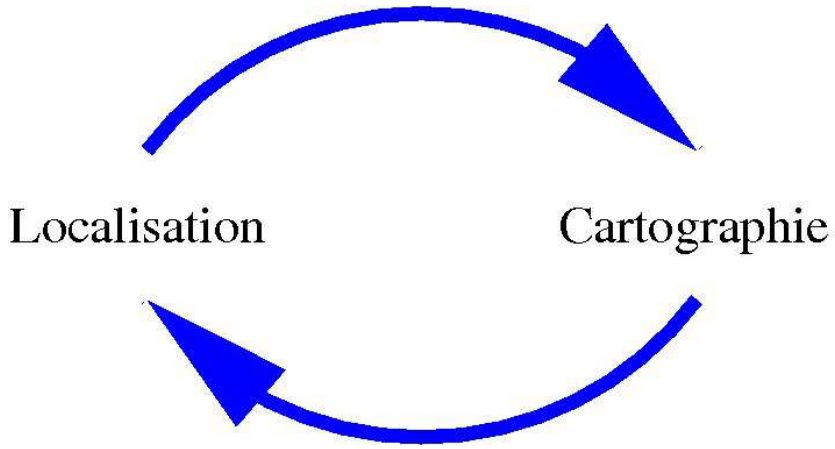


FIG. 1.1 – Le problème de SLAM.

En d'autres termes, la fonction **Cartographie et Localisation Simultanées** est le pro-

cessus qui permet de construire une représentation (carte) d'un environnement inconnu et de localiser le robot en même temps dans cette carte. Il est essentiel d'adopter une représentation stochastique de la carte afin de prendre en compte les imprécisions sur les données acquises et sur les estimées des déplacements successifs du robot et de propager ces incertitudes sur le modèle construit et sur la position du robot.

1.3 Nos principaux choix

Nous envisageons de construire un modèle géométrique dense de l'environnement intérieur. Ce modèle sera représenté par une carte hétérogène qui combine des amers plans texturés, des lignes 3D et des points d'intérêt. Afin de réaliser cette tâche, nous devons fusionner des données géométriques et photométriques. Pour cela, nous avons d'abord amélioré la stéréovision dense, en proposant une approche qui transforme le problème de la mise en correspondance stéréoscopique en un problème de minimisation d'une fonction d'énergie globale. Le minimum de cette fonction est trouvé par une coupure minimale dans un graphe. Notre contribution réside dans la construction d'un graphe réduit qui a permis d'accélérer considérablement cette méthode globale de l'appariement stéréoscopique et d'obtenir de meilleurs résultats que les méthodes locales.

Néanmoins, cette méthode reste non applicable en robotique mobile. Aussi, pour percevoir l'environnement, le robot est équipé d'un autre capteur 3D, un télémètre laser pivotant autour d'un axe horizontal. Une simple caméra permet d'acquérir les informations sur l'apparence des surfaces extraites des données laser. L'extraction des primitives à partir des données sensorielles brutes est une tâche essentielle. Ainsi nous présentons un algorithme de segmentation de l'image de profondeur issue du laser 3D afin d'extraire les facettes planes. De plus, nous détaillons la procédure pour fusionner les primitives géométriques extraites depuis les données laser et depuis les images acquises par la caméra, afin de construire de nouvelles primitives : des points et des segments 3D.

Notre chaîne algorithmique permettant de construire de manière incrémentale une carte hétérogène est fondée sur l'algorithme de Cartographie et Localisation Simultanées sous le format du filtre de Kalman étendu (EKF-SLAM). Le placage de la texture sur les facettes planes rend le modèle plus réaliste pour un opérateur, mais surtout, cela permet d'ajouter des informations d'apparence que nous utilisons pour robustifier l'étape d'association de données, une étape essentielle pour garantir la cohérence de la carte.

Nous avons validé nos travaux à partir de données acquises depuis notre robot, dans le laboratoire. Du fait du temps d'acquisition des données laser, nous n'avons pas intégré une expérimentation en temps réel. Notons que notre démonstrateur est aussi équipé d'un capteur optique à temps de vol (Swiss Ranger) : l'exploitation de cette nouvelle technologie permettra une telle intégration.

1.4 Organisation du manuscrit

Cet document est organisé en chapitres structurés de la manière suivante :

- Dans le chapitre 2, nous présentons la problématique de la modélisation 3D et rappelons les principaux concepts et formalismes sur lesquels sont basés nos travaux. Ainsi, nous mettons l'accent sur l'importance de la modélisation et les différents types de représentation, et nous détaillons l'état de l'art sur la modélisation 3D avant de terminer ce chapitre en décrivant notre cahier des charges.

- Ensuite, dans le chapitre 3, nous traitons la stéréovision, en focalisant sur la méthode de coupure de graphe, et en particulier notre contribution sur la mise en correspondance stéréoscopique par la coupure d'un graphe réduit.

- Le chapitre 4 est consacré à la perception et fusion de données multi-capteurs. Nous détaillons l'extraction des facettes planes à partir de l'image de profondeur, et l'extraction des segments 3D par la fusion de données laser et caméra. Enfin, nous traitons le problème de placage de la texture sur les facettes planes.

- Dans le chapitre 5, nous développons la construction de la carte stochastique hétérogène par l'algorithme de Cartographie et Localisation Simultanées (SLAM). Nous présentons ensuite l'utilisation des informations d'apparence pour renforcer l'appariement des données (data association).

- La conclusion et les perspectives de notre travail seront présentées au chapitre 6.

Chapitre 2

Modélisation de l'Environnement pour la Navigation

Sommaire

2.1 Introduction

Dans ce chapitre nous présentons la problématique générale de la modélisation 3D d'un environnement en robotique mobile. Nous commençons en section 2.2 par donner la définition du problème et son importance pour l'autonomie du robot mobile. Ensuite, nous présentons un rapide état de l'art sur les méthodes de type SLAM : il ne s'agit pas ici d'être exhaustif, car ce problème a généré depuis près de 20 ans, un nombre très important de contributions. En section 2.4, nous détaillons les différents types de représentions 2D proposées pour décrire un environnement intérieur. Puis, en section 2.5, nous expliquons les limitations des cartes 2D et l'importance des modèles 3D, et nous détaillons notre cahier des charges. De plus, en section 2.6, nous présentons les différents capteurs d'acquisition 3D que nous avons exploités dans nos travaux. Nos contributions seront présentées en section 2.7.

2.2 Modélisation

Une définition du robot pourrait être : "Système automatique mécanisé capable d'effectuer une ou plusieurs tâches, dans un environnement donné, de manière autonome, par l'exécution d'un programme" [1]. Le terme robot correspond donc à un type bien précis de système. Ainsi, si certaines caractéristiques ne sont pas présentes, une machine, même très complexe, ne peut être qualifiée de robot. Bien évidement, l'autonomie est un caractère intrinsèque lié au robot. Un robot autonome est un système capable d'agir et de réagir seul face à un événement imprévu. Le degré d'autonomie rapproche les robots des systèmes complètement autonomes imaginés par la science-fiction ou envisagés par la recherche de pointe.

À titre illustratif, prenons l'exemple de l'aspirateur-robot (par exemple Roomba de iRobot [2], figure 2.1) : cet appareil est capable de réagir aux obstacles qui peuvent exister dans une habitation, de les contourner et de les mémoriser. Pour cela, il sauvegarde un plan de l'appartement (**Carte**) et peut le modifier en cas de besoin. L'autonomie suppose que le robot prévoit l'occurrence de certains évènements, puis les réactions appropriées à ceux-ci. Lorsque l'aspirateur évite une chaise dont il connaît l'emplacement, il exécute un programme intégrant par exemple les coordonnées X-Y de la chaise dans la carte. Si cette chaise est déplacée ou supprimée, le robot est capable de modifier son plan, et donc de traiter une zone du sol non atteignable avant le déplacement de la chaise. Ainsi, pour que le robot atteigne un degré supérieur d'autonomie, il doit être doté de moyens pour appréhender l'environnement, construire et mettre à jour des représen-

[1] http ://fr.wikipedia.org/
[2] http ://www.irobot.com/

tations internes de cet environnement, décider des actions adéquates et réagir aux événements imprévus.

FIG. 2.1 – L'aspirateur-robot Roomba, de iRobot

En robotique mobile, nous rencontrons en général deux types d'environnements : statique et dynamique. Dans un environnement statique, tous les objets (obstacles) sont immobiles : seul le robot se déplace. Les objets peuvent se déplacer ou être déplacés dans un environnement dynamique ou évolutif. Réaliser des tâches avec un robot dans un environnement dynamique est beaucoup plus ambitieux que dans un milieu statique. Nous ne considérons dans notre travail que les environnements statiques, avec l'optique d'adapter nos contributions pour prendre en compte les aspects dynamiques dans des travaux futurs : notons qu'avec l'irruption des robots dans les lieux publics ou domotiques, l'hypothèse du monde statique ne peut être admise que dans une phase d'apprentissage contrôlée, hors présence d'humains.

Pour un robot mobile autonome, être capable de se déplacer dans un environnement non préparé est à la fois indispensable et extrêmement complexe. Avoir un modèle (plus ou moins) détaillé de son environnement est essentiel pour assurer l'exécution de ses missions.

Un modèle de l'environnement donne au robot les capacités de réalisation des tâches suivantes :

- Planification des mouvements et déplacements. En fait, sans avoir un modèle de son environnement, le robot ne peut planifier un chemin pour aller d'un point de départ à un point d'arrivée. De plus, le modèle permet au robot d'évaluer et de quantifier les actions de déplacement qu'il effectue.

- Réalisation des mouvements. Le modèle permet de détecter des éléments sur lesquels le robot s'appuie pendant son déplacement, que ce soit des amers qui lui permettent de se localiser, ou des buts successifs à atteindre dans un enchaînement de commandes référencées capteur.

- Localisation. Un bon modèle est indispensable pour le robot afin de se localiser. En effet,

la localisation du robot à chaque instant assure l'exécution du chemin planifié.

- Planification et exécution des tâches en contexte multi-robots. Un modèle commun donne les moyens à un ensemble de robots pour planifier et exécuter des tâches communes et complémentaires.

Un modèle de l'environnement peut être un ensemble de points dans un repère donné ou un modèle détaillé qui s'approche des modèles fournis par les logiciels de *Conception Assistée par l'Ordinateur* (CAO comme Autodesk 3ds Max [3] par exemple). Historiquement, le problème de modélisation en robotique mobile est traité comme un problème de construction d'une carte (Cartographie ou *Mapping*). La **Cartographie** est l'ensemble des études et des opérations, scientifiques, artistiques et techniques, intervenant à partir des résultats d'observations directes ou de l'exploitation d'une documentation, en vue de l'élaboration et de l'établissement de cartes, plans et autres modes d'expression, ainsi que dans leur utilisation [4].

La cartographie est considérée comme l'un des plus importants problèmes dans la construction d'un robot mobile quasi autonome. En dépit des progrès remarquables réalisés dans ce domaine, il reste l'un des sujets les plus ambitieux. À présent, on a seulement des méthodes sur le marché de la robotique (voir *Evolution Robotics*) pour la cartographie des environnements statiques, structurés et de tailles limitées. Cartographier des environnements non structurés, dynamiques ou de grandes tailles reste un sujet de recherche ouvert. Nous brossons un rapide état de l'art du SLAM dans la section suivante.

La cartographie en robotique mobile est habituellement divisée en deux grands types d'approches : métrique et topologique. Les cartes métriques capturent les propriétés géométriques de l'environnement, alors que les cartes topologiques décrivent la connectivité de différents lieux (pièces, couloirs, etc.). La grille d'occupation est une autre approche pour représenter une carte, dans laquelle la carte est une grille de cellules et la valeur de chaque cellule révèle si elle est occupée ou libre. Nous détaillerons ces différentes représentations 2D, puis justifierons pourquoi nous proposons une représentation 3D.

2.3 Le Problème de SLAM

L'algorithme de SLAM a suscité un intérêt remarquable dans la communauté de robot mobile car il est un outil permettant une navigation autonome totale [Castellanos *et al.*, 2000; Dissanayake *et al.*, 2000; Feder *et al.*, 1999; Leonard et Durrant-Whyte, 1991; Newman, 1999; Rencken, 1993; Thrun *et al.*, 1998]. L'algorithme de base fondé sur le filtre de Kalman (EKF-SLAM) apparaît dans le papier de Smith et al. [Smith *et al.*, 1988, 1990], qui s'inspire des travaux antérieurs de Ayache et Faugeras [Ayache et Faugeras, 1988] et de Chatila et Laumond [Chatila et Laumond, 1985]. Un robot capable de construire une carte de l'environnement et en même temps se localiser dans cette carte peut fonctionner plus longtemps de façon autonome dans des environnements inconnus. La majorité des travaux est focalisée sur les techniques d'estimation stochastique pour construire et maintenir les estimations de la position du robot et des positions

[3] http ://www.autodesk.com
[4] Serveur Éducatif sur l'Information Géographique (SEIG) http ://seig.ensg.ign.fr/

des amers. En particulier, le filtre de Kalman étendu (EKF) était proposé comme un mécanisme permettant de fusionner les informations acquises par le robot pour obtenir une carte cohérente de manière incrémentale. Une autre méthode d'estimation est la méthode FastSLAM. L'algorithme FastSLAM [Montemerlo *et al.*, 2002, 2003], utilise un filtre particulaire pour l'estimation de la position du robot. Chaque particule représente une position possible du robot, l'incertitude sur la connaissance de cette position est représentée par la distribution de ces particules dans l'espace. Un état de l'art très détaillé sur l'histoire et les différentes techniques de l'algorithme SLAM peut être trouvé dans [Durrant-Whyte et Bailey, 2006].

Citons quelques évolutions plus récentes de cette problématiques :

- De nombreux travaux essayent de limiter la complexité de la mise à jour de la carte, qui est en $O(n^2)$ si n est le nombre d'amers dans la carte. Citons les travaux de Guivant et Nebot, qui propose une approche *batch* afin de mettre à jour la carte de manière différée, sans que cela impacte sur la précision de la modélisation.

- La notion de *Hierarchical SLAM* proposée par Tardos et Neira permet également de *casser* la complexité du SLAM, en passant par des sous-cartes intégrées de manière hiérarchique dans une seule carte globale.

- De nombreux auteurs ont proposé de décrire un environnement complexe et de grande taille, par un ensemble de sous-cartes, négligeant ainsi les corrélations entre les amers intégrés dans des cartes différentes. Un tel modèle est particulièrement adapté si des contraintes de visibilité font que de tels amers ne peuvent être perçus ensemble. C'est le cas en milieu intérieur avec des cartes construites pour chaque pièce, ou en milieu extérieur lorsque les amers sont regroupés dans des lieux séparés.

- Une carte stochastique est une représentation dite *éparse*, car ne contenant que des amers utiles pour la localisation du robot. Mais un robot a besoin d'autre représentations pour gérer ses déplacements ou pour planifier des actions, représentations plus denses comme les grilles d'occupation ou les cartes d'élévation. Le concept de *Dense SLAM* a été proposé par Nebot pour densifier les modèles construits par le robot sans accroître la complexité de la carte stochastique.

- Pour l'algorithme SLAM initial, les amers sont supposés fixes ; dans des travaux récents, cet algorithme a été modifié pour donner naissance au SLAMMOT (SLAM et *Mobile Object Tracking*), avec lequel un robot peut apprendre un environnement dynamique.

Ces approches ont été validées essentiellement par la construction des représentations bidimensionnelles d'environnement intérieur à partir des données télémétriques (coupes laser). On peut trouver de nombreux robots évoluant en intérieur, exploitant un télémètre SICK et une carte de segments 2D pour se localiser en continu. Récemment, le SLAM 3D a attiré l'attention. Takezawa et al. [Takezawa *et al.*, 2004] décrivent un cadre pour le SLAM basé sur des amers 3D. Jung [Jung, 2004] construit une carte 3D de points d'intérêt dans un milieu extérieur à partir de données stéréo. Lemaire et Sola [Sola *et al.*, 2005] produisent de telles cartes par vision monoculaire.

Ces représentations 3D très éparses permettent essentiellement au robot de se localiser. Notre travail se focalise sur la production de modèles surfaciques pour représenter un milieu intérieur. Les surfaces planes y sont nombreuses (plafond, sol, murs etc.), d'où l'intérêt de les utiliser comme amers. Notre but sera donc de construire une carte géométrique stochastique faite à partir des primitives planes. De plus, nous associerons des données d'apparence à ces primitives planes. Ces informations d'apparence vont permettre une meilleure association de données, et en même temps ajoute de la richesse à la représentation. Les données issues du télémètre laser 3D (on parle souvent de scanner 3D) et d'une caméra sont fusionnées pour définir des amers lignes 2D, et des points d'intérêt collés sur les plans. Ainsi nous construisons une carte hétérogène contenant en plus des amers plans, des lignes 2D et des points d'intérêt représentés dans des repères locaux liés aux plans porteurs.

Les approches SLAM génèrent donc des modèles dédiés surtout à l'autolocalisation du robot ; pour gérer ses missions, un système robotique a besoin d'autres représentations qui sont décrites dans la section suivante dans le cas 2D.

2.4 Représentations 2D d'un environnement intérieur

En robotique mobile, plusieurs types de représentation de l'environnement (cartes) ont été développés. Il existe un consensus général sur le fait que ces différents types ont des avantages et des limitations et qu'ils sont plus ou moins adaptés selon la mission à accomplir. Par exemple, les cartes métriques sont difficiles à construire et à maintenir à cause de l'incohérence entre le mouvement du robot et la perception. Elles sont moins adaptées pour les problèmes symboliques. En revanche, les cartes topologiques sont mieux adaptées pour représenter les grands environnements, pour rajouter un niveau symbolique ou pour communiquer avec l'homme. Mais ces cartes permettent seulement au robot de se localiser de manière globale, et de planifier la trajectoire de façon sous optimale. Lors de la construction des cartes topologiques, la distinction des différentes composantes est difficile sans l'utilisation d'informations métriques.

Ces représentations s'avèrent souvent complémentaires : c'est le cas par exemple des approches métriques et topologiques, en particulier, leur utilisation conjointe est susceptible de favoriser l'exploitation de la carte résultante pour les besoins de navigation du robot. C'est pourquoi les approches hybrides, qui combinent différents types de modèles élémentaires, se généralisent.

2.4.1 Cartes Métriques

L'approche métrique vise à produire une représentation géométrique plus ou moins détaillée de l'environnement à partir des données perceptuelles. Les informations de longueur, distance, position etc., apparaissent explicitement dans les cartes métriques et sont en général définies dans un référentiel unique. Ces cartes sont souvent plus faciles à comprendre par l'homme car elles offrent une relation bien définie avec le monde réel [Thrun, 2002].

La construction des cartes métriques nécessite la gestion de la cohérence géométrique de l'ensemble de la représentation. En particulier lorsque le robot retourne sur un lieu connu après

avoir réalisé une boucle dans l'environnement (fermeture de la boucle). En effet, la fermeture de la boucle est l'un des défis majeurs de la construction des cartes métriques.

Les cartes métriques sont en général très génériques. Elles peuvent a priori représenter tous types d'environnement. La figure 2.2 illustre une carte 2D construite dans les couloirs de notre laboratoire, par un des démonstrateurs de notre équipe, muni d'un télémètre laser. Dans cette figure, on constate clairement la divergence de l'estimation de la position du robot obtenue par intégration des déplacements successifs, et la non-cohérence de la carte construite par les données odométriques et qui sera corrigée par l'algorithme SLAM.

2.4.2 Cartes Topologiques

Une carte topologique est une représentation plus abstraite décrivant les relations entre les éléments de l'environnement, sans utiliser un repère de référence absolu [Fabrizi et Saffiotti, 2002]. Elles se présentent sous forme de graphes, dont les sommets correspondent à des lieux, souvent associés à des informations perceptuelles (histogrammes de couleurs, images, données télémétriques, etc.), et dont les arêtes indiquent l'existence d'un chemin traversable par le robot, reliant les lieux associés aux deux sommets de l'arête (voir la figure 2.3).

L'avantage principal des cartes topologiques est de s'abstraire des problèmes d'incertitudes dans le mouvement des robots : les incertitudes ne s'accumulent pas globalement car le robot se contente de naviguer localement, entre endroits. Ces cartes peuvent présenter un découpage de l'espace qui facilite l'interaction avec l'homme, notamment si les lieux correspondent à des pièces ou à des couloirs : par exemple, on peut imaginer de donner l'ordre au robot d'aller à la salle de séjour plutôt qu'aux coordonnées cartésiennes (x,y). À ce propos, voir la bande dessinée (figure 2.4) créée par des doctorants de notre laboratoire, qui a gagné le prix de IEEE 2008. Thrun [Thrun, 2002] souligne que ces modèles sont bien adaptés aux planificateurs et aux systèmes de résolution de problèmes symboliques, ainsi qu'à l'interaction en langage naturel car on peut facilement y ajouter des informations sémantiques.

En revanche, le principal inconvénient des modèles purement topologiques est l'absence d'informations géométriques. Cette lacune peut empêcher le robot de réaliser des raisonnements spatiaux sur l'ensemble de son environnement. En particulier, le système robotisé peut avoir des difficultés à sélectionner le chemin optimal entre deux lieux [Thrun, 2002] : d'une part, le manque d'information sur la longueur des chemins (sur les arcs) peut l'empêcher de choisir entre deux branches du graphe qui mènent au même endroit, et d'autre part, il n'est pas possible de trouver un chemin plus direct dans l'espace métrique 2D que ceux qui sont implicitement codés dans les arêtes du graphe. Si les sommets sont impossibles à distinguer, la construction d'une carte purement topologique nécessite de mettre en oeuvre un processus très coûteux et peu efficace lors de la création d'un nouveau sommet [Dufourd, 2005].

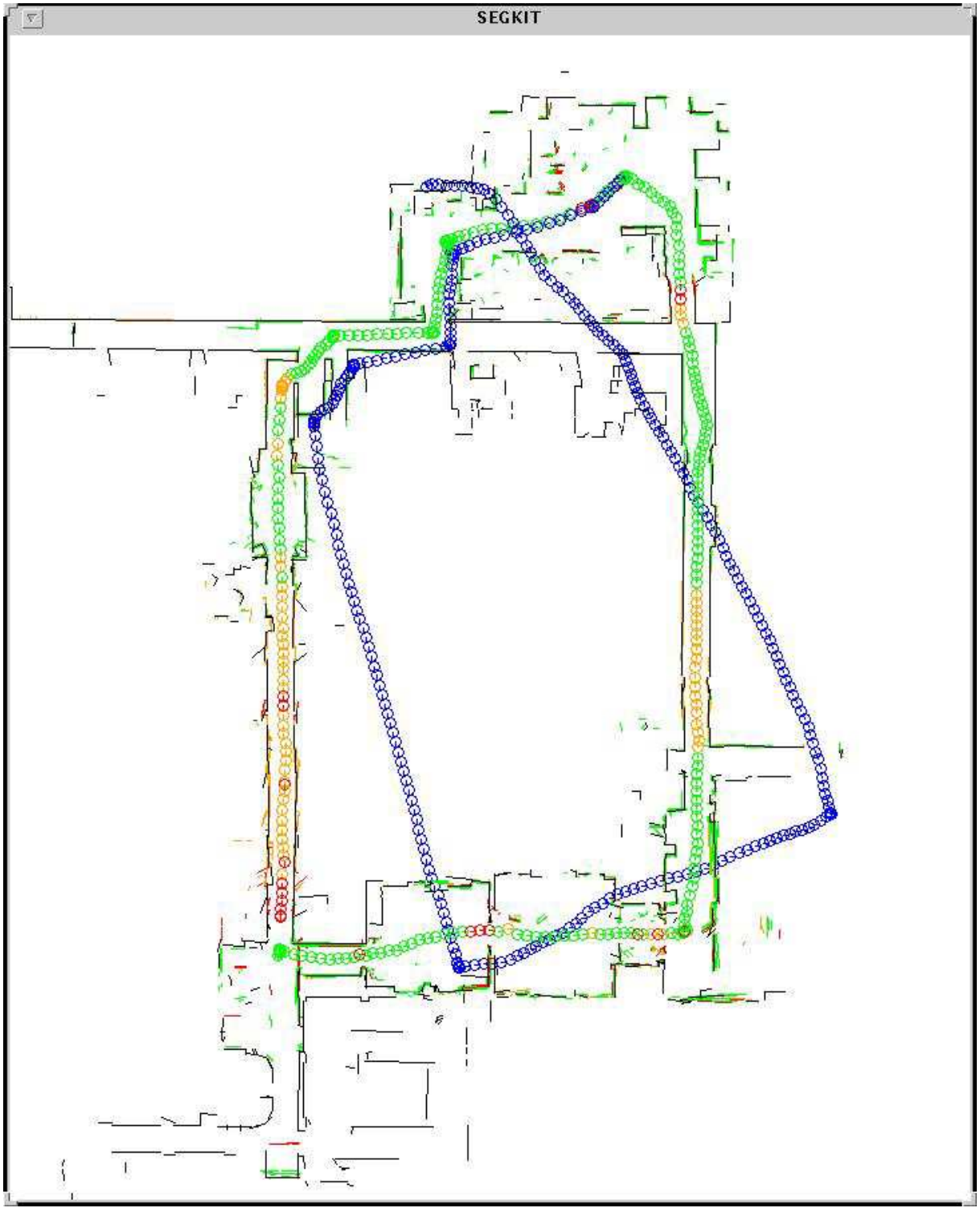


FIG. 2.2 – Carte métrique de l'environnement avec segments 2D. Les positions du robot données par l'odométrie sont en bleu, et celles corrigées par l'algorithme SLAM sont en vert et orange.

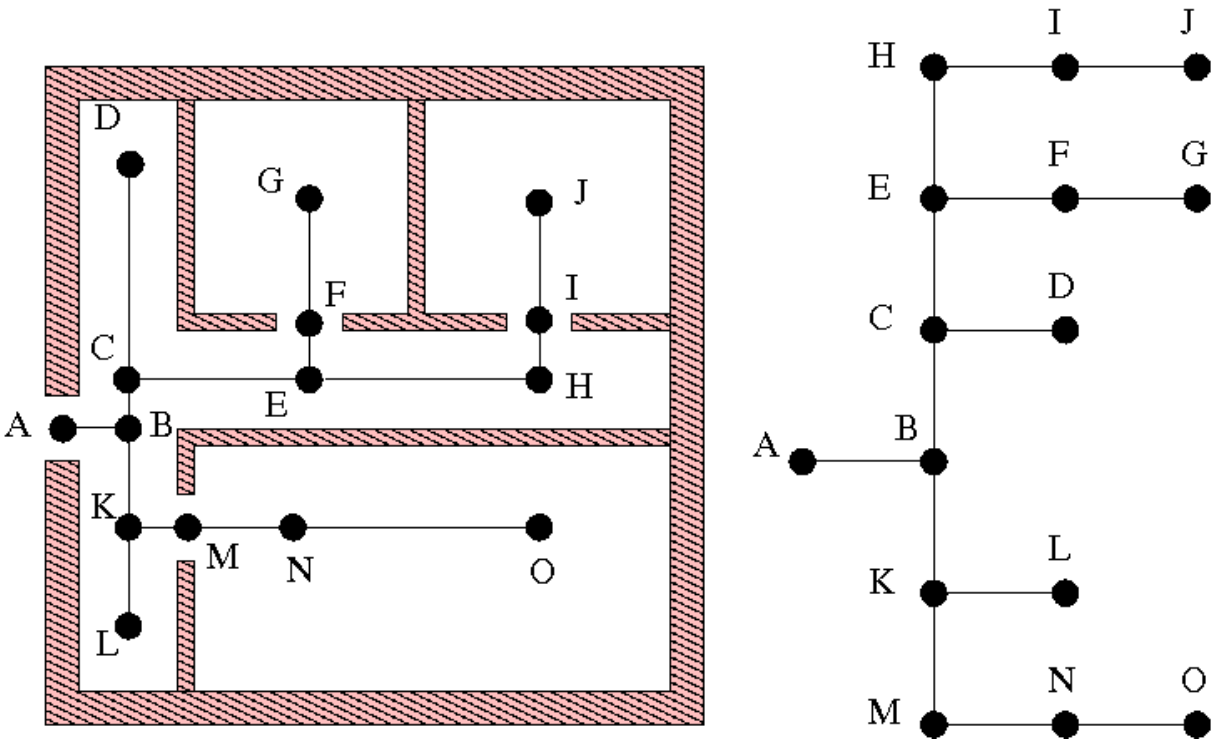


FIG. 2.3 – Carte métrique de l'environnement avec des lieux topologiques reliés entre eux par des relations d'adjacence et la carte topologique correspondante.

2.4.3 Grilles d'Occupation

Dans la représentation de l'environnement sous forme d'une grille d'occupation, l'espace est partitionné en un ensemble de cellules distinctes. Un vecteur d'attributs (éventuellement un seul nombre ou un seul bit dans le cas de cartes binaires) est attaché à chacune des cellules pour représenter ses propriétés : souvent, il s'agit du degré d'encombrement par un obstacle (indice indiquant que la cellule correspondante est occupée ou non par un obstacle).

Les grilles d'occupation constituent une représentation surfacique simple et populaire. Dans ce type de modèle, l'espace est discrétisé selon une grille régulière en cellules carrées ou rectangulaires de même taille. Chaque cellule contient un indice (probabilité, histogramme, etc.) indiquant si l'espace correspondant est plutôt libre ou occupé. La figure 2.5 illustre une grille d'occupation construite pour une partie de la salle robotique dans notre laboratoire [Baba, 2007]. La taille de cellule (5 x 5 cm^2) fournit une bonne résolution pour la carte élaborée.

L'avantage principal des grilles d'occupation est leur capacité à représenter l'espace de manière très dense, en fonction du pas de discrétisation de la grille. Elles sont adaptées à des environnements de forme quelconque, et elles donnent une estimation statistique de la confiance dans les données. De plus, elles fournissent des informations d'occupation et donc sur les positions des obstacles. Ainsi, elles sont souvent utilisées lorsque l'application visée repose sur la connaissance de l'espace libre, en particulier pour la planification de trajectoires (à partir de transformations en distance ou de champs de potentiels par exemple). Elles sont en général relativement faciles

FIG. 2.4 – Cette bande dessinée a remporté le prix de "IEEE 2008 Comic Contest", par : Mathias Fontmarty, Akin Sisbot, Mathieu Warnier

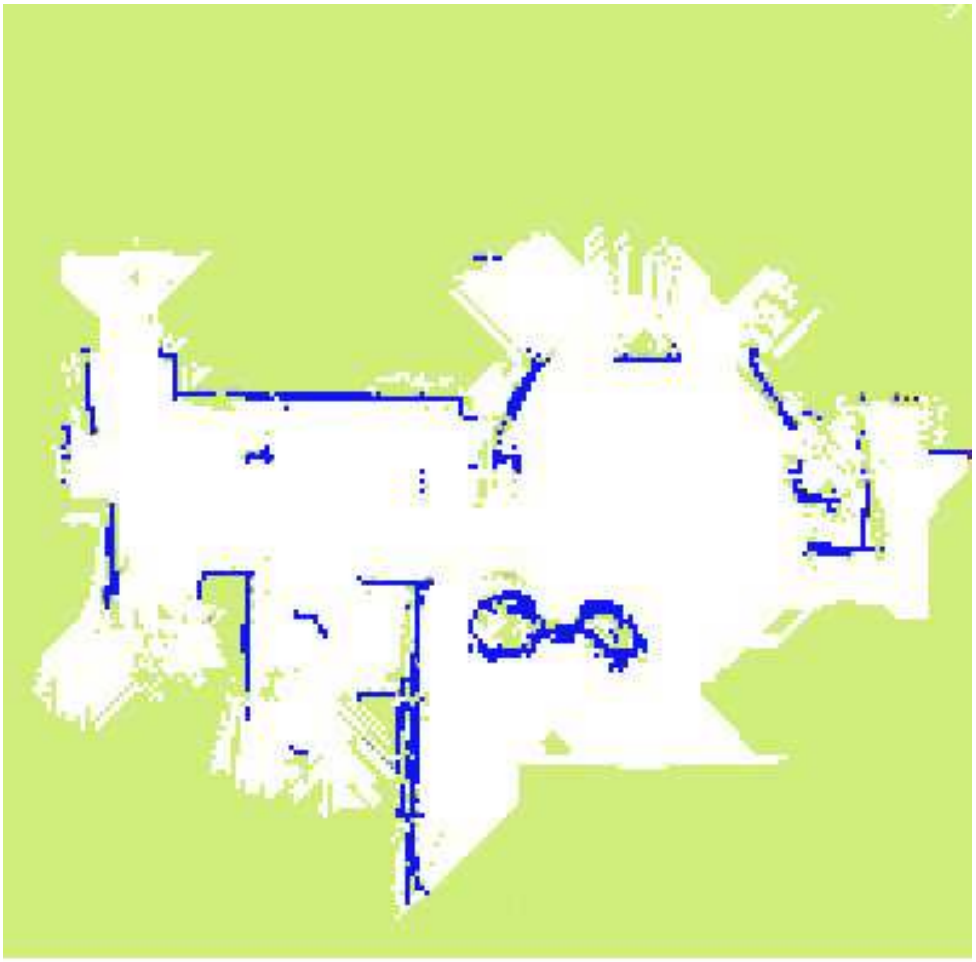


FIG. 2.5 – Exemple d'une grille d'occupation 2D construite dans notre laboratoire [Baba, 2007]. Les valeurs d'occupation sont codées comme suit : (vert : inconnu ; blanc : libre ; bleu : occupé). La superficie est (10 x 10 m), une cellule correspond à (5 x 5 cm).

à interpréter par l'homme. Le principal inconvénient des grilles d'occupation réside dans leur manque de compacité : elles sont plutôt adaptées à la représentation d'environnements encombrés mais s'avèrent particulièrement inefficaces dans les grands espaces vides. De plus, la finesse de la discrétisation étant prédéfinie, elles ne sont pas capables de s'adapter automatiquement à la densité ou à la taille des obstacles. En conséquence, si les grilles d'occupation se prêtent bien à certains algorithmes de planification, elles peuvent cependant se révéler inefficaces du fait du manque d'adaptation à l'échelle de l'environnement (en raison de la multiplication des cellules libres dans les espaces ouverts par exemple). Les approches hiérarchiques telles que les quadtrees peuvent toutefois limiter ces problèmes de compacité et d'adaptation à l'échelle. On peut également reprocher aux représentations de type grille, le processus de positionnement complexe et coûteux qu'elles entraînent pour le véhicule (notamment par rapport à une carte de balises où les objets à comparer sont moins nombreux) ; vis à vis du SLAM, ce type de représentation est inadaptée, puisqu'elles ne peuvent pas être corrigées lorsque la position du robot s'améliore. De plus, elles paraissent peu appropriées, contrairement aux cartes topologiques par exemple, aux algorithmes de résolution de problèmes exprimés de manière symbolique.

2.4.4 Modèle de primitives

Les primitives (*features* en anglais) ou les amers (*landmarks* en anglais) sont des parties distinctives de l'environnement que l'on peut facilement extraire via un type de capteur donné et qui admettent une description paramétrique : notons qu'on parle généralement de balises quand il s'agit d'amers artificiels.

Pour la navigation maritime, un amer est un point de repère fixe et identifiable sans ambiguïté [5]. Ce repère visuel identifiable sans ambiguïté est utilisable pour prendre des relèvements optiques (au compas de relèvement), ou pour naviguer sur un alignement. Un phare, un château d'eau, un clocher, un pignon ou un arbre peuvent constituer des amers. En robotique mobile, on utilise souvent le terme *amer* comme dans la marine.

Un amer géométrique pour la localisation doit vérifier plusieurs critères : pouvoir discriminant, domaine de visibilité important, stabilité, invariance et bonne adaptation à la métrologie. Et afin de maintenir une représentation cohérente de l'environnement, les cartes d'amers doivent de plus modéliser les incertitudes sur ces primitives géométriques. En général, comme nous le verrons, ces incertitudes sont représentées par des distributions gaussiennes sur les paramètres géométriques de la primitive (positions cartésiennes ou polaires, longueurs, etc.).

La représentation par amers fournit un cadre de travail mieux adapté pour résoudre le problème de la Cartographie et Localisation Simultanées (SLAM), à travers les approches basées sur le filtre de Kalman étendu (EKF) : la fermeture de boucles est gérée de manière transparente via la matrice de covariance. Le nombre des amers étant limité, les mises en correspondance sont souvent moins coûteuses que dans une grille d'occupation par exemple.

Par ailleurs, les cartes d'amers offrent une description plus compacte des larges espaces ouverts que les grilles d'occupation. Mais elles sont moins adaptées aux environnements encombrés de petits objets où les amers se multiplient. Cependant, les cartes d'amers peuvent en principe atteindre une meilleure précision que les grilles d'occupation car les objets sont localisés selon des coordonnées en nombres flottants alors que dans les grilles, l'espace est discrétisé.

En revanche, les cartes d'amers sont souvent peu denses et contrairement aux grilles d'occupation, elles ne contiennent pas les informations d'encombrement nécessaires pour la planification de trajectoires. En effet, les amers ponctuels renseignent peu sur la position des obstacles continus de grande dimension et les amers de type *segment* ne sont généralement pas jointifs : le contour des obstacles est rarement continu et fermé, ce qui génère des ambiguïtés (par exemple, on peut se demander si l'on a observé une ouverture dans l'interstice entre deux murs non jointifs). De plus, on constate fréquemment des incohérences dans la carte, comme l'illustre la figure 2.2 pour une représentation à base de segments : segments redondants suite à un appariement manqué, intersections aberrantes entre frontières d'obstacles, etc. En général, les représentations à base d'amers de haut niveau sont faciles à appréhender par l'homme. Les cartes d'amers constituées par des amers géométriques structurés (segments, plans, etc.) voire par des objets, sont plus faciles à comprendre que celles avec des amers très basiques (notamment des points).

Les cartes d'amers géométriques sont peu adaptées pour représenter l'environnement d'exté-

[5] http ://fr.wikipedia.org/

rieur. Cependant, ce type de représentations est relativement bien adapté aux environnements structurés tels que l'intérieur des bâtiments, mais, elle est souvent limitée à des régions pouvant être décrites sous forme d'éléments géométriques simples (points, segments, plans, etc.). Les approximations polygonales sont mal adaptées aux environnements présentant des courbes, et donc sont difficilement ajoutées à la carte. Thrun suggère une solution à ce problème : ajouter à la carte des modèles d'objets. On constate que les cartes d'amers sont généralement capables de s'adapter automatiquement à la densité des objets et aux changements d'échelle, ce qui constitue un avantage par rapport aux grilles d'occupation.

2.4.5 Cartes Hybrides

L'idée des cartes hybrides est d'utiliser deux (ou plusieurs) types de représentation ensemble, ce qui va permettre de profiter des points forts de chaque représentation, et pourrait aider à surmonter les points faibles. Par exemple, plusieurs auteurs ont proposé des méthodes hybrides métriques-topologiques dans l'intention de combiner la précision des cartes métriques avec l'extensibilité des cartes topologiques [Thrun, 1998; Chong et Kleeman, 1997; Simhon et Dudek, 1998; Gasós, 1999; Duckett et Saffiotti, 2000; Tomatis, 2001; Bosse *et al.*, 2003].

Par exemple, afin d'obtenir une représentation dense de l'environnement, les grilles d'occupation peuvent être utilisées pour représenter les informations denses, un algorithme de SLAM basé sur les *features* est utilisé pour estimer les positions du robot. Mais, si les deux cartes sont indépendantes, la carte globale sera incohérente. Un exemple d'incohérence peut être illustrée lors la fermeture de boucle. L'algorithme *feature-based SLAM* est capable de propager l'incertitude en arrière pour toute la carte, ce qui permet d'avoir une carte cohérente d'amers. Mais, si les deux cartes sont indépendantes, les corrections réalisées par l'algorithme SLAM ne seront pas propagées dans la carte d'occupation. Cela donne par exemple qu'un objet se trouve dans deux positions différentes dans les deux cartes. La solution est de maintenir les correlations entre la carte d'amers et la grille d'occupation. Maintenir toutes les corrélations entre les grilles d'occupation et la carte d'amers résout le problème de cohérence, mais cette solution est envisageable seulement pour les petits environnements.

Nieto et al. [Nieto *et al.*, 2004] ont proposé une solution à ce problème par la construction d'une carte métrique et hybride (*HYbrid Metric Maps* (HYMMs)) qui combine les éléments de la carte avec d'autres informations sensorielles. Les auteurs présentent comment combiner efficacement un algorithme SLAM basé sur les *features* et une carte sous forme d'une grille d'occupation. La carte globale est partionnée en un ensemble de régions triangulaires locales connectées, qui fournit un repère pour une description multi-échelles de l'environnement. Pour eux, la carte hybride est représentée par un graphe de systèmes de coordonnées, telle que chaque sommet dans le graphe représente un repère local, et les arêtes représentent les transformations entre les repères adjacents.

Bien que les cartes hybrides offrent plus d'avantages que les cartes individuelles, elles possèdent des problèmes qui n'apparaissent pas dans les cartes simples. Un problème évident est la complexité introduite par les synergies. Par exemple, la sortie de localisation provenant d'une composante va être utilisée comme l'entrée d'un capteur virtuel, et donc on doit définir un modèle sensoriel pour cette entrée, ce qui n'est pas toujours évident.

D'autre part, on risque d'avoir une propagation de l'erreur via les différentes composantes. Une erreur dans une composante va se propager sans qu'on puisse la contrôler dans les autres composantes [Gasós, 1999], [Tomatis, 2001].

Enfin, nous mentionnons le problème connu sous le nom de *location seeding*. Considérons une carte hybride, dont plusieurs cartes sont utilisées pour couvrir les différentes parties de l'environnement. Supposons que le robot est en train de sortir d'une carte et d'entrer dans une autre. Comment initialiser son état ? La robustesse et la flexibilité de la carte hybride résultante dépendront de la façon dont le problème de *location seeding* est traité.

2.5 Représentations 3D d'un environnement

Les approches citées précédemment ont été essentiellement validées par la construction de représentations bidimensionnelles d'environnement intérieur à partir de données télémétriques. De nombreux robots évoluant en intérieur exploitent un télémètre laser et une carte de segments 2D pour se localiser en continu. Récemment, la modélisation 3D a attiré l'attention, et les applications exploitant une carte 3D commencent à apparaître.

2.5.1 Pourquoi la Modélisation 3D

Nous cherchons dans notre travail à obtenir un modèle tridimensionnel de l'environnement d'intérieur (*Indoor environment*). Un environnement d'intérieur, appelé aussi structuré, peut être l'intérieur d'un appartement, d'un laboratoire, d'une usine, ou tout autre structure construite par l'homme. Par contre, un environnement d'extérieur correspond généralement à un espace naturel peu structuré, s'il n'a pas été modifié par l'Homme.

La modélisation de l'environnement tente de représenter au mieux l'environnement réel. Un modèle bidimensionnel n'est pas suffisant pour les raisons suivantes :

- Un modèle 2D ne peut pas représenter un monde tridimensionnel. Pour beaucoup d'applications développées auparavant, le robot ne se déplaçait sur un sol horizontal, et donc pouvait être considéré comme se déplaçant que sur un plan horizontal, et en conséquence, un modèle bidimensionnel (pour la localisation et la commande) était suffisant pour achever le travail. Dans de nouvelles applications robotiques, un modèle 2D ne peut pas garantir le bon fonctionnement. Prenons par exemple le scénario suivant : dans le projet Cogniron, qui concerne un compagnon domestique, le robot est mis en situation où il doit aller chercher un objet (un verre d'eau par exemple) situé sur une table. Si on utilise un modèle 2D de l'environnement, comme par exemple une carte des segments 2D qui se trouve dans le plan horizontal du scanner laser (voir la figure 2.2), la table serait (au mieux) représentée par quatre points (ou très petits segments). Donc pour le robot, l'espace entre les pieds de la table est vide et le robot peut y accéder sans problème. Nous voyons dans cet exemple, les lacunes du modèle 2D.

FIG. 2.6 – Le robot humanoïde HRP2 essaie d'attraper une balle posée sur un support.

- Une carte 2D considère que le sol est parfaitement horizontal. Alors que dans la réalité ceci n'est pas toujours garanti (présence des plans inclinés par exemple).

- Les robots humanoïdes sont les robots du futur, voir en figure 2.6 la photo du robot HRP2 qui essaie d'attraper une balle posée sur un support 3D. Il existe déjà des humanoïdes qui sont capables de monter sur des marches. Comment peut-on représenter des escaliers avec un modèle 2D ?

- Pour que les robots mobiles munis d'un bras (voir par exemple le robot Jido en figure 2.7) puissent planifier leurs missions, ils doivent connaître ce qui existe autour de leur bras et autour de leurs trajectoires. Pour que le robot soit capable d'attraper le verre (dans notre exemple précédent), il doit connaître les coordonnées (x, y, z) tridimensionnelles de l'objet, et en plus les obstacles qui se trouvent dans l'espace 3D autour de lui.

Pour ces raisons, et pour d'autres, nous voyons clairement les besoins de construire un modèle 3D. Il nous reste à choisir la façon de le représenter. Une carte topologique ne peut pas répondre au besoin cité auparavant, ce qui nous dirige vers des cartes métriques. Les grilles d'occupation généralisées pour le 3D sont les "voxel maps" (un voxel est une cellule volumique : petit cube),

obtenues en divisant l'espace en cubes ; chaque voxel contient un indice indiquant la présence d'obstacle ou non. On peut imaginer l'explosion en nombre de voxels, même pour une petite chambre. Par exemple, pour une chambre de longueur 4 m , de largeur 4 m, et de hauteur 3 m, si on prend par exemple un pas de discrétisation de 5 cm (fréquemment utilisé pour les cartes 2D), le nombre de voxels dans cette chambre s'élève à 384000 voxels. On peux imaginer le nombre énorme lorsqu'on essaie de représenter un appartement ou un laboratoire par des voxels. Donc, la solution est d'adapter des cartes d'amers. Et comme nous l'avons déjà vu, ces cartes sont mieux adaptées pour l'environnement d'intérieur.

2.5.2 État de l'art de la Modélisation 3D

Suite à des besoins récents, la modélisation 3D a attiré l'attention. Nous pouvons classifier les travaux de modélisation 3D selon les capteurs utilisés ou selon les types des cartes construites. Pour ce qui est des capteurs utilisés, nous trouvons plusieurs approches de la modélisation 3D qui exploitent différents capteurs : caméras, stéréovision, capteurs lasers 2D multiples, ou une combinaison de ces capteurs.

Iocci et al. [Iocchi *et al.*, 2000] construisent un modèle fondé sur des facettes planes de l'environnement d'intérieur en utilisant la stéréovision, et en introduisant quelques assistances manuelles dans le processus de construction pour surmonter le problème de manque d'informations dû à l'homogénéité (mur monocouleur par exemple) des environnements d'intérieur. Pour surmonter les problèmes de stéréovision passive, [Diebel *et al.*, 2004] utilisent une stéréovision active pour construire une carte métrique 3D. Thrun et al. [Thrun *et al.*, 2003] utilisent deux scanners laser 2D orthogonaux pour la construction d'un modèle 3D de l'environnement d'intérieur. Alors que Weingarten [Weingarten, 2006] utilise un scanner laser 2D monté sur un axe motorisé afin de pouvoir acquérir des données 3D précises, mais cela demande un temps d'acquisition considérable du fait de la rotation du scanner. Pour la génération d'une grande carte 3D pour une portion d'une ville, réalisée par [Früh et Zakhor, 2004], utilisent un scanner laser horizontal (pour la localisation du véhicule), une caméra de grand champ de vue et un scanner laser vertical pour la reconstruction des façades des immeubles. On trouve aussi l'approche basée sur la vision monoculaire proposée par Davison [Davison *et al.*, 2007]. Jo et al. [Jo *et al.*, 2006a] utilisent un scanner et une caméra pour construire un modèle 3D texturé. Biber et al. [Biber *et al.*, 2004] utilisent un robot mobile équipé d'un scanner laser et d'une caméra panoramique. Nous détaillons les capteurs d'acquisition 3D en section 2.6.

Notons sur la construction de cartes de facettes planes 3D, les travaux préliminaires de Nashashibi [Nashashibi et Devy, 1993], qui ont conduit à des validations hors ligne uniquement. Les travaux de Thrun et al. [Thrun *et al.*, 2000] sont fondés sur l'exploitation de deux télémètres laser fournissant des coupes dans des plans horizontal et vertical, et exploitent le mouvement pour produire un modèle dense de points 3D, sur lequel un maillage peut être construit a posteriori. La position du robot est estimée en ligne en utilisant le scanner laser à balayage horizontal. La carte 3D construite est une collection de polygones locaux obtenues directement des données brutes. Liu et al. [Liu *et al.*, 2001] se basent sur ces travaux pour obtenir des estimations initiales qu'ils utilisent pour leur algorithme de maximum de vraisemblance (*Expectation Maximization*). Ils utilisent une caméra panoramique pour texturer les facettes obtenues.

Hähnel et al. [Hähnel *et al.*, 2003] ont proposé un algorithme pour la génération d'un modèle 3D simplifié de l'environnement d'intérieur. Ils utilisent aussi deux scanners lasers, l'un est horizontal et l'autre est vertical pour acquérir les données 3D. Ensuite, un algorithme d'identification récursive des surfaces est utilisé pour extraire les grandes facettes planes. Leur modèle 3D contient des facettes planes 3D et des nuages de points pour les régions qui ne peuvent pas être modélisées par des facettes planes.

Weingarten [Weingarten, 2006] utilise un scanner laser 3D pour acquérir les données. Il propose un algorithme de segmentation de l'image de profondeur pour extraire les facettes planes 3D. Ensuite il met en oeuvre un algorithme de type EKF-SLAM pour reconstruire un modèle 3D. Cette contribution de Weingarten est proche de la méthode que nous proposons dans notre travail. En revanche, pour lui, il ne traite que des données laser, et donc, il n'utilise aucune information photométrique.

Takezawa et al. [Takezawa *et al.*, 2004] décrivent un cadre pour le SLAM basé sur des amers 3D en utilisant la stéréovision. Ils utilisent des amers artificiels distribués dans l'environnement pour faciliter l'extraction et la reconnaissance des ces amers, ce qui rend leur méthode assez restrictive. Jung [Jung, 2004] construit une carte 3D de points d'intérêt dans un milieu extérieur à partir de données stéréo. Lemaire et Sola [Sola *et al.*, 2005] produisent de telles cartes en vision monoculaire. Pour eux, la carte stochastique est un ensemble de points 3D distribués dans l'espace. C'est une variante de l'approche *Bearing-Only SLAM* fondée sur l'extraction de points d'intérêts (Harris, SIFT, etc.) et les techniques de vision monoculaire [Davison *et al.*, 2007].

Abuhadrous et al. [Abuhadrous *et al.*, 2004] ont développé une approche pour modéliser des sites urbains, en exploitant une méthode hybride qui combine une centrale inertielle et l'odométrie pour localiser le véhicule porteur, donc sans traiter réellement du SLAM : de ce fait les résultats obtenus sont biaisés par les erreurs incrémentales de l'odomètre. Signalons enfin dans les travaux en vision monoculaire, les résultats obtenus sur la détection de plans à l'aide d'homographie dans [Silveira *et al.*, 2006], et sur l'exploitation de tels plans dans des approches SLAM.

D'autres travaux sont focalisés sur l'extraction des primitives sémantiques (spécialement les murs) à partir des données 3D. Les cartes composées des structures sémantiques sont connues aussi comme les cartes des objets (*object maps*) [Thrun, 2002]. Un premier exemple est proposé par [Iocchi *et al.*, 2000], dans lequel les plans texturés du sol et des murs d'environnement d'intérieur (office) sont extraits par des données stéréoscopiques. Cette approche exploite l'hypothèse de planarité et traite le problème de fermeture de boucle en supposant que les murs sont orthogonaux. L'acquisition d'une carte multi-plans est étudiée aussi par [Thrun *et al.*, 2003], en utilisant deux scanners laser orthogonaux et un algorithme de *Expectation Maximization* pour l'extraction des primitives planes des données 3D, et ils considèrent que le robot est bien localisé dans ce travail. Une approche similaire est proposée par [Nüchter *et al.*, 2003], où les facettes planes sont détectées en utilisant une méthode qui exploite les techniques bien connues ICP et RANSAC. Des étiquettes (*Labels*) sont affectées aux facettes planes selon une sémantique prédéfinie qui implémente des connaissances générales de la scène. De plus, certaines contraintes comme le parallélisme, perpendicularité, etc. sont déduites par les étiquettes. D'abord, l'ensemble des plans est filtré : les facettes voisines de normale parallèle sont fusionnés. Puis, l'ensemble des plans est raffiné afin de respecter les contraintes sémantiques introduites dans l'étape d'étiquetage. Cette méthode offre une solution bien adaptée pour modeler l'environnement d'intérieur.

Enfin, nous citons deux travaux qui utilisent la fusion de données acquises par deux capteurs. Biber et al. [Biber *et al.*, 2004, 2005] utilisent un robot mobile équipé d'un scanner laser et d'une caméra panoramique. Le traitement démarre pour obtenir d'abord un modèle 2D par le scanner laser, ensuite les primitives 3D (murs) sont extraits en se basant sur la carte 2D, enfin la texture est ajoutée aux plans. Jo et al. [Jo *et al.*, 2006a,b] utilisent un scanner laser et une caméra pour construire un modèle 3D texturé. Tout d'abord le scanner laser 2D est utilisé avec une méthode de *scan matching* pour construire le modèle 2D. Ensuite le modèle 3D est construit en utilisant l'hypothèse suivante : tous les murs sont verticaux. Cette hypothèse est bien justifiable en milieu d'intérieur. La génération de la carte 3D est automatisée à partir de la carte 2D. En effet, ils ne construisent pas un modèle 3D à part, mais ils essaient d'afficher le modèle 2D sous forme d'un modèle 3D, en considérant que la carte 2D est la projection sur le sol d'une carte 3D composée par des murs. Pour améliorer l'affichage, la génération de texture est focalisée sur les murs. Pour ces deux travaux, la construction de modèle 3D est basée sur le modèle 2D, et donc le modèle 3D va hériter de toutes les lacunes (voir le paragraphe précédent) du modèle 2D. Par exemple, pour eux, une petite boite (un cube d'une hauteur de 30 cm par exemple) sera considérée comme un mur (de hauteur 3 m) si la boite se trouve en face du scanner laser.

Remarque : dans tous les travaux que nous avons mentionnés, nous remarquons que la caméra en plus des scanners laser sert à texturer les facettes planes seulement, et n'est pas pour exploitée comme un capteur de mesure. De plus, les modèles possèdent un seul type d'amers (les plans).

2.5.3 Notre Modèle

Définissons le modèle que nous cherchons à construire :

Quels types d'amers utiliser ?

Un environnement structuré possède un nombre considérable de surfaces planes : murs, sol, plafond, portes, etc., auquel on peut ajouter les surfaces planes des meubles qui peuvent se présenter, comme par exemple, tables, armoires, réfrigérateur, etc. Cette présence massive des facettes planes explique le choix des plans comme amers dans notre carte.

En outre, les segments 3D sont eux aussi omniprésents : les intersections des murs avec le sol ou avec le plafond, les coins (intersections entre deux murs), les bords des portes, etc. Donc les segments 3D seront aussi des bons candidats comme amers pour notre carte.

En milieu d'intérieur, on peut trouver des informations colorimétriques (poster, texture sur les murs, peinture, décorations, etc.). Ces informations peuvent être représentées comme des points d'intérêt (Harris, ou SIFT par exemple). Nous pouvons ajouter ces informations comme un troisième type d'amers.

De plus, le but de notre travail est de créer une carte pour le robot pour qu'il s'y déplace, se localise et réalise ses missions. Cette carte sera visualisée par l'homme afin de pouvoir donner des commandes au robot de manière "conviviale" (*user friendly*). Ajouter la texture aux facettes planes sera une étape nécessaire pour que l'utilisateur (l'homme) puisse reconnaître les endroits plus facilement. Par exemple, au lieu d'afficher deux murs comme deux facettes planes (non

colorées), avoir la texture sur ces plans peut aider facilement à distinguer un mur (blanc par exemple) d'une porte marron. Les informations de couleur et de texture seront ajoutées d'une part pour produire des cartes plus lisibles par l'homme, et d'autre part pour rendre l'appariement des facettes (*data association)* plus robuste, comme on le verra au chapitre 5.

Bien évidemment, ceci ne nous permettra pas de modéliser parfaitement tout l'environnement d'intérieur, mais le résultat de notre travail sera considéré comme une étape vers un modèle plus riche. Par exemple, pour les objets, une fois que nous avons une méthode robuste pour les reconnaître et les classifier, nous pourrons a priori, les ajouter à la carte comme amers objets. Ceci sera une des perspectives éventuelles de notre travail.

Avoir plusieurs types d'amers nous amène à construire des cartes **Hétérogènes**. Notre carte hétérogène contiendra des amers plans, amers lignes 3D, et des points d'intérêt. Nous traitons au chapitre 5 la construction de cette carte.

Nous avons déterminé notre cahier des charges, il nous reste à définir les moyens pour le satisfaire au mieux. C'est-à-dire les capteurs que nous devons exploiter pour réaliser ces buts.

En général, pour avoir une carte 3D, il faut des capteurs 3D. Nous décrivons dans la section suivante les différents types de capteurs 3D, et notre choix parmi eux.

2.6 Acquisition 3D

Pour construire un modèle 3D, le robot doit être doté d'un capteur qui fournit des données 3D. Plusieurs capteurs sont utilisés dans ce domaine : la stéréovision, le scanner 3D, et récemment le Swiss Ranger. Le capteur qui fournirait à la fois des mesures 3D précises et des informations chromatiques peu bruitées, serait idéal afin de faire la modélisation de l'environnement : malheureusement il n'existe pas (encore).

2.6.1 Stéréovision

Un des buts de la vision par ordinateur est de représenter la structure tridimensionnelle (3D) de l'espace. La vision stéréoscopique binoculaire utilise deux images prises avec deux caméras. Un modèle géométrique dit sténopé (*Pin Hole*) est considéré pour la caméra. Une étape de calibrage permet de trouver les différents paramètres identifiant un banc stéréo. Ainsi, le calibrage permet de trouver le modèle de projection de chaque caméra et la relation spatiale entre elles. Cette connaissance permet de calculer les coordonnées 3D d'un point à partir de ses deux projections dans les deux images par une simple triangulation. Nous détaillons dans le chapitre 3 l'algorithme de stéréovision en mettant l'accent sur notre contribution dans la méthode de coupure de graphe pour établir la mise en correspondance stéréoscopique.

Nous avons étudié profondement ce type de capteur (chapitre 3), pour plusieurs raisons. Tout d'abord, parce qu'il s'agit d'un capteur pas cher. De plus, la caméra est un capteur riche qui fournit beaucoup d'informations. Ensuite, la stéréovision peut travailler en temps réel. Enfin, avec

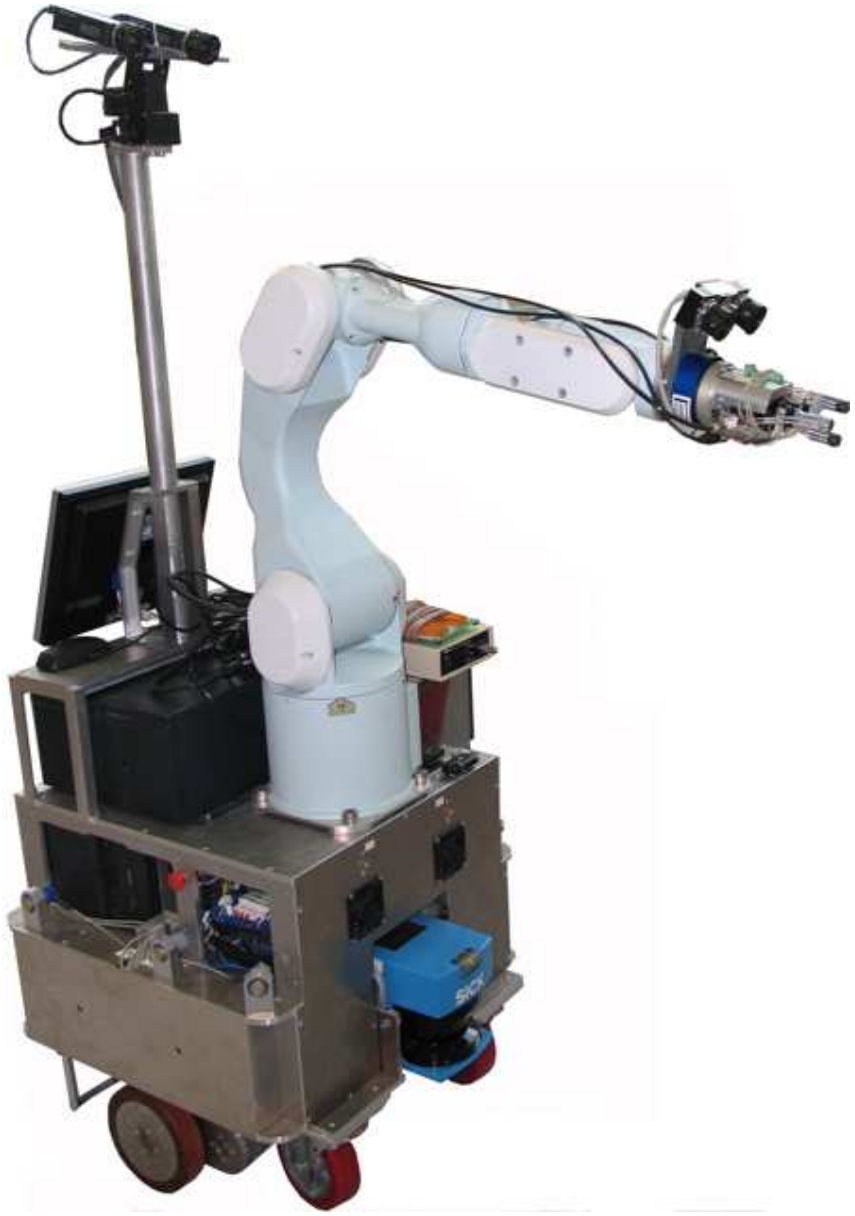


FIG. 2.7 – Le robot mobile Jido

les informations de profondeur données par la stéréovision et les informations photométriques (intensité, couleur, texture, etc.) données par l'image, la stéréovision est, en principe, un capteur idéal pour la modélisation de l'environnement. Il nous a fallu déchanter : en milieu intérieur, les surfaces homogènes (unicolores), le manque de texture, et les motifs répétitifs (carrelage), sont omniprésents, ce qui fait de la stéréovision un capteur peu performant, et même inutilisable pour la modélisation de l'environnement d'intérieur. Pour surmonter ce problème, l'utilisation de la stéréovision active (projection d'une lumière pour créer une texture artificielle) est une solution, mais l'utilisation d'illuminateurs doit être limitée dans un environnement humain car l'homme est gêné par la lumière visible utilisée. Très récemment, des systèmes de stéréovision utilisant des *Near Infrared Light* (NIR) [Guan *et al.*, 2007], ont été exploités en milieu habitable.

2.6.2 Télémètre Laser 3D

Plusieurs solutions sont envisageables pour dépasser les insuffisances des scanner laser 2D. Premièrement, deux scanners laser peuvent être montés avec un angle de 90 degrés entre eux. Cela peut ajouter des informations 3D et permettre de construire une carte tridimensionnelle comme dans les travaux de [Thrun *et al.*, 2003]. Une autre solution est de monter le scanner laser sur un axe pivotant comme dans les travaux de [Weingarten, 2006]. Sur notre robot Jido, le scanner laser LMS 200 est monté sur un axe horizontal motorisé par un stepper. La résolution angulaire du scanner est de 0.5 degré. Bien que le scanner laser donne des données 3D précises, le manque de données photométriques est une lacune considérable de ce capteur. En effet, ce capteur peut donner les mesures 3D d'un obstacle qui se trouve en face de lui, mais il ne peut pas identifier sa nature. Donc le scanner laser tout seul ne permet pas d'aller plus loin dans la construction des cartes sémantiques. L'utilisation conjointe de scanner laser avec une caméra (panoramique ou non) peut aider à surmonter cette lacune.

2.6.3 Swiss Ranger

The Swiss Center for Electronics and Microtechnology (CSEM) a développé un nouveau type de capteur nommé *Swiss Ranger* avec la capacité de produire des données tridimensionnelles denses en temps réel (30 Hz).

Le Swiss Ranger est un capteur qui permet d'acquérir une image 3D, grâce à la mesure du temps de vol d'une onde lumineuse dans le proche infrarouge, en chaque pixel [Lange et Seitz, 2001] : l'onde lumineuse est créée par une batterie de LEDs qui créent un éclairage diffus. Ce capteur rend des informations de distance et de niveau de gris (intensité). La caméra a une grille bidimensionnelle d'éléments photosensibles et une source de lumière modulée.

La portée limitée (jusqu'à 7.5 m), sa sensibilité aux autres sources d'éclairage proche infrarouge (en particulier, le soleil) et la dégradation rapide de la précision avec l'éloignement sont les points faibles de ce capteur. Peut-être ces points seront corrigés dans les futures versions du SwissRanger pour le rendre plus utilisable pour la modélisation 3D de l'environnement, mais pour l'instant, il reste peu exploitable.

2.7 Nos Contributions

- Implémentation d'un algorithme de mise en correspondance stéréoscopique. En fait, cet algorithme est inspiré des travaux de Roy et Cox [Roy et Cox, 1998]. Cette contribution n'est pas originale, mais elle nous a permis de réaliser deux objectifs. Le premier est d'avoir un module fonctionnel de mise en correspondance globale, vu que tous les algorithmes déjà implémentés dans notre laboratoire sont basés sur des critères locaux. Le deuxième est de pouvoir comparer les performances de ce module avec l'algorithme que nous allons développer nous même par la suite.

- Conception et réalisation d'un algorithme de mise en correspondance stéréoscopique basé sur la coupure d'un graphe réduit. Même si nous nous sommes basés sur les travaux de Roy, l'originalité de notre approche est que nous sommes les premiers à travailler sur un graphe réduit. L'idée du graphe réduit est d'éliminer tous les noeuds du graphe sauf les plus potentiels. Cette idée simple et innovante nous a permis d'avoir un graphe plus allégé, et plus facile à manipuler, et surtout l'algorithme est beaucoup plus rapide. Notre algorithme prend quelques secondes de temps de calcul, au lieu de plusieurs minutes avec le graphe complet. De plus, avec notre graphe réduit, nous pouvons traiter des images de taille plus grande que la méthode initiale de Roy, et en même temps utiliser des plages de disparités plus larges. Enfin, avec notre méthode, l'utilisation de la disparité sous-pixellique est possible, alors que ce n'était pas le cas avec la méthode du graphe complet.

- Développement d'un algorithme de segmentation d'image de profondeur pour extraire les facettes planes. On trouve dans la littérature une abondance d'algorithmes de ce type. Mais avoir sous la main un algorithme qui donne de bons résultats n'est pas toujours trivial. En fait, nous considérons cette étape comme apprendre l'alphabet avant d'apprendre à écrire : une phase basique mais indispensable. De plus, notre algorithme a deux différences par rapport aux autres : le choix des paramètres et la méthode d'estimation.

- Implémentation d'un algorithme de SLAM avec des amers plans 3D. Même si nous ne sommes pas les premiers à le faire, c'était une étape nécessaire et basique pour notre travail qui nous a permis d'aller plus loin, notamment pour la construction d'une carte hétérogène

- Fusion de données : nous avons utilisé deux capteurs pour l'acquisition 3D. La calibration des capteurs est une étape vitale avant toute sorte de fusion des données. De plus, dans presque la majorité des travaux, lorsqu'on fusionne les données laser et caméra, cela est fait seulement afin de projeter la texture sur le modèle extrait par les données laser uniquement. Dans notre travail, nous avons proposé une autre méthode pour réaliser la fusion. En fait, nous utilisons les données du scanner laser et de la caméra comme données de mesures. Ainsi, la caméra non seulement permet d'obtenir la texture des facettes planes, mais en plus permet de mesurer la position de droites ou de points. Nous avons pu extraire par fusion de données, les segments 3D dans la scène. De plus, pour des raisons d'optimisation, nous avons interprété ces segments 3D comme étant des segments 2D attachés à un plan 3D (déjà inclus dans la carte). Cette partie est détaillée dans les chapitres 4 et 5.

- Proposition d'une carte hétérogène qui contient à la fois des amers plans, des amers lignes 2D, et des points d'intérêt. De plus, elle pourrait contenir les amers objets. La construction complète de cette carte fera l'objet de travaux futurs : nous fournissons ici des résultats partiels.

- Utilisation des informations d'apparence. Ces informations vont permettre une meilleure appariement des amers plans, et en conséquence, elles consolident la cohérence de la carte. De plus, elles ajoutent de la richesse à la représentation.

- Mise en oeuvre des techniques développées sur le robot. Les données traitées proviennent des capteurs embarqués sur le robot. Le traitement de ces données nous a permis d'obtenir les résultats figurant dans ce manuscrit.

Chapitre 3

Stéréovision et Coupure de Graphe

Sommaire

3.1 Introduction

La stéréovision vise à estimer la position des points dans la scène à partir de deux (ou plusieurs) images de la même scène prises sous différents angles de vue. Utilisée comme un capteur de mesure 3D sur des plateformes robotiques, la stéréovision possède plusieurs avantages. Tout d'abord, c'est un capteur peu cher (par rapport au télémètre laser par exemple). De plus, il est compact (voir par exemple le capteur Sth-MDCS vendu par la compagnie Videre [6]). Ensuite, il peut être adapté pour travailler sous les contraintes temps réel (voir de la même compagnie Videre, la librairie stéréo, qui donne à 30 Hz des images 3D de tailles 640 x 480). En revanche, nous savons aussi qu'il s'agit d'un capteur assez myope, peu précis pour des objets lointains et surtout, que les méthodes temps réel de stéréovision peuvent aussi produire de nombreux artefacts.

L'algorithme de stéréovision est composé de plusieurs étapes. La première étape est le calibrage, qui peut être hors ligne ou en ligne (autocalibration) ; elle permet d'estimer les paramètres intrinsèques des deux caméras, ainsi que la matrice de transformation inter-caméras. L'étape suivante est l'appariement ou la mise en correspondance, qui consiste à trouver dans les deux images les motifs homologues, c'est-à-dire, les projections du même objet de la scène dans les deux images. Cette étape produit une image de disparité : la disparité est définie comme la variation de position de la projection d'un même objet sur les deux images. Une fois l'image de disparité calculée, la troisième étape de reconstruction 3D peut être achevée par une triangulation.

Malgré les efforts importants consacrés aux problèmes de la vision stéréoscopique pendant ces dernières décennies, il n'existe pas de nos jours un capteur stéréoscopique fiable et efficace pour des applications temps réel, notamment en Robotique. Pourtant, dans les dernières années, les techniques de coupure de graphe ont donné des meilleurs résultats. Les méthodes de coupure de graphe transforment le problème de l'appariement en un problème de minimisation d'une fonction d'énergie globale qui représente le coût total de la mise en correspondance pour tous les pixels de l'image. Malheureusement, ces techniques sont très gourmandes pour les ressources informatiques en matière de mémoire (RAM) et de temps de calcul. Nous présentons dans ce chapitre notre contribution pour rapprocher les techniques de la coupure de graphe vers les contraintes temps réels. Nous verrons que les résultats obtenus sont très encourageants même

[6] http ://www.videredesign.com

si nous ne les avons pas utilisés dans la suite de nos travaux sur la construction du modèle de l'environnement.

En écartant pour l'instant les contraintes temps réel, nous évaluons dans ce chapitre, une méthode globale de mise en correspondance, exploitant la coupure de graphe, plus populaire sous son petit nom anglais *Graph Cuts*. Nous commençons par donner un petit rappel du modèle géométrique de la caméra. Ce rappel va non seulement, nous faciliter la présentation de ce chapitre, mais aussi nous aider à développer les équations pour la fusion de données au chapitre 5. Dans la section 3.3, nous rappelons ce qu'est la stéréovision et nous détaillons l'état de l'art de la mise en correspondance. La section 3.4 est consacrée aux définitions de graphe, de la coupure minimale, du flot, et des différentes méthodes pour résoudre le problème de coupure minimale. En section 3.5.1, nous donnons la formulation générale du problème d'étiquetage, avant de l'instancier pour la mise en correspondance en stéréovision en section 3.5.2. En sections 3.5.3 et 3.5.4, nous décrivons deux méthodes que nous avons évaluées : coupure sur un graphe complet, puis coupure sur un graphe de taille réduite. Nous décrivons notre implémentation de ces méthodes, ainsi que les résultats expérimentaux en section 3.6.

3.2 Le Modèle d'une Caméra

Nous commençons par présenter un modèle parfait d'une caméra, ensuite nous ajoutons le modèle de la distorsion radiale, et les méthodes de détermination des différents paramètres de ce modèle.

3.2.1 Le Modèle Géométrique d'une Caméra

Dans notre travail, nous adaptons le modèle de caméra dit sténopé ou "trou d'épingle" (*pinhole camera model*), présenté en figure 3.1. Le modèle sténopé est le plus fréquemment utilisé en vision par ordinateur, car il permet de modéliser de manière fine la plupart des capteurs projectifs et de simplifier la mise en équations [Horaud et Monga, 1995]. Ce modèle est fondé sur l'hypothèse que tous les rayons passent par un seul point : le centre optique de la caméra O_c. Cette hypothèse est d'autant plus respectée pour les focales de faibles dimensions à condition de prendre en compte aussi les distorsions optiques.

De ce fait, avec ce modèle, une caméra réalise une projection perspective centrale. Ainsi, un pixel de l'image est la projection d'un point de l'espace dans le plan image. Cette projection peut être modélisée à l'aide de deux transformations :

- Une projection du point de l'espace 3D en un point dans le plan image (2D).
- Une transformation du repère métrique lié à la caméra au repère image (pixel).

Pour mettre en équation ces opérations, on définit les repères suivants, voir figure 3.1 :

- $\mathcal{R}_w$: Un repère 3D appelé repère monde $\mathbf{O_w}(\mathbf{X_w}, \mathbf{Y_w}, \mathbf{Z_w})$, un point de l'espace 3D sera représenté par ses coordonnées $P_w = (x_w, y_w, z_w, 1)^T$.

- $\mathcal{R}_c$: Un repère 3D lié à la caméra dont l'origine est le centre optique de la caméra, et où l'axe Z_c est confondu avec l'axe optique, et les axes X_c et Y_c sont parallèles aux lignes et colonnes du plan image. Le point P sera donné par ses coordonnées $P_c = (x_c, y_c, z_c, 1)^T$.

- $\mathcal{R}_I$: Un repère 2D lié au plan image, dont l'unité est le pixel. Ce plan est situé à une distance f dans le repère caméra, où f est la distance focale de la caméra : les coordonnées entiers (u, v) sont les indices des pixels de l'image.

Un point P se projette dans le plan image le long d'une droite passant par P et O_c (rayon optique). Les coordonnées de sa projection notées (x_i, y_i, z_i) exprimées dans le repère caméra sont données par :

$$\begin{cases} x_i &= f\frac{x_c}{z_c} \\ y_i &= f\frac{y_c}{z_c} \\ z_i &= f \end{cases} \tag{3.1}$$

Les points images sont mesurés en pixels dans le repère $\mathcal{R}_I$ associé à l'image. Le passage du repère caméra au repère image demande la définition des paramètres dits intrinsèques de la caméra. Ces paramètres sont u_0, v_0 donnant les coordonnées du point principal (projection du centre optique sur le plan image) dans le repère image (en pixels), et k_u, k_v définissant les facteurs d'échelle vertical (pixels/mm) et horizontal respectivement. La relation entre les coordonnées métriques et coordonnées en pixel du point de projection dans le plan image est :

$$\begin{cases} u &= k_u x_i + u_0 \\ v &= k_v y_i + v_0 \end{cases} \tag{3.2}$$

Les signes de k_u et k_v dépendent du choix fait pour les axes X_c et Y_c du repère caméra. Alors :

$$\begin{cases} u &= k_u f\frac{x_c}{z_c} + u_0 \\ v &= k_v f\frac{x_c}{z_c} + v_0 \end{cases} \tag{3.3}$$

ou sous forme matricielle :

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \tag{3.4}$$

avec $\alpha_u = k_u f$, $\alpha_v = k_v f$, et λ est un facteur d'échelle. La matrice $\mathbf{I_c}$ est appelée la matrice des paramètres intrinsèques.

$$\mathbf{I_c} = \begin{bmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.5}$$

Soit $\mathbf{E}$ la transformation homogène qui amène du repère caméra au repère monde, elle est composée d'une rotation $\mathbf{R}_{c,w}$ et d'une translation $\mathbf{t}_{c,w}$, on écrit :

$$\mathbf{P_c} = \mathbf{E}\,\mathbf{P_w} \tag{3.6}$$

avec :

$$\mathbf{E} = \begin{bmatrix} \mathbf{R_{c,w}} & \mathbf{t_{c,w}} \\ 0 & 1 \end{bmatrix} \tag{3.7}$$

$\mathbf{E}$ est appelée la matrice des paramètres extrinsèques.

3.2.2 Modèle de Distorsion Radiale

Le modèle sténopé présenté ci-dessus ne prend pas en considération les aberrations géométriques des optiques. En effet, l'objectif est composé en général de plusieurs lentilles, dont les caractéristiques sont calculées pour appliquer au mieux une projection perspective directe. Mais dans les optiques de faible coût, ou avec une focale très courte (grand angle), le modèle sténopé n'est pas respecté et les rayons optiques subissent des déviations dans l'objectif, ce qui provoque des distorsions dans l'image.

Les distorsions géométriques sont de plusieurs types : le décentrage et les distorsions radiales et tangentielles. Les distorsions de décentrage dérivent d'un mauvais alignement des lentilles à l'intérieur de l'objectif monté sur la caméra ou de la non orthogonalité du plan image. La distorsion radiale provient d'un défaut de courbure sur les lentilles de l'objectif. Elles sont donc

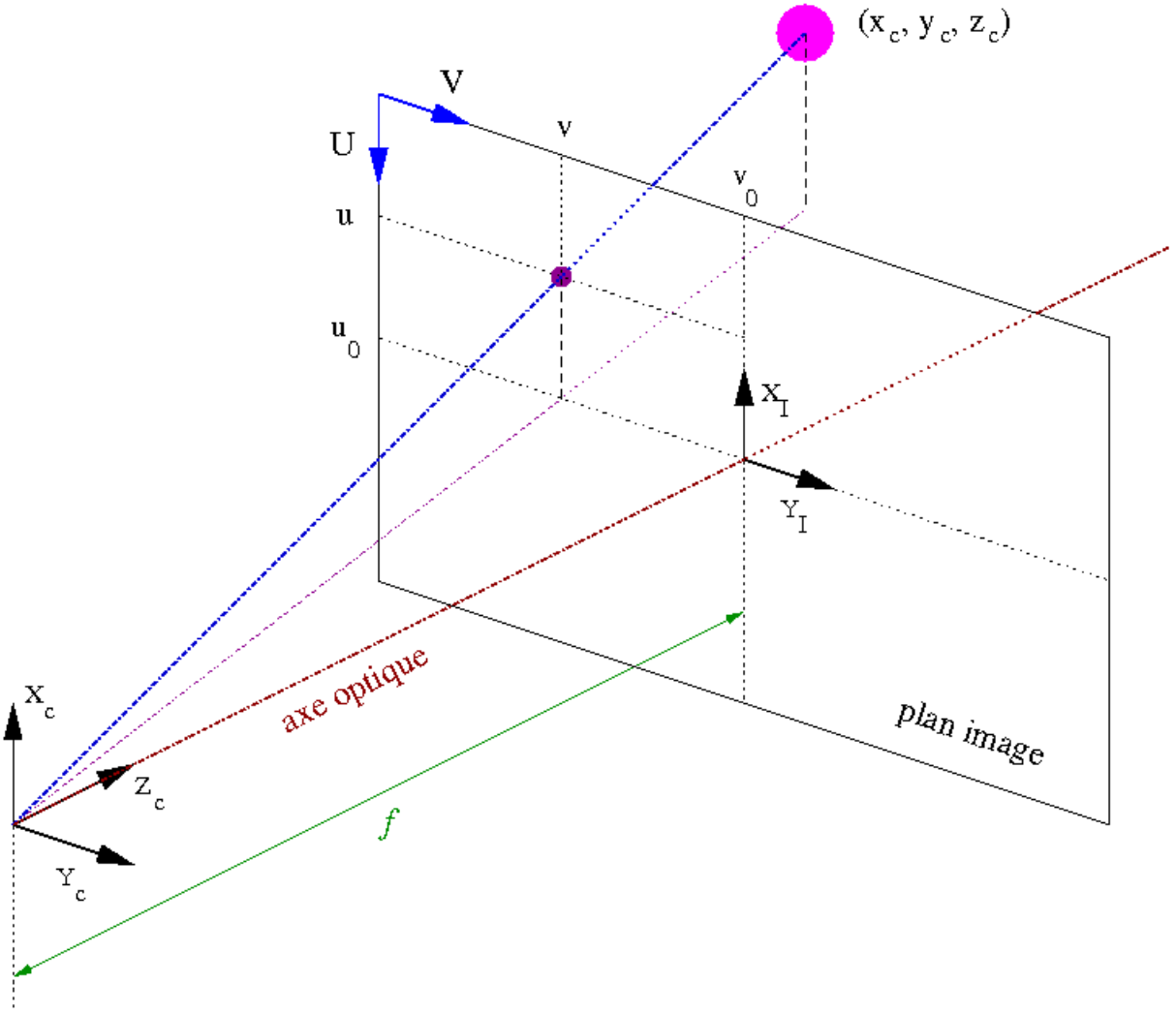


FIG. 3.1 – Le modèle géométrique d'une caméra

d'autant plus fortes que la courbure de l'objectif est importante (grand angle). Cette distorsion se manifeste dans le plan image, comme une translation le long du rayon (voir figure 3.2). La distorsion prismatique tangentielle provient d'un défaut de parallélisme entre le plan image et les plans des lentilles sur l'objectif. Cela provoque en un pixel, une translation le long de la tangente au cercle centré sur la projection du centre optique et passant par ce pixel. En général, pour une caméra standard et pour une application ne nécessitant pas une précision importante (robotique mobile en général), les distorsions de décentrage et tangentielle sont négligeables.

La figure 3.3 illustre une image initiale non corrigée. On peut remarquer clairement dans cette image que les lignes droites (bords des portes) sont curvilignes, en particulier près des coins de l'image. La figure 3.4 montre la même image après correction de la distorsion radiale.

Nous introduisons donc uniquement les paramètres décrivant les distorsions radiales. Sur le plan image, en coordonnées métriques, les coordonnées de la projection d'un point P sur le plan image exprimé dans le repère caméra $\mathcal{R}_c$ sont idéalement données par (x_i, y_i). Due à la distorsion

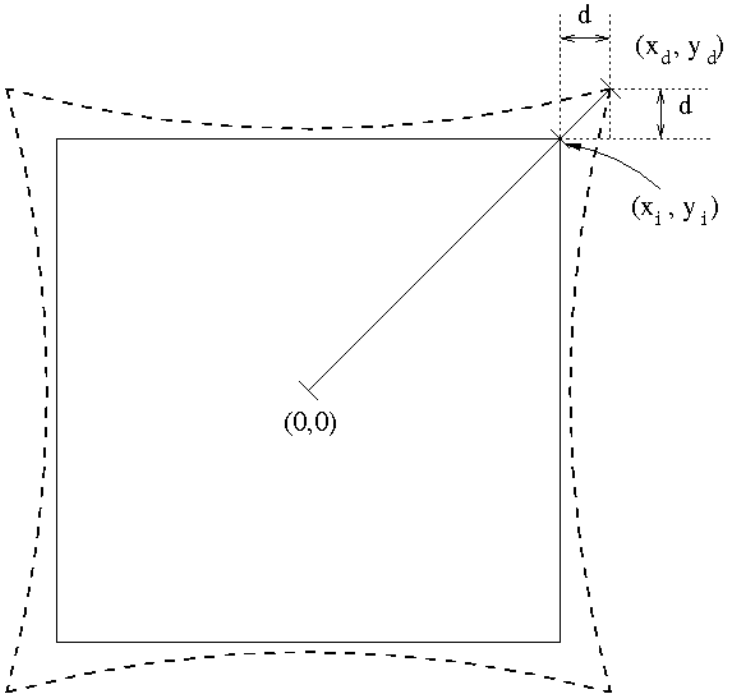


FIG. 3.2 – Distorsion Radiale

radiale, les coordonnées deviennent (x_d, y_d) (voir la figure 3.2). La relation entre les coordonnées réelles et idéales peut être exprimée sous la forme :

$$\begin{cases} x_d = x_i \left[1 + k_1\ r^2 + k_2\ r^4 + k_3\ r^6\right] \\ y_d = y_i \left[1 + k_1\ r^2 + k_2\ r^4 + k_3\ r^6\right] \end{cases} \tag{3.8}$$

où :

- $k_1, k_2, k_3, \ldots$: les coefficients de la distorsion radiale sur les coordonnées métriques.
- $r^2 = x_i^2 + y_i^2$: la distance au point principal (projection du centre optique).

Le nombre de coefficients $(k_1, k_2, k_3, \ldots)$ nécessaires pour décrire une caméra donnée peut être déterminé lors de la procédure de calibrage.

3.2.3 Calibrage

Dès lors que l'on souhaite utiliser une caméra pour obtenir des informations métriques, il est nécessaire de la calibrer. Le calibrage d'une seule caméra (pour les applications monoculaires) tente à estimer ses paramètres intrinsèques ainsi que sa position par rapport au référentiel du monde. Pour un capteur stéréoscopique, trouver sa configuration revient à calibrer les deux

FIG. 3.3 – Une image sans correction de la distorsion radiale

FIG. 3.4 – L'image après correction de la distorsion radiale

caméras (paramètres intrinsèques et coefficients de distorsion de chacune des caméras) et de trouver la relation spatiale (la translation et la rotation) entre elles.

De nombreux travaux ont été menés concernant le calibrage qui peut être hors ligne ou en ligne (autocalibration) d'un capteur stéréoscopique. La première méthode nécessite une mire de calibrage de géométrie parfaitement connue pour calibrer hors-ligne le système de vision [Faugeras, 1993], [Horaud et Monga, 1995]. La mire sous forme d'échiquier est fréquemment utilisée. Dans notre travail, nous utilisons la boite à outils *Camera Calibration Toolbox for Matlab*, pour calibrer les caméras et le banc stéréo.

La connaissance des paramètres de calibrage permet de calculer les coordonnées 3D d'un point à partir de ses deux projections dans les deux images par une simple triangulation.

3.2.4 La géométrie épipolaire et Rectification

Pour un système de vision à deux caméras, l'axe passant par les deux centres optiques est appelé **baseline**. Un plan épipolaire (π) lié à un point M dans l'espace est défini comme le plan passant par les deux centres optiques des caméras et par un point M. En fait, ce plan contient bien évidemment la baseline.

La géométrie épipolaire est la géométrie étudiant les propriétés liées à l'intersection des plans images avec l'ensemble des plans épipolaires. La figure 3.5 illustre cette géométrie. Les projections d'un point M dans les plans images sont m et $m^{'}$. Les points M, m, $m^{'}$, C, $C^{'}$ et la baseline sont tous coplanaires.

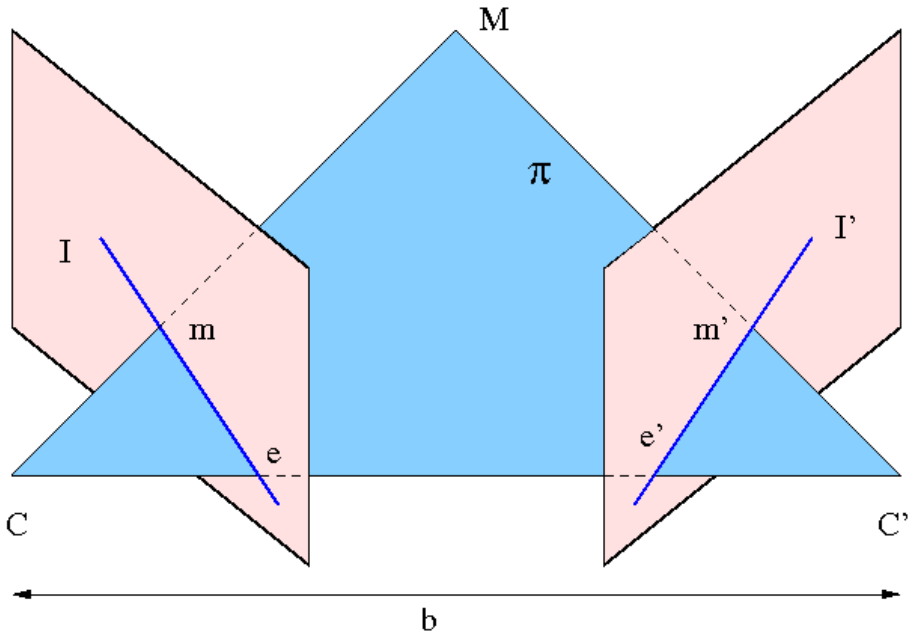


FIG. 3.5 – Illustration de la géométrie épipolaire

L'intersection de la baseline avec chaque plan image est appelée épipole. On associe à un

point m de la première image une droite $L^{'}$ dite épipolaire, qui passe par le point correspondant $m^{'}$ et par l'épipole $e^{'}$ de la deuxième caméra. En effet, si la position du point M varie sur le même rayon optique passant par le centre optique de la première caméra et par le point M, sa projection sur le premier plan image (m) restera invariante, alors que sa projection sur l'autre plan image $(m^{'})$ évoluera sur la droite épipolaire $L^{'}$. Il existe donc une relation entre les points d'une image et les primitives épipolaires. Cette relation est indépendante de la structure de la scène et ne dépend que des paramètres intrinsèques et extrinsèques des caméras. Cette relation est appelée la contrainte épipolaire [Faugeras, 1993].

Rectification de l'image

En stéréovision, la rectification d'une paire d'images stéréoscopiques permet de se ramener à une géométrie épipolaire simple où les droites épipolaires sont parallèles aux lignes des images. Ceci permet de simplifier considérablement le processus de mise en correspondance. En effet, la rectification stéréoscopique permet de restreindre l'espace de recherche des appariements de manière très importante : on passe d'un espace de recherche initialement bidimensionnel à un espace monodimensionnel où la recherche se fait le long d'une ligne épipolaire.

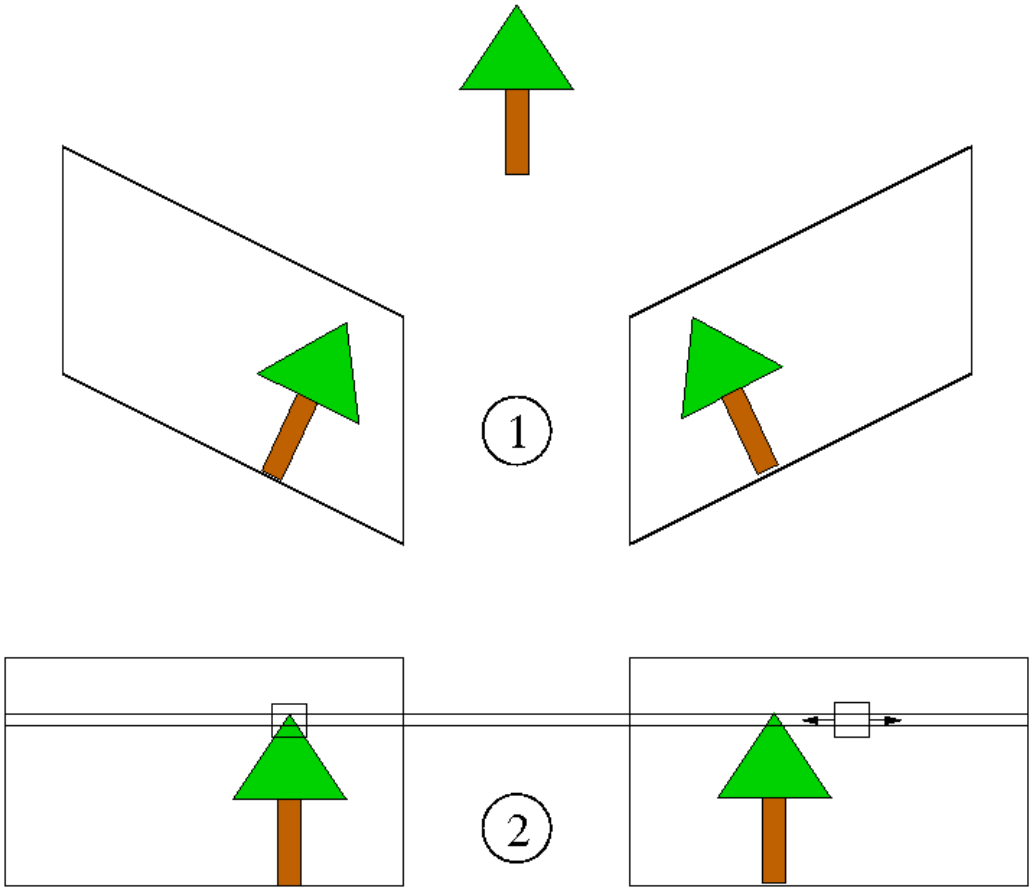


FIG. 3.6 – Illustration de la rectification d'images, (1) les images initiales, et (2) les images rectifiées

La figure 3.6 illustre un exemple synthétique de rectification d'images. On peut remarquer sur cet exemple que pour un point de l'image gauche, la recherche d'un point correspondant dans l'image droite est réduite à la recherche sur une droite horizontale située à la même ordonnée.

Dans la suite de notre travail, nous ne traitons que des paires d'images déjà rectifiées.

3.3 Mise en Correspondance Stéréoscopique

L'objectif principal de la vision par ordinateur depuis longtemps est de construire un capteur visuel capable d'assumer le rôle de l'oeil humain. Le principe de la stéréovision ou de la vision stéréoscopique consiste à utiliser plusieurs vues d'une même scène et d'en reconstituer la géométrie tridimensionnelle grâce à la disparité des images. La stéréovision est un moyen de reconstruire le relief à partir de deux images prises au même moment de deux points de vue différents, ce qui imite la vision stéréoscopique de l'homme.

La reconstruction stéréoscopique est fondée sur l'aptitude à retrouver dans chaque image la projection d'un même objet dans la scène. En effet, dans la vision stéréoscopique, l'information de profondeur d'un objet est liée d'une part à la disparité entre ses projections dans les deux images et d'autre part, à la position relative des deux caméras. La reconstruction stéréoscopique pose ainsi deux problèmes distincts. Le premier est le calcul de la disparité qui est attachée aux problèmes de mise en correspondance. L'autre problème est la capacité d'inverser le problème de la géométrie projective, c'est à dire, la reconstruction 3D ou la capacité de retrouver les informations tridimensionnelles à partir de la connaissance de la position relative des capteurs d'images et de la disparité. Les études de Faugeras [Faugeras, 1993] sur la géométrie projective ont fondé une base solide à la reconstruction tridimensionnelle. En ce qui concerne les problèmes d'appariement, aucune méthode ne s'est révélée suffisamment fiable, robuste et efficace pour permettre une utilisation simple de la stéréovision comme un capteur de mesure de profondeur.

La mise en correspondance stéréoscopique est un des problèmes les plus traités en vision par ordinateur depuis presque un demi siècle [Julesz, 1962], et la littérature la concernant est abondante. Le problème d'appariement ou de mise en correspondance consiste à retrouver dans les images gauche et droite les primitives homologues, c'est-à-dire, les primitives qui sont les projections de la même entité de la scène. Ces primitives peuvent être les pixels de l'image : on parle alors de stéréo dense ou *pixel-based stereo*. Elles peuvent aussi être des points d'intérêt (comme les points de Harris [Harris et Stevens, 1988]) ou encore des segments dans le cas de stéréo segment : l'algorithme sera alors de type stéréo éparse. Par la suite, nous ne nous intéresserons qu'à la stéréo dense. Pour l'appariement, on peut caractériser deux types de contrainte : les contraintes locales qui s'appliquent sur un pixel d'intérêt et un certain nombre de pixels dans ses voisinages, et les contraintes globales qui portent sur tous les pixels portés sur une ligne de l'image (ou sur une ligne épipolaire en absence de rectification) ou sur toute l'image. Les méthodes utilisant des contraintes locales, appelées aussi méthodes locales, se focalisent sur l'amélioration des techniques de sélection du correspondant afin d'augmenter les caractères discriminants des éléments remarquables. Les méthodes locales cherchent le meilleur correspondant pour un pixel donné sans prendre en compte les appariements des pixels voisins. Les méthodes utilisant des contraintes globales, aussi appelées méthodes globales, tentent de définir un modèle global de la

scène observée et de minimiser une fonction de coût globale. Les méthodes globales cherchent à trouver d'un seul coup, les correspondants pour tous les pixels d'une même ligne ou pour tous les pixels de l'image.

De manière générale, le problème d'appariement revient à un problème de minimisation. L'approche locale tente de minimiser plusieurs fonctions d'énergie, représentant les coûts locaux de chaque appariement supposé indépendant les uns des autres. L'approche globale essaie de minimiser une seule fonction d'énergie, représentant le coût global de tous les appariements entre les primitives des images : un tel coût global est basé sur les coûts locaux de tous les appariements potentiels, ainsi que les compatibilités entre ces appariements. En français, on peut trouver, parmi d'autres, deux états de l'art qui sont très bien détaillés, celui décrit par Sylvie CHAMBON dans sa thèse [Chambon, 2005], et celui de Christophe Rabaud détaillé dans sa thèse aussi [Rabaud, 2005]. Une récente étude et taxonomie des algorithmes de mise en correspondance est fournie par Scharstein et Szeliski [Scharstein et Szeliski, 2002]. Les auteurs distinguent quatre éléments pour caractériser les méthodes de mise en correspondance :

- Le coût local d'une correspondance.
- La zone d'agrégation considérée lors du calcul du coût local.
- La méthode d'optimisation exploitée pour minimiser les coûts.
- L'affinement des résultats.

Les méthodes développées et exploitées au LAAS depuis 1995 [Grandjean et Lasserre, 1995], sont basées sur les travaux de Faugeras [Faugeras *et al.*, 1993]. Ces méthodes sont adaptées à des applications robotiques, pour lesquelles les contraintes temps réel sont essentielles. Elles peuvent être classées dans les algorithmes locaux car elles ne possèdent pas d'étape d'optimisation globale. Les images sont rectifiées au préalable, afin de limiter la zone de recherche, dans l'image droite, du pixel correspondant à un pixel donné de l'image gauche. Le coût local est basé sur une mesure de ressemblance inter-pixels : la mesure la plus classique est un score de corrélation entre des fenêtres centrées sur les pixels potentiellement appariés.

3.3.1 Méthodes Locales

Les méthodes locales cherchent le meilleur correspondant pour un pixel donné sans prendre en compte les appariements des pixels voisins. Le coût d'appariement de deux pixels est fondé sur une mesure de similarité de la fonction d'éclairement local. Selon Marr et Poggio [Marr et Poggio, 1977], les projections d'un même point tridimensionnel devraient avoir des intensités semblables dans les deux images. En effet, le modèle Lambertien [Horn, 1986] présume que la surface des objets reflète la lumière uniformément dans toutes les directions. En utilisant ce modèle, on peut faire l'hypothèse que les pixels correspondants se ressemblent, mais surtout que leurs voisinages se ressemblent aussi, d'un point de vue photométrique. Un coefficient de corrélation donne une mesure de ressemblance entre deux ensembles de données. La méthode d'appariement essaye de trouver le correspondant p2 dans l'image droite d'un point p1 qui se trouve dans l'image gauche.

La mesure de corrélation utilise l'information donnée par p1 et p2, mais aussi celle fournie par leur voisinage respectif. Le pixel considéré et son voisinage dans l'image gauche constituent un premier ensemble de données, et un pixel et son voisinage dans l'image droite constituent un deuxième ensemble de données. Un score de corrélation évalue la ressemblance entre ces deux ensembles.

Plusieurs approches locales utilisent des fenêtres de comparaison contenant le voisinage du pixel considéré. La ressemblance entre deux fenêtres est obtenue par une mesure statistique de la distance entre les deux fonctions d'illuminations échantillonnées. Parmi les mesures les plus couramment utilisées on peut trouver : la somme des écarts quadratiques (Sum of Squared Differences ou SSD) [Cox *et al.*, 1996], la somme des écarts absolus (Sum of Absolute Differences ou SAD) [Hirschmüller, 2001], la corrélation croisée centrée et normalisée (Zero-mean Normalized Cross-Correlation ZNCC) [Chen et Medioni, 1999], [Sára, 2002], etc.

Les méthodes locales de mise en correspondance de pixels sont exposées à plusieurs sources de défaillance, en particulier les occultations, les imperfections de réflection et les fluctuations d'illumination entre les différentes vues. Les variations de niveau de gris peuvent être causées par plusieurs sources : aléas lumineux, échantillonnage de l'espace, quantification du niveau de gris, sensibilités différentes entre les caméras utilisées. De plus, les artefacts qui se produisent à cause de l'échantillonnage de l'image peuvent troubler la mesure de ressemblance.

Les formes répétitives et le manque de texture dans les environnements d'intérieur font partie des facteurs d'échec les plus redoutables pour les algorithmes de stéréo corrélation. En figure 3.7, nous présentons, pour une scène de couloir carrelé, l'image gauche rectifiée. La figure 3.8 représente l'image de disparité (les pixels blancs sont non appariés), et la figure 3.9 illustre l'image des points 3D en vue de face. Nous constatons d'abord l'absence d'appariements sur les zones uniformes (murs). De plus, une région de disparité erronée sur le sol proche, qui sera reconstruite en 3D à près de deux mètres sous le niveau du sol. Cet exemple illustre deux faiblesses connues de la stéréovision : dans les grandes surfaces uniformes, le nombre des points appariés est très limité. De plus, l'algorithme affronte de grands problèmes de faux appariements lorsque la scène possède des répétitions dans la texture, comme c'est le cas pour un sol carrelé. Ce problème de motifs répétitifs a fait l'objet d'une étude par Dima [Dima, 2002].

3.3.2 Méthodes Globales

Les méthodes globales tentent de définir un modèle global de la scène observée et de minimiser une fonction de coût global. Les correspondants pour tous les pixels d'une ligne ou de l'image sont retrouvés en même temps. Dans une méthode globale, l'appariement d'un pixel dans l'image gauche avec un pixel dans l'image droite ne dépend pas seulement des données photométriques de leurs voisinages, mais aussi des appariements de leurs voisinages. Ainsi, l'appariement d'un pixel de l'image gauche avec un autre dans l'image droite influence les appariements des pixels voisins. Cette influence est modélisée par un modèle de régularisation (contrainte) de l'ensemble des appariements. Certaines méthodes se basent seulement sur la contrainte épipolaire pour transformer ce problème bidimensionnel en un problème monodimensionnel [Belhumeur, 1996], [Cox, 1992]. Tandis que d'autres méthodes abordent une approche bidimensionnelle en utilisant en plus de la contrainte épipolaire, des relations entre lignes [Boykov *et al.*, 1998], [Ishikawa et

FIG. 3.7 – Un cas difficile pour la stéréo-corrélation : image gauche rectifiée

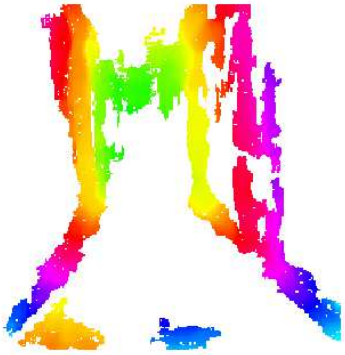


FIG. 3.8 – Un cas difficile pour la stéréo-corrélation : image de disparité (fausse couleur). Deux zones appariées sur le sol : la première est en bleu (appariements corrects), l'autre est en orange (appariements erronés). En fait la couleur orange correspond à la disparité au fond de l'image (les points les plus lointains)

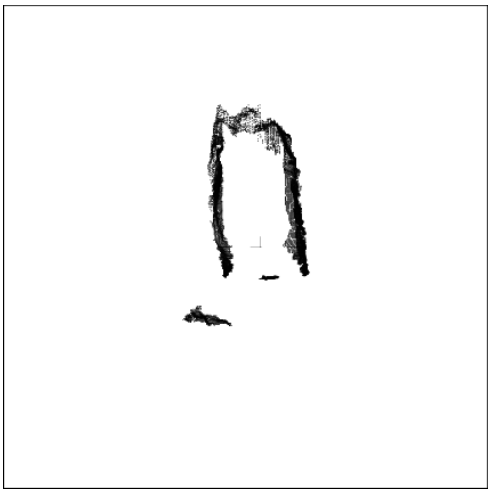


FIG. 3.9 – Un cas difficile pour la stéréo-corrélation : 3D vu de face

Geiger, 1998].

Le but de la régularisation globale est de réduire la sensibilité de la mise en correspondance aux ambiguïtés apportées par les occultations, une faible texture locale ou les fluctuations d'illumination. Cette amélioration nécessite forcément un prix à payer qui est l'augmentation de la complexité de l'algorithme, donc le temps de calcul, à quoi s'ajoute un effet secondaire de lissage de l'image de disparité, effet dû à la régularisation.

Nous détaillons en particulier deux types de méthodes globales : la programmation dynamique et la coupure de graphe.

Programmation Dynamique

La programmation dynamique, inventée par Richard Bellman [Bellman, 1957], permet de résoudre des problèmes d'optimisation dont la fonction objectif se décrit comme la somme de fonctions monotones non-décroissantes des ressources. En pratique, cela veut dire qu'on peut inférer la solution optimale d'un problème en se basant sur une solution optimale d'un sous problème. La programmation dynamique appliquée à la mise en correspondance stéréoscopique cherche un chemin de coût minimal à travers une matrice composée de tous les appariements possibles. Pour limiter la complexité de l'optimisation, cette technique est appliquée entre les deux séquences de pixels sur les droites épipolaires conjuguées. Ainsi, l'appariement stéréoscopique est transformé en problème de mise en correspondance pour tous les pixels d'une ligne de la première image avec ceux de la ligne épipolaire correspondante dans la deuxième image [Ohta et Kanade, 1985].

Pour obtenir un coût du chemin global égale à la somme des coûts des chemins partiels, il est indispensable d'utiliser des coûts additifs. On définit le coût local pour chaque point dans l'espace de recherche comme le coût de la mise en correspondance locale (SAD, SSD, etc.). Pour introduire les occultations, on relie un groupe de pixels dans une image à un seul pixel dans l'autre image et on pénalise cette relation par un coût d'occultation appliqué au coût global du chemin [Ohta et Kanade, 1985], [Cox *et al.*, 1996]. Cette formulation présente plusieurs inconvénients comme la sensibilité au choix du coût d'occultation, la difficulté de garder une cohérence inter-lignes de recherche [Bobick et Intille, 1999], et la possibilité de ne pas pouvoir satisfaire les contraintes d'ordre et de continuité.

La programmation dynamique pourrait aider à trouver une solution pour la mise en correspondance dans les zones localement peu texturées, et d'écarter les problèmes liés aux occultations. Pourtant, cette méthode possède des points faibles, qui sont essentiellement la complexité du calcul et la possibilité de propagation d'une erreur locale tout le long de la ligne de recherche, ce qui pourrait perturber une partie des appariements corrects pouvant être obtenus par des algorithmes locaux.

Coupure de Graphe

La majorité des méthodes utilisant la programmation dynamique pour la mise en correspondance tentent d'apparier les pixels appartenant à une ligne épipolaire dans les deux images sans prendre en compte la cohérence inter-lignes. Elles n'utilisent pas les contraintes bidimensionnelles du problème. Pour surmonter ces insuffisances et afin de prendre en compte les contraintes de continuité à deux dimensions lors de la mise en correspondance, une solution est introduite par l'utilisation de la théorie des graphes. Initialement, la coupure de graphe appliquée à la stéréovision a été proposée par Roy et Cox [Roy et Cox, 1998], puis reformalisée par Veksler qui a renormalisé le problème en mettant en évidence la fonction d'énergie à minimiser [Veksler, 1999]. Ensuite, Kolmogorov et Zabih ont contribué à introduire les méthodes itératives [Kolmogorov et Zabih, 2001], [Kolmogorov et Zabih, 2002a], [Kolmogorov et Zabih, 2002b].

La première méthode globale fondée sur la coupure de graphe pour résoudre la mise en correspondance en stéréovision, a été introduite par Roy et Cox [Roy et Cox, 1998], [Roy, 1999]. En partant de la formulation 1D de la contrainte d'ordre utilisée par la méthode de programmation dynamique appliquée uniquement et séparément sur chaque ligne des images, Roy a essayé de trouver une formulation 2D plus générale de cette contrainte pour l'appliquer sur toutes les lignes à la fois. Il a proposé une *contrainte de cohérence locale* qui suggère que la fonction de disparité est localement continue, ce qui veut dire que les pixels proches dans toutes les directions ont des disparités similaires. Il prétend que l'avantage de cette contrainte est qu'elle relie non seulement les pixels voisins le long d'une ligne épipolaire, mais aussi les pixels à travers les lignes épipolaires. Roy intègre cette contrainte de cohérence locale avec une contrainte de ressemblance qui dépend de la variation des intensités des pixels appariés. Dans le cas de deux caméras, le coût d'appariement est la différence au carré des intensités (sans prise en compte d'un voisinage).

L'étape suivante de la méthode proposée par Roy et Cox, est de résoudre la fonction de disparité optimale sur toute l'image. Ceci peut être visualisé comme une maille 3D composée de plans, eux-mêmes constitués d'une image de noeuds. Il existe un plan pour chaque niveau de disparité, et chaque noeud représente un appariement entre deux pixels dans les images originales.

La maille 3D est converti en un graphe de flot maximal en connectant chaque noeud avec ses quatre voisins dans le plan par des arêtes dites d'occultation, et avec les deux noeuds dans les plans voisins avec des arêtes dites de disparité. Toutes ces arêtes sont non orientées. On ajoute deux noeuds spéciaux : une **source** connectée à tous les noeuds du plan à disparité minimale, et un **puits** connecté à tous les noeuds du plan à disparité maximale. Le poids associé à une arête de disparité est la valeur moyenne des coûts d'appariement des noeuds. Pour les arêtes d'occultation, le poids est multiplié par une constante pour contrôler le lissage de la fonction de disparité optimale. Une coupure minimale de ce graphe sépare les noeuds en deux sous-ensembles : la fonction de disparité optimale peut être construite par l'affectation à chaque pixel, de la valeur de disparité la plus grande pour laquelle le noeud correspondant dans le graphe est connecté à la source.

Ishikawa et Geiger [Ishikawa et Geiger, 1998] ont montré que la méthode de Roy détaillée ci-dessus peut traiter seulement des fonctions convexes. Ainsi, elle peut seulement prendre en compte des pénalités linéaires sur les discontinuités, ce qui peut donner du flou dans l'image 3D du fait du sur-lissage des discontinuités. Inspirés par le travail de Roy, Ishikawa et Geiger

proposent un graphe à arêtes orientées, donc il est possible avec cette formulation de garantir les contraintes d'unicité et d'ordre. Pour inclure les discontinuités et les occultations, leur graphe comporte des contraintes géométriques qui relient les discontinuités de disparité le long de la ligne épipolaire dans une image et une région occultée dans l'autre image. Le graphe est composé d'une source s et d'un puits t, et des noeuds représentent tous les appariements possibles entre les pixels d'une même ligne r dans les deux images. Ce graphe encourage la régularité de surface le long des lignes épipolaires. La surface de profondeur est obtenue aussi par une coupe minimale. Notons que leur méthode donne une image 3D imprécise pour les discontinuités à cause des pénalités linéaires.

Algorithmes d'optimisation sous optimales

Boykov, Veksler et Zabih [Boykov *et al.*, 1998], [Boykov *et al.*, 1999] ont proposé une autre approche pour résoudre la mise en correspondance par coupure de graphe. Ils ont montré que le problème de mise en correspondance peut être formulé par un *Markov Random Field* (MRF) qui est susceptible de maintenir les discontinuités. Ils ont montré ensuite que l'estimée MAP (Maximum A Priori) d'un tel MRF, peut être obtenue par une coupure minimale à multi-voies, en utilisant le flot maximal. L'avantage de cette méthode est qu'elle permet des pénalités non linéaires pour les discontinuités, ce qui donne des cartes de disparité plus précises notamment près des bords des objets. Comme le problème général de coupure minimale multi-voies est NP-complet (voir les travaux de Dahlhaus et al. [Dahlhaus *et al.*, 1994]), Boykov et al. ont choisi d'introduire un algorithme approximatif, qui peut résoudre itérativement quelques sous-problèmes jusqu'à convergence. Kolmogorov [Kolmogorov et Zabih, 2001] a continué les travaux de Boykov en essayant d'améliorer la fonction d'énergie pour représenter plus explicitement les occultations.

Cette approche a un spectre d'application beaucoup plus large que celles proposées par Roy ou par Ishikawa et Geiger. En revanche, les algorithmes considérés étant itératifs et sous optimaux, il convient de surveiller la vitesse de convergence et la qualité du minimum obtenu.

En dépit de leurs résultats améliorés, les méthodes basées sur la coupure de graphe possèdent des restrictions. D'une part, à cause de l'utilisation de la contrainte de continuité bidimensionnelle, ces méthodes peuvent provoquer un *excès de régularisation* (lissage), ce qui peut apparaître comme l'effet d'un filtrage local. D'autre part, vu que la fonction de pénalité (coût de l'attribution des disparités différentes aux deux pixels voisins) n'est pas forcément convexe, cela rend la minimisation de la fonction d'énergie un problème NP-complet [Kolmogorov et Zabih, 2001], en conséquence, la solution obtenue n'est qu'une approximation.

La méthode de coupure de graphe transforme le problème de l'appariement stéréoscopique en problème de minimisation d'une fonction d'énergie qui traduit le coût global de la mise en correspondance de tous les pixels de l'image. Dans la section 3.4, nous rappelons la définition du graphe et le problème de coupure minimale avec ses solutions. Ensuite, nous rappelons la formulation générale du problème d'étiquetage, puis nous formulons le problème de la mise en correspondance comme un problème d'étiquetage en mettant l'accent sur la fonction d'énergie globale à minimiser, et puis nous détaillons la construction de graphe de Roy et Cox pour nous permettre de mettre en valeur notre contribution.

3.4 Le Problème de Coupure Minimale

3.4.1 Définition de Graphe

Rappelons qu'un graphe est un ensemble de sites (appelés aussi : noeuds, sommets ou *vertex*) connectés par des arêtes. Une telle connection est définie par une relation binaire entre sites, dite relation de contiguïté. Un graphe est donc une paire (V, E), où V est l'ensemble des sites et E est l'ensemble des arêtes entre les sites $E = \{(u, v)|u, v \in V\}$. Le graphe est non orienté (figure 3.10) si la relation de contiguïté est symétrique, les paires (u, v) ne sont pas ordonnées. Dans un graphe orienté (cf. figure 3.11), les arêtes (on utilise plutôt le terme *arc* dans ce cas) sont orientées. Lorsqu'il existe une arête $e = (u, v)$ qui relie les sites u et v, ces deux sites sont dits *adjacents*, et on les appelle les *extrémités* de l'arête e. Deux arêtes sont appelées adjacentes si elles ont en commun un même site. Un graphe pondéré est un graphe où chaque arête a un poids. Quelques algorithmes exigent que tous les poids soient non négatifs, entiers, etc.

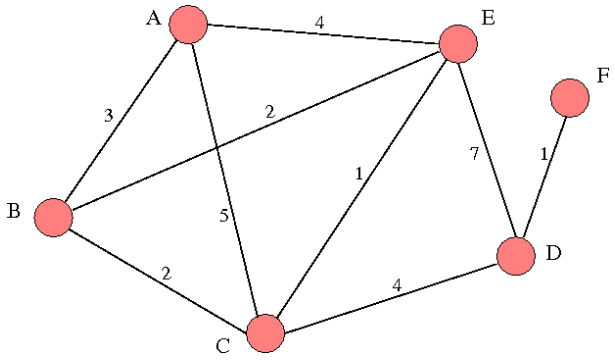


FIG. 3.10 – Graphe Pondéré et non Orienté.

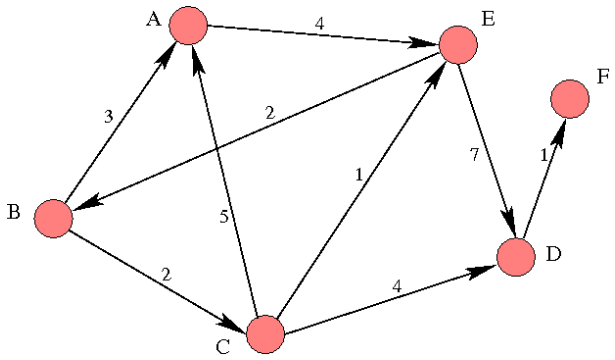


FIG. 3.11 – Graphe Pondéré et Orienté.

3.4.2 Représentation de Graphe

Un graphe peut être représenté par deux structures de données : soit par une matrice de contiguïté soit par une liste de contiguïté.

Représentation par matrice de contiguïté

Un graphe orienté qui possède v sites est représenté par une matrice carrée de dimension v x v. L'élément (i, j) de la matrice vaut 1 si il y a une arête (arc) qui part du site i vers le site j, et vaut 0 dans le cas contraire. Pour un graphe pondéré, la valeur de l'élément (i, j) est égale au poids de l'arête qui part du site i vers le site j, et vaut 0 dans le cas contraire. Un graphe non orienté peut être représenté par une matrice symétrique en donnant la même valeur pour les éléments (i, j) et (j, i) ou par une matrice triangulaire (supérieure ou inférieure).

La figure 3.12 illustre la représentation du graphe orienté de la figure 3.11 par une matrice de contiguïté.

Représentation par liste de contiguïté

Un graphe orienté contenant v sites peut être représenté par un ensemble de v listes de sites. La liste i contient le site j si et seulement si il existe une arête (arc) partant du site i vers le site j. Un graphe pondéré et orienté peut être représenté par une liste des doublets (site, poids). Pour un graphe non orienté, lorsqu'il y a une arête entre les sites i et j, le site j est ajouté à la liste i et le site i est ajouté à la liste j.

La figure 3.13 illustre la représentation du graphe orienté de la figure 3.11 par une liste de contiguïté. La flèche ($\rightarrow$) signifie qu'il y a un lien dans la liste.

De manière générale, la représentation par une liste de contiguïté est plus compacte que les matrices de contiguïté pour les graphes *épars*.

3.4.3 Définition d'une Coupure

Soit $G = (V, E)$ un graphe, où V est l'ensemble de sites et E est l'ensemble des arêtes. Soit S un sous ensemble de V. Une coupure $\delta(S)$ induite par S est le sous ensemble des arêtes (i, j) dont

$$\Big|\{i, j\} \cap S\Big| = 1 \tag{3.9}$$

Ainsi, $\delta(S)$ est composée de toutes les arêtes ayant une seule extrémité dans S. En d'autres termes, une coupure est une partition de l'ensemble des sites V en deux partie S et T. Les arêtes

	A	B	C	D	E	F
A	0	0	0	0	1	0
B	1	0	1	0	0	0
C	1	0	0	1	1	0
D	0	0	0	0	0	1
E	0	1	0	1	0	0
F	0	0	0	0	0	0

FIG. 3.12 – Représentation par une matrice de contiguïté d'un graphe orienté.

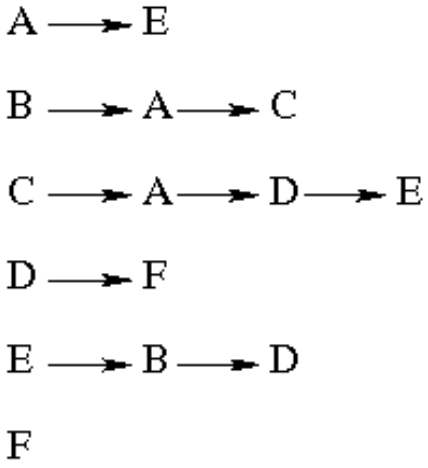


FIG. 3.13 – Représentation par une liste de contiguïté d'un graphe orienté.

(u, v) où le site u est dans S et l'autre extrémité v est dans T sont appelées arêtes de coupure (*cut edge*). Dans le cas d'un graphe pondéré dont chaque arête e a une capacité (coût) c_e, la somme des capacités des arêtes de la coupure s'appelle *le coût de la coupure* :

$$c\Big(\delta(S)\Big) = \sum_{e \in \delta(S)} c_e \tag{3.10}$$

Le problème de la coupure minimale est de trouver la coupure qui a le coût minimal.

3.4.4 Le Problème de Coupure $s - t$

Pour un graphe orienté $G = (V, E)$, les arêtes qui ont le même noeud de départ v sont appelées les arêtes sortantes de v, et la somme de leurs capacités est appelée le degré sortant du site v. les arêtes qui ont le même noeud d'arrivée v sont appelées les arêtes entrantes dans v, et

la somme de leurs capacités est appelée le degré entrant du site v, voir la figure 3.14.

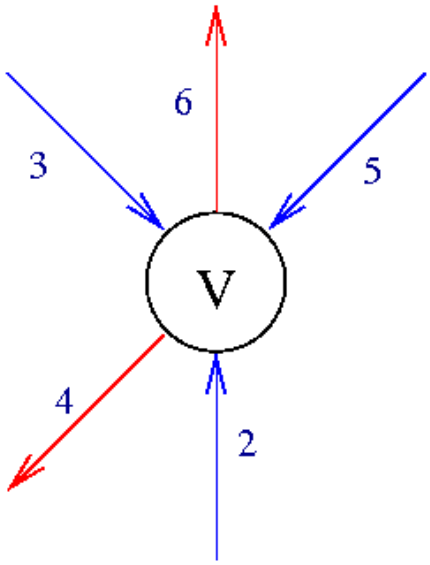


FIG. 3.14 – Les arêtes entrantes sont en bleu et les arêtes sortantes sont en rouge

$$\text{degré entrant dans v} = \sum_{(u,v)\in E} c(u,v) \tag{3.11}$$

$$\text{degré sortant de v} = \sum_{(v,w)\in E} c(v,w) \tag{3.12}$$

On appelle une **source** s un site d'un graphe orienté dont le degré entrant est égal à 0, et un **puits** t un sommet d'un graphe orienté dont le degré sortant est égal à 0.

Le problème de la coupure $s-t$ est le problème de trouver une coupure du graphe qui sépare les sites s et t. Une coupure $\delta(S)$ est une coupure $s-t$ si seulement un site (s par exemple) appartient à S. Une coupure $s-t$ minimale est donc une coupure à coût minimal qui divise le graphe en deux partie, le site s appartient à la première partie tandis que t est dans l'autre partie.

3.4.5 Flot dans un Graphe

Pour bien définir le problème de flot dans un graphe, il est courant de l'introduire par l'exemple de l'écoulement de l'eau dans un réseau de tuyaux. On commence par présenter ce problème afin de donner l'intuition de la formulation théorique qui suit.

Écoulement d'eau dans un réseau

Supposons qu'on ait une source d'eau, un puits et un réseau de tuyaux reliant la source au puits. On admettra que la source peut fournir et que le puits peut recevoir une quantité illimitée d'eau. Le problème du flot maximal est la recherche de l'écoulement maximal qui peut passer à travers le réseau de la source vers le puits. Vue la capacité illimitée de la source et du puits, ce problème est alors uniquement contraint par la capacité du réseau. La capacité de chaque tuyau est proportionnelle à son diamètre. Dans le réseau, il y aura des engorgements de la circulation dans certains tuyaux. Pour aller de la source au puits, l'eau doit nécessairement emprunter un de ces tuyaux. Dans le cas trivial, si tous ces tuyaux sont pleins, le flot sera égal à la somme de leurs capacités. L'engorgement correspond alors à un ensemble de tuyaux qui sépare la source du puits et dont la somme des capacités est égale au flot maximal qui peut traverser le réseau. Si le flot est maximal à travers le réseau, on est dans le cas où tous les tuyaux d'engorgement sont pleins.

Définition mathématique

Un flot dans un graphe $G = (V, E)$ pondéré où chaque arête $e = (i, j)$ a une capacité $c_e \in \mathbb{R}^+$, est une fonction ϕ qui relie l'ensemble des arêtes E à l'ensemble des nombres réels $\mathbb{R}$.

Un flot $\phi : V \mathrm{x} V \rightarrow \mathbb{R}$ vérifie les propriétés suivantes :

- $\phi(i, j) = -\phi(j, i)$.
- $\sum_{i \in V - \{s,t\}} \phi(i, j) = 0$: conservation du flot, cela veut dire que tout ce qui rentre dans un site est égale à ce qui sort de ce site (sauf pour la source et le puits).
- $\phi(i, j) \leq c(j, i) \quad \forall (i, j) \in E$: le flot dans une arête est inférieur ou égal à sa capacité.
- $c(j, i) = 0 \quad \forall (i, j) \notin E$.

Une arête est dite saturée si le flot passant par elle est égale à sa capacité non nulle :

$$e = (i, j) \textbf{ est saturée} \Leftrightarrow \Big(c(i, j) = \phi(i, j) \text{ et } c(i, j) > 0 \Big) \tag{3.13}$$

3.4.6 Le graphe résiduel

Soit ϕ un flot dans le graphe $G = (V, E)$. Le graphe résiduel $G_r = (V, E_r)$ est construit selon le procédé suivant :

$$\forall (i,j) \in E : \begin{cases} \text{si } \phi(i,j) < c(i,j) & \Rightarrow (i,j) \in E_r \text{ , } c_r(i,j) = c(i,j) - \phi(i,j) \\ \text{si } \phi(i,j) > 0 & \Rightarrow (j,i) \in E_r \text{ , } c_r(j,i) = \phi(i,j) \end{cases} \tag{3.14}$$

Ainsi, pour chaque arête du graphe G à capacité non nulle $c(i,j)$ et de flot $\phi(i,j)$, il y aura deux arêtes dans le graphe résiduel G_r à capacités $c_r(i,j) = c(i,j) - \phi(i,j)$ et $c_r(j,i) = \phi(i,j)$. La figure 3.15 représente un flot maximal dans un graphe, et la construction du graphe résiduel est illustrée dans la figure 3.16.

Chemin Augmenté

Un chemin augmenté dans le graphe résiduel G_r est un chemin de la source s vers le puits t composé des arêtes à capacité résiduelle non nulle.

3.4.7 Solution du problème de Coupure $s-t$ Minimale

Le problème de coupure $s-t$ minimale a été traditionnellement résolu en utilisant les algorithmes de flot dans les réseaux. De plus, il peut être réduit à un problème de flot maximal. Soit s une source et t un puits du graphe G, le théorème de flot maximal - coupure minimale s'écrit :

$$MaxFlot(s,t) = \min_{S : s \in S, t \notin S} c\Big(\delta(S)\Big) \tag{3.15}$$

Ce théorème montre que la valeur maximale du flot de la source s vers le puits t est égale au coût de la coupure $s-t$ minimale. Ford et Fulkerson [Ford et Fulkerson, 1962] ont montré que le flot de la source s vers le puits t fait saturer un ensemble d'arêtes divisant les sites en deux parties S et T, tel que $s \in S$ et $t \in T$. Ce théorème relie la valeur du flot maximal de la source s vers le puits t avec le coût de la coupure $s-t$ minimale. Le théorème ne spécifie aucune relation entre la valeur de la coupure $s-t$ minimale et l'ensemble des arêtes de la coupure. Pourtant, si on a un flot maximal, on peut obtenir une coupure $s-t$ minimale. L'algorithme 3.1 illustre la procédure pour obtenir une coupure $s-t$ minimale.

La figure 3.15 illustre une coupure minimale dans un graphe orienté et pondéré. Les points des arcs sont en noirs, tandis que les valeurs des flots sont en bleu. La ligne rouge pointillée coupe les arêtes de la coupure minimale. Dans cet exemple, on constate que la valeur du flot maximal est 11, et le coût de la coupure minimale est aussi 11.

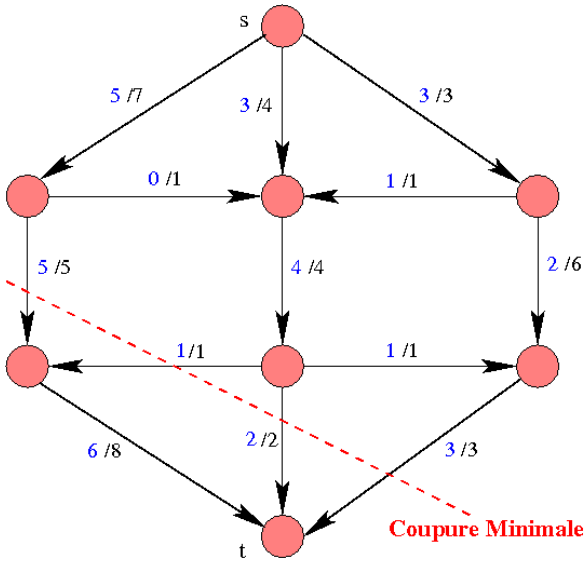


FIG. 3.15 – Flot maximal dans un graphe, notation : flot $\phi(i,j)$/ capacité $c(i,j)$, la ligne rouge pointillée représente la coupure minimale.

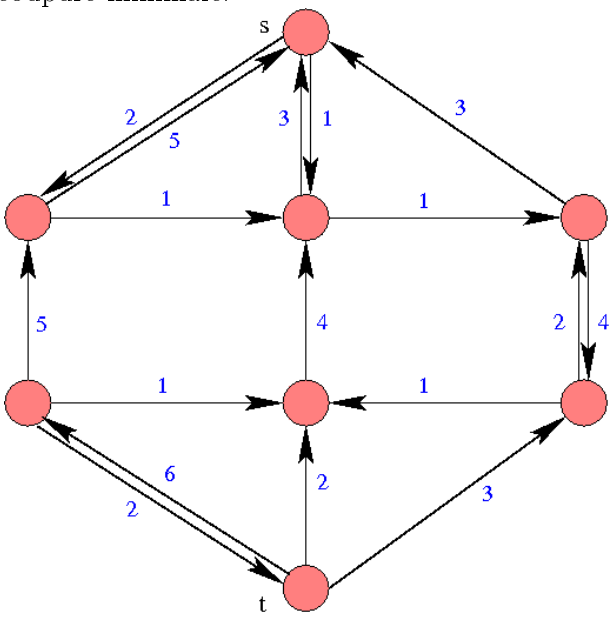


FIG. 3.16 – Graphe Résiduel construit à partir du graphe de la figure 3.15

Algorithme 3.1 : L'algorithme de Coupure $s-t$ Minimale

début

1. Résoudre le problème de Flot $s-t$ maximal.

2. Trouver l'ensemble des sites S atteignables de la source s dans le graphe résiduel.

3. Trouver l'ensemble des sites T non atteignables de la source s dans le graphe résiduel.

4. Considérer les arêtes qui commencent d'un site dans S et se terminent dans un site dans T. Ces arêtes forme la coupure $s-t$ minimale recherchée.

fin

3.4.8 L'algorithme du flot maximal

On trouve dans la littérature deux approches pour résoudre le problème de flot maximal [Cormen *et al.*, 2001]. La première est l'algorithme de chemin augmentant dû à Ford et Fulkerson [Ford et Fulkerson, 1962], et la deuxième est de type *Push-Relabel*.

L'algorithme du chemin augmentant

Ford et Fulkerson [Ford et Fulkerson, 1962] ont élaboré une solution du problème de flot maximal basée sur le chemin augmenté. En effet, si il existe un chemin augmenté dans le graphe résiduel G_r, alors le flot n'est pas maximal. L'algorithme commence par trouver un chemin augmenté, puis on accroît le flot par une valeur égale à la capacité du chemin augmenté (la capacité minimale de toutes les arêtes dans le chemin augmenté), et ensuite on remet à jour le graphe résiduel G_r. Cette opération sera répétée tant qu'il y aura un chemin augmenté.

La complexité de l'algorithme du chemin augmenté est de $O(e*MaxFlot)$ [Ford et Fulkerson, 1962], où e est le nombre des arêtes dans le graphe et $MaxFlot$ la valeur du flot maximal.

L'algorithme de *Push-Relabel*

L'idée de l'algorithme de *Push-Relabel* peut être clarifiée par l'exemple suivant : supposons que le graphe est représenté comme suit : la source s est au sommet et le puits t en bas du graphe (hauteur nulle). On fait couler l'eau de la source s vers le bas. À chaque site, l'eau en *excès* va couler vers le site le plus bas avec un débit majoré par la capacité de l'arc reliant ces deux sites. Après cette opération, l'eau est maintenant dans le site du bas, et l'excès est diminué dans le site du haut. On suit l'eau jusqu'à ce qu'elle arrive au puits t. L'algorithme s'arrête lorsqu'il n'y aura plus d'excès dans aucun site.

Algorithme 3.2 : Flot maximal : L'algorithme du chemin augmentant

$G = (V, E)$: un graphe

V : l'ensemble des sites

E : l'ensemble des arêtes

s : la source

t : le puits

pour $(u, v) \in E$ **faire**

$f(u, v) \leftarrow 0$

$f(v, u) \leftarrow 0$

fin

Trouver le graphe résiduel G_r

tant que *il existe un chemin augmenté p dans le graphe résiduel G_r entre s et t* **faire**

$c_f(p) = \min\{c_f(u, v) : (u, v) \in p\}$

pour $(u, v) \in p$ **faire**

$f(u, v) \leftarrow c_f(p) + f(u, v)$

$f(v, u) \leftarrow -f(u, v)$

fin

Mettre à jour le graphe résiduel G_r

fin

La **Hauteur** d'un site : On observe que le chemin le plus long de la source au puits contient au maximum V noeuds. Ainsi, on peut affecter la hauteur de chaque site de manière suivante :

- $Hauteur(s) = V$.
- $Hauteur(t) = 0$.
- $\phi(u, v) > 0 \Longleftrightarrow Hauteur(u) > Hauteur(v)$

En quelques mots, dans l'algorithme de *Push-Relabel*, les hauteurs des sites (sauf s et t) sont ajustées, et le flot est envoyé entre noeuds, jusqu'à son arrivée au puits t. De plus, les hauteurs des sites internes sont augmentées jusqu'à ce que tout flot dans le graphe n'ayant pas encore arrivé au puits t, puisse y arriver.

Afin d'introduire les notions d'excès, on définit d'abord le *préflot*.

Algorithme 3.3 : Flot maximal : L'algorithme général de Push-Relabel

$G = (V, E)$: un graphe

V : l'ensemble des sites

E : l'ensemble des arêtes

s : la source

t : le puits

tant que *il existe un site v dans le graphe G qui a un excès ($\epsilon(v) > 0$)* **faire**

Sélectionner le site v.

Faire une opération légale de *Push(v)*

ou Faire une opération légale de *Relabel(v)*

fin

Définition de Préflot

Un préflot est similaire au flot, à l'exception que la quantité totale qui s'écoule dans un site peut excéder le flot sortant (le principe de conservation du flot n'est plus respecté). La notion de préflot à été introduite par Karzanov [Karzanov, 1974]. Soit $G = (V, E)$ un graphe. Un préflot est une fonction :

$$P : V \mathrm{x} V \rightarrow \mathbb{R}^+ \qquad \text{telle que :}$$

$$\begin{cases} * & \forall (u,v) \in E : P(u,v) \leq c(u,v) \\ * & \forall (u,v) \in E : P(v,u) = -P(u,v) \\ * & \forall u \in V - \{s\} : \sum_{(w,u)\in E} P(w,u) - \sum_{(w,u)\in E} P(u,w) \geq 0 \end{cases} \tag{3.16}$$

Soit $\epsilon(u) = P(V, u)$, $\epsilon(u)$ est appelé l'excès en u, qui est la différence entre le débit entrant dans le site u et le débit sortant de ce site. Un site $u \in V - \{s, t\}$ est dit débordé si $\epsilon(u) > 0$.

L'opération Push(v)

Faire un *push* de u à v signifie : envoyer une partie de l'excès de flot du site u au site v. Avant de faire une opération de push, on doit vérifier trois conditions :

- $\epsilon(u) > 0$: il y a un excès dans le site u ce qui veut dire qu'il y a plus de flot vers le site

qu'à partir de lui.

- $c(u,v) - f(u,v) > 0$: l'arête (u,v) n'est pas saturée.
- $h(u) > h(v)$. Le site u est situé à une "hauteur" plus élevée que le site v.

Quand ces conditions sont satisfaites, on va envoyer une quantité de flot égale à :

$$\text{valeur de } Push = \min\Big\{\big(\epsilon(u)\ ,\ c(u,v) - \phi(u,v)\big)\Big\} \tag{3.17}$$

L'opération Relabel(v)

Durant l'opération de Relabel d'un site u, on augmente sa *hauteur* jusqu'à ce qu'il soit plus haut qu'au moins un site v dont il existe une arête non saturée de u vers v. Les conditions de re-label sont :

- $e(u) > 0$: il y un excès de flot.
- $Hauteur(u) \leq Hauteur(v)$ pour tous les sites v tels que $c(u,v) - f(u,v) > 0$. Les seuls sites ayant des arêtes entrantes non saturées sont plus hauts.

Quand on re-label un site u, on ajuste sa hauteur $h(u)$ pour qu'elle soit égale à la plus petite valeur vérifiant $h(u) > h(v)$ pour tous les sites v ayant $c(u,v) - f(u,v) > 0$.

Complexité des algorithmes de flot maximal

- L'algorithme 3.3 donne une implémentation générale de l'algorithme *Push-Relabel*, qui utilise les operations de *Push* et *Relabel* définies juste après l'algorithme [Goldberg et Tarjan, 1988].
- L'algorithme de *Push-Relabel* est un des algorithmes les plus performants pour calculer le flot maximal. L'algorithme général a une complexité de $O(v^2e)$ [Cormen *et al.*, 2001], avec v le nombre de sites dans V et e est le nombre des arêtes dans E, alors qu'une implémentation avec *FIFO vertex* (file d'attente des sites) a une complexité de $O(v^3)$ [Cormen *et al.*, 2001]. La complexité théorique de l'algorithme proposé par Goldberg est de $\mathcal{O}(v\ e\ \log(v^2/e))$ [Goldberg et Tarjan, 1988].
  Pour notre implémentation, nous avons choisi l'implémentation de *Push-Relabel* proposée par Goldberg [Goldberg, 1985], inclue dans *Boost Graph Library* [1]. La complexité est de $O(v^3)$.

[1] http ://www.boost.org/libs/graph/doc/index.html

Notons que Boykov et Kolmogorov [Boykov et Kolmogorov, 2004] ont proposé un nouvel algorithme, et ils ont montré que sur des graphes typiques pour la vision, leur algorithme est 2 à 5 fois plus rapide que les autres algorithmes, y compris l'approche de *push-relabel*. En effet, leur algorithme est de type de chemin augmenté. L'algorithme commence par construire des arbres de détections des chemins augmentés. La bonne performance de l'algorithme vient du fait qu'il réutilise ces arbres dans les étapes suivantes pour ne pas recommencer à zéro (*from scratch*) leur reconstruction.

3.5 Mise en Correspondance Stéréoscopique par Coupure de Graphe

3.5.1 Formulation générale du problème d'étiquetage

Plusieurs problèmes en vision peuvent être formulés comme un problème d'étiquetage (*Labelling Problem*). Dans un tel problème, on distingue un ensemble de sites et un ensemble d'étiquettes. Les sites représentent les *features* de l'image, pour lesquels on veut estimer une quantité : des pixels, segments, etc. Les étiquettes représentent les quantités associables à ces sites : l'intensité, la disparité, un numéro de région, etc.

Soit $\mathcal{P} = 1, 2, \ldots, n$ un ensemble de n sites, et soit $\mathcal{L} = \{l_1, ..., l_k\}$ un ensemble de k étiquettes. L'étiquetage est définie par une application de $\mathcal{P}$ dans $\mathcal{L}$

$$\begin{aligned} f: \quad & \mathcal{P} \rightarrow \mathcal{L} \\ & s_p \mapsto f_p \quad = f(s_p) = l_i \end{aligned}$$

Donc $f(\mathcal{P}) = \{f_1, \ldots, f_n\}$

On affecte une fonction d'énergie à cette fonction d'étiquetage : une **forme générale des fonctions d'énergie** est donnée par :

$$E(f) = E_{data}(f) + \lambda . E_{prior}(f) \tag{3.18}$$

Le premier terme $E_{data}(f)$ représente l'énergie intrinsèque aux données (*data energy*), qui traduit les contraintes de l'association des étiquettes aux données en fonction de leurs propriétés intrinsèques sans prendre en compte aucune information a priori. Le deuxième terme $E_{prior}(f)$ regroupe les énergies extrinsèques (*prior energy*) qui traduisent les contraintes définies par des connaissances a priori, comme par exemple la contrainte de 'continuité' qui dit que la probabilité que deux sites voisins ou adjacents aient le même label, est élevée. La constante λ peut contrôler l'importance relative des deux termes ; plus λ est grand plus on donne de l'importance aux informations a priori.

La fonction d'énergie $E_{data}(f)$ doit être choisie pour affecter un coût important aux associations donnée/étiquette qui sont les moins pertinentes.

$$E_{data}(f) = \sum_{p \in \mathcal{P}} D_p(f_p) \tag{3.19}$$

où $D_p(f_p) \geq 0$ mesure le coût de l'affectation de l'étiquette f_p avec le site p.

La fonction d'énergie a priori $E_{prior}(f)$ doit attribuer un coût important aux associations f_p non compatibles avec l'information a priori. Le choix de cette fonction dépend du type de problème, mais en général, une des fonctions d'énergie a priori exprime des contraintes de lissage (*smoothing*). Cette contrainte est très connue en vision par ordinateur, et elle est bien adaptée quand la qualité à estimer varie lentement partout ou presque partout : en 3D, cela correspond à l'hypothèse que le monde est continu par morceaux. Une telle hypothèse est prise en compte en introduisant une énergie d'information a priori de type lissage E_{smooth}.

Pour formuler l'énergie de lissage, on a besoin de modéliser comment les pixels interagissent entre eux : souvent il est suffisant d'exprimer comment un pixel interagit avec ses voisins. Notons N_p l'ensemble des pixels voisins du pixel p, et $\mathcal{N}$ l'ensemble de paires voisines $\{p, q\}$: $\mathcal{N}$ est dit un système de voisinage. L'énergie de lissage peut s'écrire :

$$E_{smooth}(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p, f_q) \tag{3.20}$$

où $V_{\{p,q\}}(f_p, f_q)$ est une fonction d'interaction de voisinage : cette fonction doit attribuer des pénalités aux paires $\{p, q\}$ si les pixels p et q ont des étiquettes différentes. La forme de $V_{\{p,q\}}(f_p, f_q)$ détermine le type de lissage a priori. Avec ces notations, l'énergie de lissage globale est la somme des fonctions d'interaction de voisinage de tous les pixels voisins. Souvent on choisit d'écrire $V_{\{p,q\}}(f_p, f_q)$ sous la forme :

$$V_{\{p,q\}}(f_p, f_q) = u_{\{p,q\}} V(f_p, f_q) \qquad , \quad u_{\{p,q\}} \in R^+ \tag{3.21}$$

où V est un potentiel homogène, et $u_{\{p,q\}}$ est un terme multiplicateur dépendant de deux sites considérés p et q. La figure 3.17 représente un potentiel linéaire, $V(f_p, f_q) = |f_p - f_q|$. En stéréovision le terme $u_{\{p,q\}}$ est souvent une fonction décroissante de la norme du gradient entre les sites p et q, ce qui permet de favoriser la coïncidence des discontinuités de la disparité avec les contours de l'image de référence. Ce choix est traduit par la fonction $u_{\{p,q\}}$: $u_{\{p,q\}} = U(|I_p - I_q|)$. Le terme $u_{\{p,q\}}$ représente la pénalité d'attribuer des disparités différentes aux pixels voisins p et q ; la valeur de cette pénalité doit être petite pour une paire $\{p, q\}$ qui est sur un contour, donc avec une large différence d'intensité $|I_p - I_q|$. En pratique, on utilise une fonction empirique décroissante $U()$. La figure 3.18 montre la fonction $u_{\{p,q\}}$.

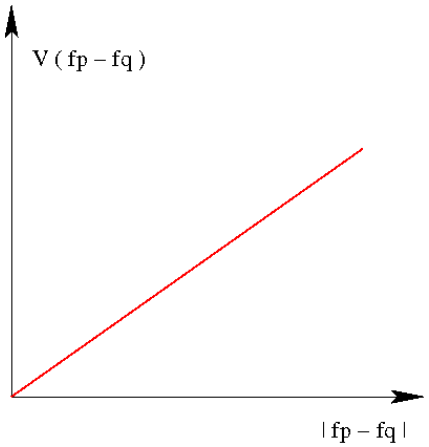


FIG. 3.17 – Fonction de potentiel linéaire.

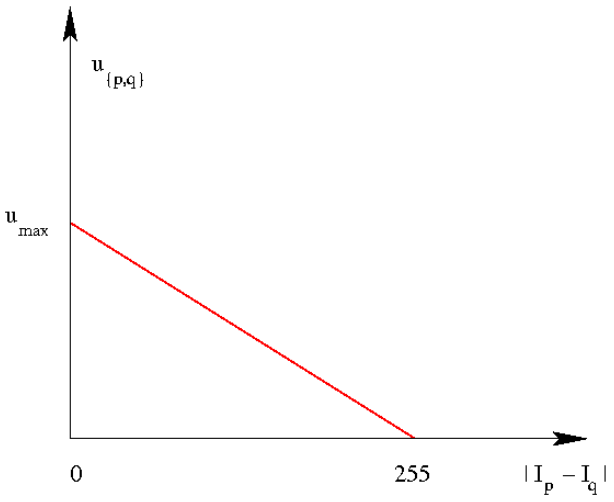


FIG. 3.18 – Fonction de Potentiel $u_{\{p,q\}}$

3.5.2 Graphe et Énergie

Le principe de la méthode de coupure de graphe est de créer un graphe de manière à ce qu'une coupure représente une fonction et que la valeur de cette coupure soit l'énergie associée à la fonction. La valeur de la coupure minimale sera donc égale à l'énergie minimale de cette fonction.

En reprenant la formulation générale des problèmes d'étiquetage, considérons le potentiel linéaire suivant :

$$V_{\{p,q\}}(f_p, f_q) = u_{\{p,q\}} \Big| f_p - f_q \Big| \tag{3.22}$$

qui conduit à l'énergie globale :

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \lambda \sum_{\{p,q\} \in \mathcal{N}} u_{\{p,q\}} \Big| f_p - f_q \Big| \tag{3.23}$$

Nous montrons ci-dessous comment construire un graphe auquel on associe une fonction d'énergie, dont la coupure minimale donne la valeur minimale de la fonction d'énergie. Cette approche est due à Roy [Roy et Cox, 1998], mais on utilise la reformulation proposée par Olga Veksler [Veksler, 1999] qui met en évidence l'énergie à minimiser.

3.5.3 Construction du Graphe Complet

L'objectif est de construire un graphe $G = (V, E)$ où V contient deux noeuds particuliers : une source s et un puits t et dont la coupure minimale représente une minimisation de la mise en correspondance stéréoscopique. Le graphe est construit de la manière suivante :

- Soit k le nombre d'appariements possibles (en stéréovision, k est donné par la plage admissible de disparités, liée à la profondeur minimale et maximale à laquelle se trouvent des objets dans la scène perçue). Par exemple, pour une plage de disparités $[0, 40]$, la valeur de k est 41.

- À chaque pixel p de l'image, on associe une chaîne composée de $k-1$ noeuds (sites) nommés $p_1, p_2, ..., p_{k-1}$. Ces noeuds sont connectés par des arêtes appelées *t-link* ou **arêtes de disparité**, et notées $t_1^p, t_2^p, ..., t_k^p$ avec $t_1^p = [s, p_1]$, $t_j^p = [p_{j-1}, p_j]$ et $t_k^p = [p_{k-1}, t]$. Les *t-links* sont alors l'ensemble des arêtes pour un pixel donné qui relie la source au premier noeud, puis relie ce noeud au noeud suivant, ainsi de suite jusqu'à relier le dernier noeud de la chaîne avec le puits. Pour une plage de disparités de largeur k, et pour le pixel p, on aura $k-1$ noeuds dans la chaîne et $k-2$ *t-links* entre ces noeuds, à qui s'ajoutent deux *t-links*, un avec la source et l'autre avec le puits, ce qui fait au total k *t-links*. Alors pour

chaque pixel, on a k *t-links* dont la capacité de chacun est égale au coût de la mise en correspondance avec la disparité correspondant à l'indice du noeud dans la chaîne.

- À chaque *t-link* on affecte une capacité $K_p + D_p(l_j)$ où D_p est le coût d'appariement du pixel considéré pour la valeur de disparité correspondante, et K_p est une constante qui satisfait la contrainte 3.24.

$$K_p > (k-1) \sum_{q \in N_p} u_{\{p,q\}} \tag{3.24}$$

En fait, la constante K_p est choisie pour qu'elle soit plus grande que la somme des capacités des arêtes de pénalité (*t-links*) qui ont une extrémité dans la chaîne de noeuds associée au pixel p.

- À chaque paire de pixels voisins p et q les chaînes correspondantes sont reliées par des arêtes appelées *n-link* ou **arêtes de pénalité**. L'arête *n-link* qui se trouve au niveau $j \in \{1, 2, \ldots, k-1\}$, notée $\{p_j, q_j\}$ est de capacité $u_{\{p,q\}}$.

La figure 3.19 illustre les noeuds pour un pixel donné, avec les *t-links* (qui sont colorés en bleu) et les *n-links* (en vert).

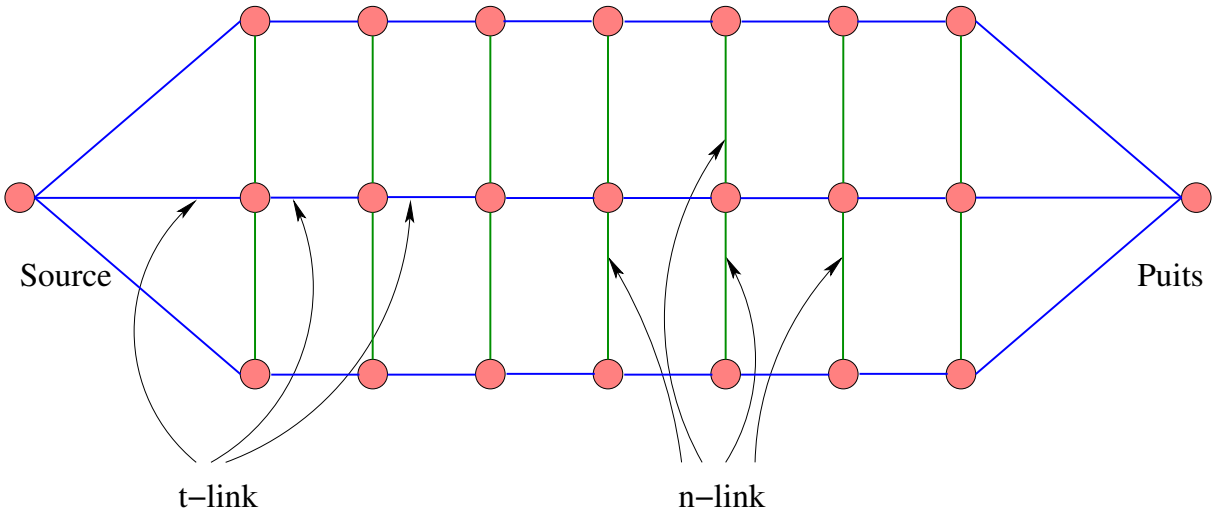


FIG. 3.19 – Les *t-links* et les *n-links*.

La capacité d'une coupure $s-t$ de ce graphe est la somme des capacités de toutes les arêtes coupées. Vue la méthode de construction du graphe, la capacité de la coupure est constituée de deux parties : la première est la somme des capacités des arêtes *t-link*, et la deuxième est la somme des capacités des arêtes *n-link*, voir l'équation 3.25. Dans l'équation 3.25, le premier terme représente la somme des capacités des *t-links* (arêtes de disparité) coupées, et le second terme est égal à la somme des capacités des *n-links* (arêtes de pénalité) coupées. En effet l'addition de la constante K_p sert à assurer l'unicité de la coupure de chaque *t-link*, voir [Roy et Cox, 1998] pour une preuve de cette unicité.

$$E(\delta) = \sum_{p \in Image} \Big(K_p + D_p(f_p) \Big) \; + \; \lambda \sum_{\{p,q\} \in \mathcal{N}} u_{\{p,q\}} \Big| f_p - f_q \Big| \tag{3.25}$$

Une autre méthode pour assurer l'unicité de la coupure de chaque *t-link* est proposée par Ishikawa et Geiger [Ishikawa, 2000]. Leur graphe est similaire au graphe de Roy, mais il est orienté et il contient en plus de la chaîne des sites associés au pixel p, une chaîne inverse à capacité infinie.

Une coupure de graphe consiste à diviser le graphe en deux parties. Les *t-link* coupés forment la surface de profondeur recherchée, ce qui permet d'associer une disparité à chaque pixel. Le problème de coupure de graphe peut se résoudre par le flot maximal. Ford et Fulkerson [Ford et Fulkerson, 1962] ont montré que le flot de la source s vers le puits t fait saturer un ensemble d'arêtes divisant les sites en deux parties S et T. Pour déterminer la disparité d'un pixel p, la coupure minimale va forcément couper un *t-link* dans la chaîne correspondante de ce pixel. L'indice (*zero-based index*) de cette arête coupée dans la chaîne nous permet de retrouver la disparité de ce pixel. Ainsi, pour une plage de disparités $[d_{min}, d_{max}]$, un *t-link* dont l'indice est j correspond à une disparité $d_{min} + j$.

Cette méthode permet d'attribuer une disparité à chaque pixel de l'image. La surface de disparité peut être visualisée comme une surface qui traverse toutes les arêtes *t-links* coupées du graphe. La figure 3.20 illustre une coupure minimale et la surface de disparité correspondante. Notons aussi que cette méthode permet d'obtenir une disparité entière, et qu'elle ne permet pas d'obtenir des disparités réelles (sous-pixelliques). En fait, la construction du graphe complet exige que chaque arêtes dans les *t-links* du pixel p correspond à une valeur de la disparité dans la page $[d_{min}, d_{max}]$, et en conséquence les valeurs non entière (réelle) de la disparité ne sont pas acceptables par cette méthode. Nous verrons dans la section suivante (sur le graphe réduit) comment on peut surmonter ce problème.

Exigences de cette méthode

Nous allons calculer le nombre total des noeuds et des arêtes dans ce graphe pour mettre en évidence deux contraintes : la contrainte de mémoire demandée, et la contrainte de temps de calcul.

Soient W et H les largeur et hauteur de l'image, $[0, d_{max}]$ la plage de disparités. Soit v le nombre de sites dans le graphe. Le nombre des appariements possibles est donc $k = d_{max} + 1$. Vu que pour chaque pixel de l'image on construit une chaîne de $k - 1 = d_{max}$ noeuds, donc le nombre total des sites dans le graphe est de (le terme $+2$ est pour l'ajout de la source et du puits) :

$$v = W \; H \; d_{max} + 2 \tag{3.26}$$

Dans chaque chaîne (sauf aux frontières), chaque noeud est relié avec ses 6 voisins par des arêtes. Le nombre total des arêtes dans le graphe est donc :

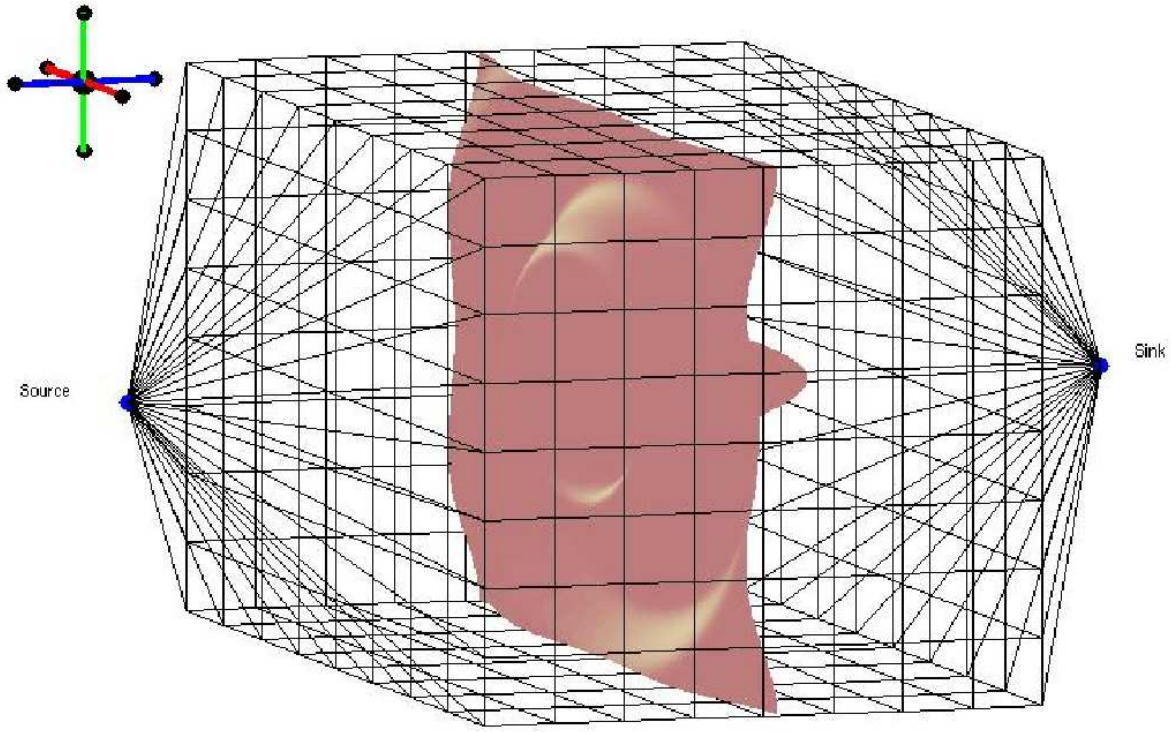


FIG. 3.20 – La surface de disparité correspondant à la coupure de graphe comme proposée par Roy et Cox [Roy et Cox, 1998].

$$e = 6\,W\,H\,d_{max} - 2\,d_{max}(W + H) \tag{3.27}$$

Le terme $2\,d_{max}(W+H)$ vient du fait que sur les frontières du graphe, il manque des arêtes.

Le tableau 3.1 donne une idée simplifiée des nombres de sites et d'arêtes dans un graphe construit pour une image de taille 512 x 512 pixels et pour différentes plages de disparités. Dans ce tableau, et pour une plage de disparités égale à $[0, 40]$, on aura plus de 10 x 10^6 sites et plus de 60 x 10^6 arêtes. Cet exemple illustre clairement le besoin d'avoir des ressources énormes en matière de mémoire (RAM). De plus, vu que la complexité de l'algorithme du flot maximal qu'on a choisi est $O(v^3)$, le temps de calcul sera très important, en particulier pour les images de grande taille et pour les plages de disparités larges.

3.5.4 Construction d'un Graphe Réduit

Les problèmes majeurs de la méthode de graphe complet présentée dans la section précédente 3.5.3 est qu'elle est gourmande en mémoire et en temps de calcul. Pour surmonter ces problèmes, nous proposons de construire un graphe réduit. Ce graphe réduit contient un nombre réduit de sites et d'arêtes, afin de diminuer la taille de la mémoire et réduire le temps pour rechercher la coupure minimale [Zureiki *et al.*, 2007a], [Zureiki *et al.*, 2007b].

	d_{max}=30	d_{max}=40	d_{max}=50
Nombre de sites (v)	7864320	10485760	13107200
Nombre d'arêtes (e)	47124480	62832640	78540800

TAB. 3.1 – Nombre de sites et d'arêtes dans le graphe construit pour une image de taille 512x512 et pour différentes plages de disparités

Dans le graphe réduit, et pour chaque pixel de l'image, on garde seulement quelques disparités potentielles, issues d'une méthode locale de mise en correspondance, alors que la méthode générale proposée notamment par Roy, considère toutes les valeurs possibles de disparité.

L'étape de construction du graphe réduit ressemble à celui du graphe complet, mais avec quelques différences :

- Par une méthode d'appariement local (stéréo corrélation avec par exemple un score CENSUS), on calcule pour chaque pixel, les coûts de mise en correspondance du pixel p avec toutes les valeurs possibles de disparité dans la plage de disparités $[d_{min}, d_{max}]$.

- Puis, on choisit les N meilleures valeurs (pour notre exemple, on va considérer $N = 4$ sans manque de généralité). Ce choix peut être fait selon différents critères, par exemple, avec un score ZNCC très classique en stéréo, on peut garder les valeurs de disparité autour du sommet, ou les quatre (s'ils existent) maximums locaux, etc. On va noter les N disparités choisies pour le pixel p, comme $d_{1,p}, d_{2,p}, \ldots, d_{N,p}$, et les coûts correspondants par $D_{\{p,d_{1,p}\}}, D_{\{p,d_{2,p}\}}, \ldots, D_{\{p,d_{N,p}\}}$.

- Pour réduire la taille du graphe, nous faisons la simplification suivante : pour chaque pixel de l'image, au lieu de garder une chaîne de $k-1$ noeuds et de k arêtes *t-links* (voir la section précédente), on supprime tous les noeuds et les arêtes *t-links* dans la chaîne sauf $N-1$ noeuds (et donc N arêtes).

- Ainsi, à chaque pixel p on associe une chaîne de noeuds $\{p_{L_1}, p_{L_2}, \ldots, p_{L_{N-1}}\}$, où p_{L_i} est un noeud au niveau i. Ces noeuds sont connectés par des arêtes (*t-link*) notés $\{t_1^p, t_2^p, \ldots, t_N^p\}$ avec $t_1^p = [s, p_{L_1}]$, $t_2^p = [p_{L_1}, p_{L_2}]$, $t_3^p = [p_{L_2}, p_{L_3}], \ldots, t_N^p = [p_{L_{N-1}}, t]$.

- À chaque *t-link* on affecte une capacité $C + D_{\{p,d_{i,p}\}}$, où C est une constante qui satisfait la contrainte 3.29. L'ajout de cette constante est fait pour assurer l'unicité de la coupure de la chaîne des *t-links* en une seule arête.

- À chaque paire de pixels voisins p et q, les chaînes correspondantes sont reliées par des arêtes (*n-link*) aux $(N-1)$ niveaux, et de capacité selon l'équation 3.28.

La figure 3.21 détaille la construction du graphe réduit. Les arêtes pointillées sont les arêtes du graphe complet de la section 3.5.3. Les noeuds non supprimés sont en rouge, les nouveaux *t-link* sont en bleu et les nouveaux *n-links* sont en vert. La figure 3.22 illustre une projection frontale du graphe réduit.

$$\textbf{Capacité de n-link au niveau i} = u_{\{p,q\}} * \Big(\big|d_{i,p} - d_{i,q}\big| + 1\Big) \tag{3.28}$$

La constante C vérifie la contrainte suivante :

$$C > N * \max_{\{p,q\}\in\mathcal{N}}(u_{\{p,q\}}) * |d_{max} - d_{min}| \tag{3.29}$$

L'Énergie à Minimiser par le Graphe Réduit

Nous avons vu que l'énergie globale à minimiser par le graphe complet est composée de deux termes (équation 3.23). Le premier terme représente l'énergie intrinsèque des données, qui traduit le coût d'attribuer des disparités aux pixels de l'image. Le deuxième terme regroupe les contraintes de continuité (les pixels voisins ont des disparités voisines). La constante λ peut contrôler l'importance relative des deux termes. Ainsi, l'énergie a priori apparaît seulement dans les poids associés aux arêtes de pénalité (*n-links*) du graphe complet. En revanche, dans le graphe réduit, on distingue deux types d'informations a priori. Le premier type représente les informations acquises par les méthodes locales de la mise en correspondance ; il se manifeste dans le choix des noeuds du graphe réduit. Le deuxième type est le lissage qui intervient dans les pénalités associées aux arêtes *n-links*. En conséquence, nous exploitons la connaissance a priori que la disparité du pixel p a seulement N valeurs possibles (les plus probables) d'une autre manière. En effet, nous considérons la suppression des noeuds et arêtes non potentiels comme une forme nouvelle pour représenter cette connaissance préalable.

L'énergie totale minimisée par la coupure de graphe réduit peut être écrite :

$$E(f) = \sum_{\substack{p\in\mathcal{P}\\ f_p\in\{d_{p,1},\ldots,d_{p,N}\}}} D_p(f_p) + \lambda \sum_{\substack{\{p,q\}\in\mathcal{N}\\ f_p\in\{d_{p,1},\ldots,d_{p,N}\}\\ f_q\in\{d_{q,1},\ldots,d_{q,N}\}}} u_{\{p,q\}}|f_p - f_q| \tag{3.30}$$

Exigences de cette méthode

Nous allons calculer le nombre total des noeuds et des arêtes du graphe réduit. Avec les mêmes notations utilisées en section 3.5.3 pour W, H, d_{max}, k, etc. Soit $N = 4$ le nombre des disparités à choisir pour un pixel parmi les valeurs possibles dans la plage de disparités. On obtient les nombres des sites et arêtes en remplaçant $(N-1)$ par d_{max} dans les équations 3.26 et 3.27 :

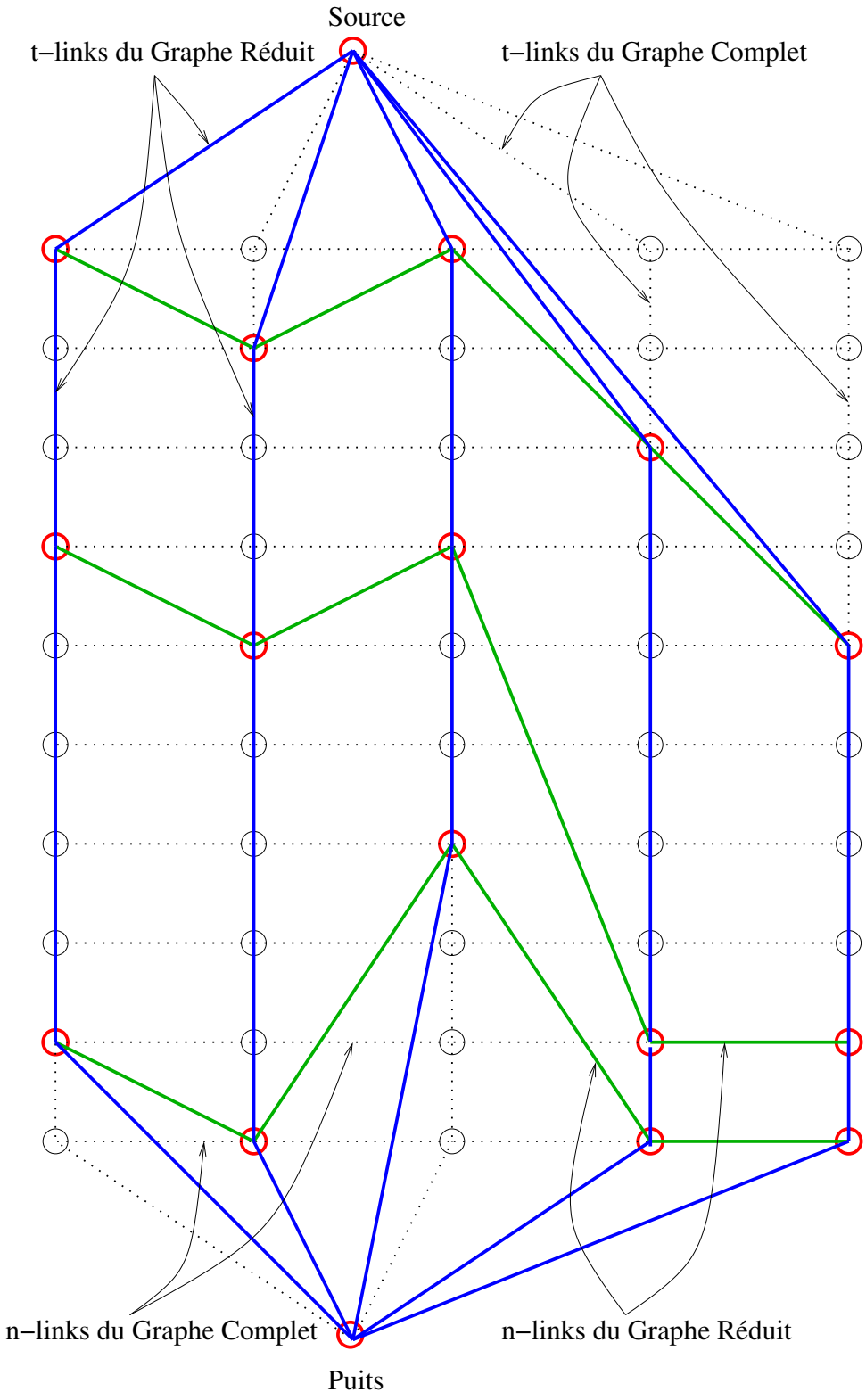


FIG. 3.21 – Construction du Graphe Réduit

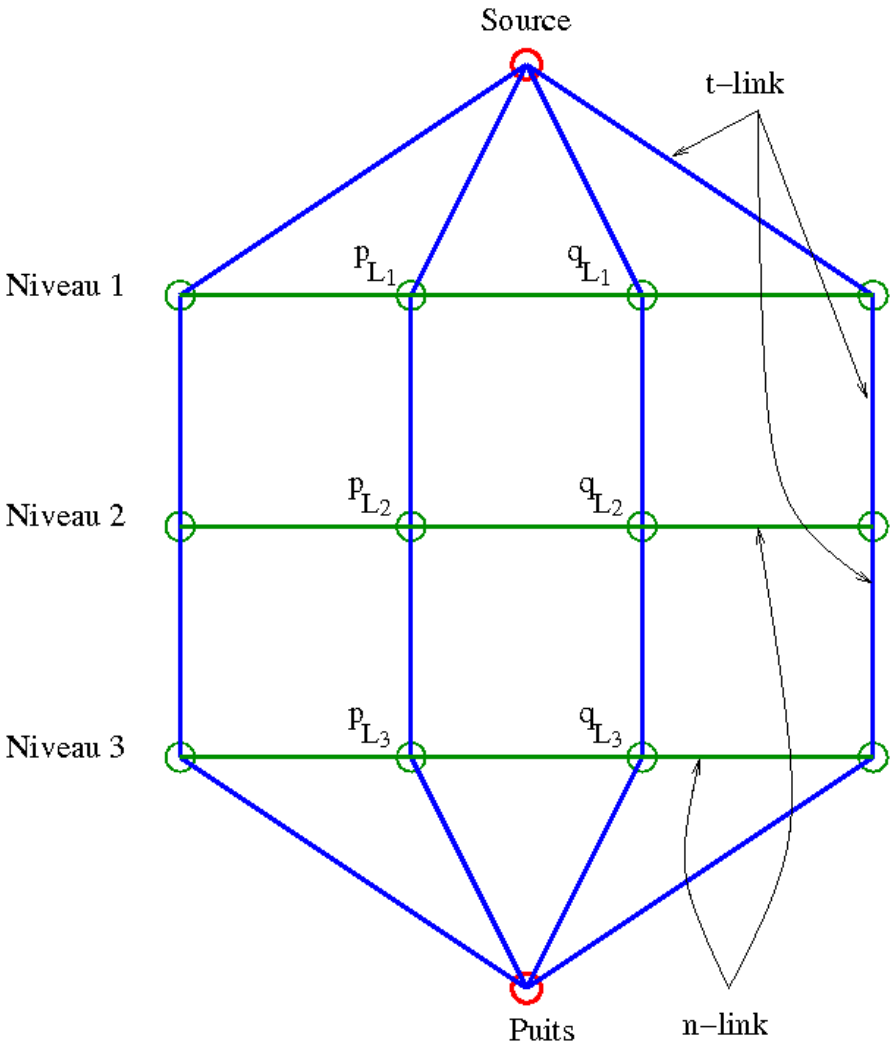


FIG. 3.22 – Graphe Réduit

$$v = W\ H\ \big(N-1\big) + 2 \tag{3.31}$$

$$e = 6\ W\ H\ \big(N-1\big) - 2\ \big(N-1\big)\big(W+H\big) \tag{3.32}$$

Le tableau 3.2 donne les nombres des sites et arêtes dans un graphe réduit construit pour une images de taille 512 x 512 pixels et pour toutes les plages de disparités avec $N = 4$ et 5. La largeur de la plage de disparités n'influence plus le nombre de sites et arêtes. Pour $N = 4$, on obtient moins de 0.8 x 10^6 sites et moins de 0.160 x 10^6 arêtes. Cet exemple illustre clairement le gain énorme en mémoire (RAM).

	$N = 4$	$N = 5$
Nombre de sites (v)	786432	1048576
Nombre d'arêtes (e)	153600	1564672

TAB. 3.2 – Nombre de sites et d'arêtes dans le graphe construit pour une image de taille 512x512 avec le nombre (N) des préchoix égal à 4 ou 5

L'importance du graphe réduit

Les acquis de la construction du graphe réduit sont les suivants :

1. Le graphe devient moins volumineux, le nombre des sites est divisé par exemple par 10 pour une plage de disparités de largeur 40 avec $N = 4$.

2. La taille du graphe ne dépend plus de la largeur de la plage de disparités. Ceci nous permet d'utiliser des plages de disparité plus larges.

3. L'indice de l'arête dans la chaîne des *t-links* associée à un pixel dans le graphe complet correspond à une valeur entière de disparité. Cela ne permet pas d'utiliser des disparités réelles (sous-pixellique). Alors que dans le graphe réduit, la valeur de l'arête est ajoutée comme un attribut à l'arête, et aucune contrainte n'est exigée pour qu'elle soit réelle ou pas. Ainsi, on peut utiliser des valeurs réelles pour la disparité, et en particulier, l'implémentation des disparités sous-pixelliques peut être prise en compte avant la coupure du graphe réduit, contrairement à la coupure du graphe complet.

4. Vu la taille du graphe réduit en nombre de sites et d'arêtes, et sachant que les algorithmes de coupure de graphe sont proportionnels (d'ordre 2 ou 3) à ces nombres (voir la section 3.4), le temps de calcul pour trouver une coupure minimale dans le graphe réduit est beaucoup plus petit que celui dans le graphe complet. Le tableau 3.3 donne des résultats expérimentaux en temps de calcul.

3.6 Implémentation, Résultats expérimentaux

Dans notre travail, nous avons utilisé un banc stéréo précalibré. De plus, notre algorithme de coupure de graphe considère que les images droite et gauche sont rectifiées, c'est-à-dire, les lignes épipolaires sont horizontales et sur la même ligne dans les images gauche et droite. Les images rectifiées issues des caméras précalibrées servent d'entrées pour l'algorithme de coupure

de graphe. De part l'algorithme, il est supposé que la rectification est exacte, c'est-à-dire que la disparité s'applique uniquement sur l'indice colonne des pixels : le pixel (u, v) de l'image gauche est mis en correspondance avec le pixel $(u, v - d)$ de l'image droite. La précision sous-pixellique ne s'applique que sur d.

Pour la représentation de graphe et ses différents algorithmes, nous avons utilisé la librairie *Boost Graph Library* (BGL) [1]. Boost est une librairie *open source* distribuée sous licence *Boost Software License* [2]. La librairie BGL est une librairie en C++ utilisant les principes de programmation générique (*generic programming*) pour construire des structures des données et des algorithmes utilisés dans les calculs des graphes, pour plus des détails voir [Siek *et al.*, 2001]. Parmi les dizaines d'algorithmes pour manipuler les graphes, BGL contient en particulier une implémentation de l'algorithme de flot maximal *Push-Relabel* proposée par Goldberg [Goldberg, 1985], dont la complexité est de $O(v^3)$. Cet algorithme n'est pas le meilleur en performance, mais il nous permet d'évaluer le temps de calcul pour le graphe complet et réduit. De plus, pour le graphe réduit, l'influence de la méthode de résolution du flot maximal est déjà réduite par le fait que le graphe est déjà allégé.

Nous avons implémenté l'algorithme de construction et de coupure minimale dans un graphe complet détaillée dans la section 3.5.3. Ensuite, nous avons implémenté la construction et la coupure du graphe réduit détaillé en section 3.5.4. Pour évaluer les performances de ces méthodes, nous comparons les résultats (images de disparités) et le temps d'exécution. Pour comparer les résultats des deux méthodes, nous avons utilisé deux sources d'images. La première source est l'image *sawtooth* [Scharstein et Szeliski, 2002] illustrée en figure 3.23. Pour cette image, la disparité exacte (réalité du terrain) est donnée par la figure 3.24. L'autre série d'images est composée par des images prises au laboratoire. Pour le graphe réduit, on a utilisé un critère local fondé sur le SAD avec une fenêtre de taille 7 pixels.

Le test est fait sur un P4 à 3 GHz avec 512 Mo de RAM. Le tableau 3.3 donne les valeurs du temps de calcul pour l'image *sawtooth*. On note que pour la taille 434x380, et une plage de disparités $[0, 20]$, nous n'avons pas pu tester la méthode sur l'image avec sa taille totale (explosion de mémoire). Alors qu'avec le graphe réduit, nous arrivons à obtenir l'image de disparité en moins d'une minute. De plus, on a testé les deux méthodes sur une image *sawtooth* de taille réduite 217x190 : on observe un facteur d'environ 40 entre le graphe complet et réduit (150 secondes contre 4 secondes). En outre, nous constatons que, dans le graphe réduit, la valeur de N (le nombre de disparités préchoisies par la méthode locale) influence explicitement le temps d'exécution pour les grandes images, alors que son influence est presque négligeable pour les petites images. Ainsi, nous constatons un gain énorme en temps de calcul acquis par la méthode de graphe réduit. On voit sur cet exemple, que le graphe complet ne peut pas être manipulé sur des machines ordinaires (à mémoire normale), alors que le graphe réduit peut être traité dans un temps acceptable. L'algorithme du graphe réduit est plus rapide, malgré une étape supplémentaire de calcul des coûts locaux (qui peut être négligée devant le temps d'exécution de la coupure du graphe).

Bien qu'on n'ait pas réussi à faire tourner la mise en correspondance stéréoscopique en temps réel par les méthodes de coupure de graphe, les gains en temps de calcul sont très remarquables.

[1] http ://www.boost.org/libs/graph/doc/index.html
[2] http ://www.boost.org/LICENSE_1_0.txt

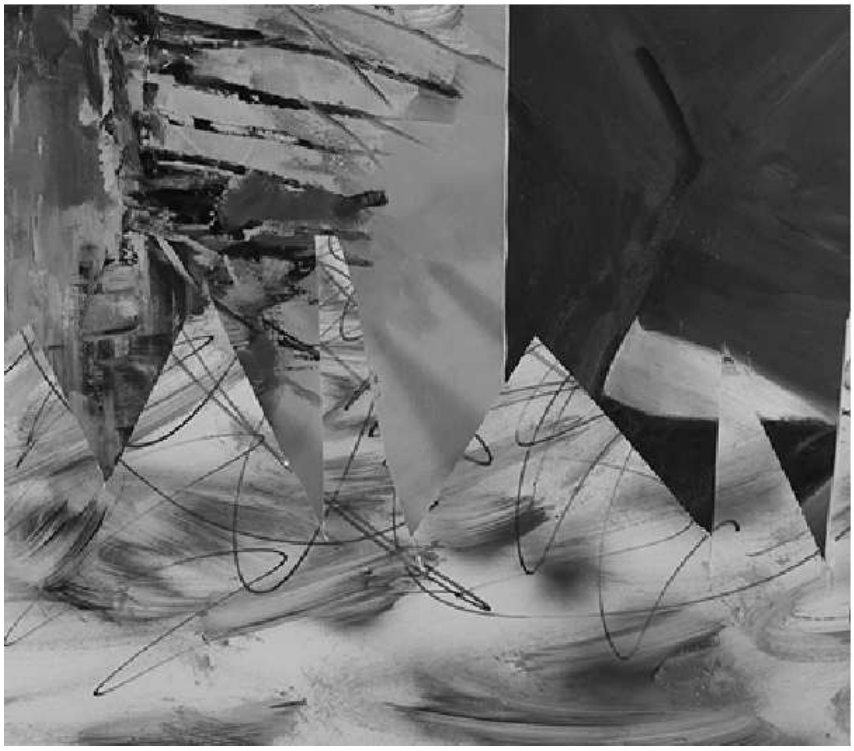


FIG. 3.23 – L'image de sawtooth

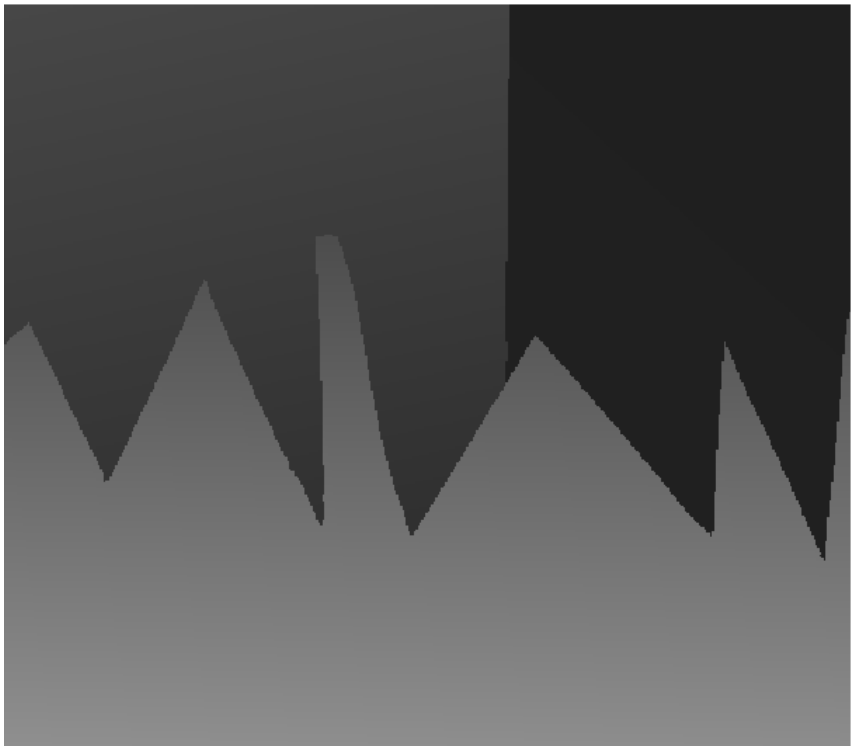


FIG. 3.24 – Sawtooth : image de disparité exacte

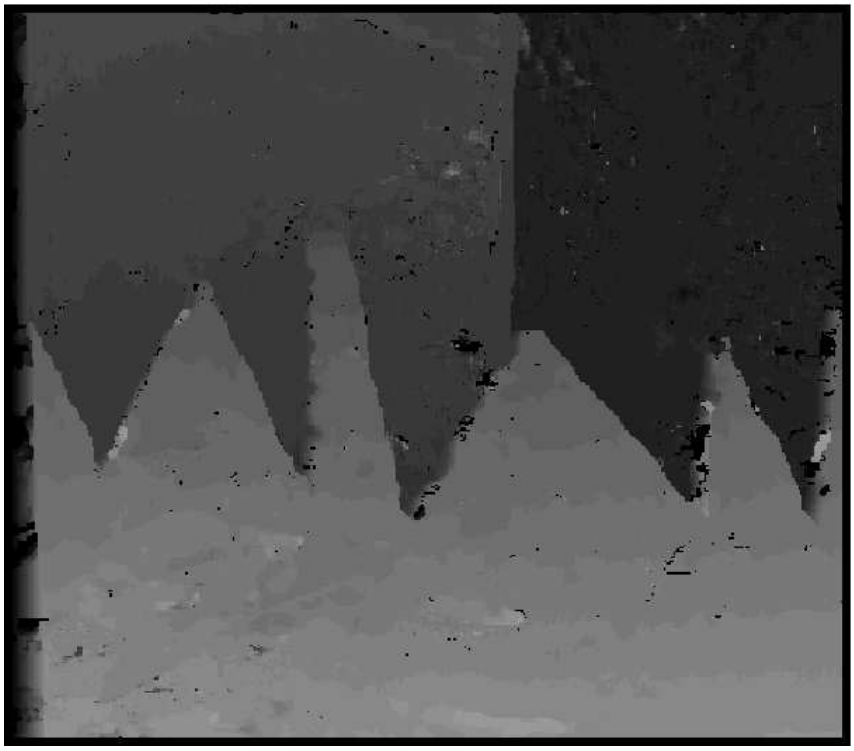


FIG. 3.25 – Sawtooth : l'image de disparité par graphe réduit

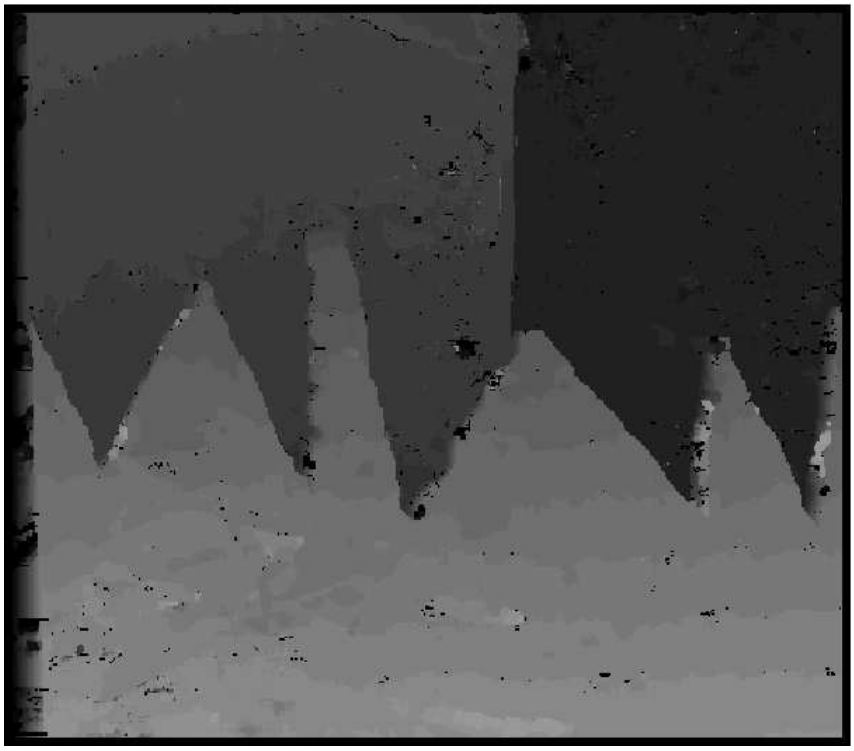


FIG. 3.26 – Sawtooth : l'image de disparité par graphe complet

Implémenter une méthode de coupure de graphe sur des images de tailles 256x256 en 15 secondes au lieu de 30 minutes comme cité par Roy [Roy, 1999], et indépendamment de la largeur de plage de disparités, est un avancement considérable qui mérite d'être mieux développé dans des travaux futurs. Le code que nous avons écrit n'a pas été optimisé pour augmenter la vitesse. De plus, on n'a pas utilisé l'algorithme de flot maximal le plus performant. Améliorer ces performances fait partie des perspectives de notre travail.

Taille de l'image	Graphe Complet	Graphe Réduit	
		N=4	N=5
434x380	NA	15 s	50 s
217x190	150 s	4 s	5 s

TAB. 3.3 – Temps de calcul en secondes pour la mise en correspondance par coupure de Graphe Complet et Graphe Réduit pour une plage de disparités $[0, 20]$

La figure 3.25 représente l'image de disparité obtenue avec l'algorithme de coupure du graphe réduit, et la figure 3.26 est celle obtenue par la coupure du graphe complet. On peut apprécier visuellement la qualité des images de disparité obtenues. La figure 3.27 illustre une autre image de test dont la disparité réalité de terrain est donnée par la figure 3.28. La figure 3.29 donne l'image de disparité obtenue par une méthode locale seulement SAD avec une fenêtre de taille 7 pixels. On peut remarquer les améliorations introduites par les techniques de coupure de graphe réduit (figure 3.30), en particulier, pour éliminer la zone blanche qui est une zone de fausse appariement de la figure 3.29. Notons que pour nos applications en robotique, on vise à utiliser cet algorithme dans un contexte robotique mobile : alors un temps de calcul très réduit sera beaucoup apprécié, même au détriment de la qualité de l'image de disparité.

3.7 Conclusion

Nous avons décrit dans ce chapitre, notre évaluation des méthodes globales de mise en correspondance en stéréovision, fondées sur la coupure de graphe. La combinaison d'une méthode locale, capable de sélectionner pour chaque pixel gauche, un ensemble réduit de correspondants possibles dans l'image droite, et d'une méthode globale, fondée sur la coupure de graphe, nous a permis de réaliser deux avancées :

- Améliorer sensiblement la qualité des images de disparité obtenues uniquement par une méthode locale.
- Éviter l'explosion combinatoire des méthodes de coupure de graphe exécutées sans élagage préalable du graphe.

FIG. 3.27 – L'image de Flowerpots

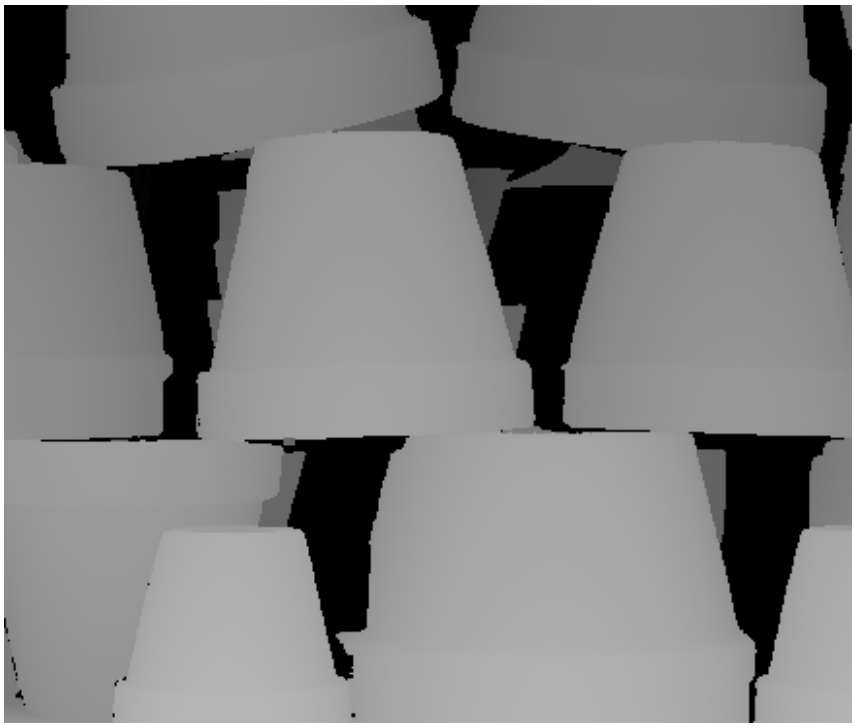


FIG. 3.28 – Flowerpots : image de disparité exacte

FIG. 3.29 – Flowerpots : l'image de disparité par une méthode locale (SAD)

FIG. 3.30 – Flowerpots : l'image de disparité par graphe réduit

Néanmoins, le temps d'exécution, de l'ordre de 15 secondes pour des images de 512x512 n'est pas pour l'instant compatible avec les contraintes temps réel de nos applications en robotique. Nous étudions en conséquence des optimisations de cet algorithme. Notons par ailleurs, que nous travaillons toujours sur des images rectifiées au préalable, et que nous produisons une image de disparité entière. Nous avons montré en section 3.5.4 qu'en principe, le graphe réduit est capable de gérer des disparités réelles (interpolation sous-pixellique de la disparité), mais la mise en oeuvre de cette technique reste à faire pour améliorer la précision de l'image de disparité. Comment adapter une telle méthode globale pour traiter des images non rectifiées reste aussi une piste à explorer. À ce titre, il faudra traduire la contrainte épipolaire par un coût d'appariement, coût plus élevé si le pixel droit associé à un pixel gauche, est plus éloigné de la ligne épipolaire associé à ce pixel.

Malgré les bons résultats de l'algorithme de stéréovision implémenté, il n'est pas exploitable dans l'environnement d'intérieur pour la modélisation 3D, à cause de la présence de grandes surfaces homogènes (mur monocouleur par exemple). Pour cette raison, nous avons décidé d'utiliser un autre capteur pour l'acquisition 3D (un scanner laser 3D), et nous avons en même temps gardé une caméra afin de pouvoir exploiter la richesse d'information qu'elle peut fournir (l'image). Dans le chapitre suivant, nous traitons de la perception de l'environnement par ces deux capteurs et la fusion de leurs données, dans l'optique de la modélisation 3D de l'environnement intérieur.

Chapitre 4

Perception et Fusion de Données Multi-Capteurs

Sommaire

Ce chapitre est consacré à la perception. Nous commençons par présenter les capteurs d'acquisition, qui sont le scanner laser 3D et une caméra. Ensuite nous détaillons le processus de calibrage de ces deux capteurs. La section suivante est consacrée à la segmentation de l'image 3D provenant du scanner laser pour en extraire les surfaces planes. Ensuite dans la section 4.7, nous traitons comment fusionner les données des deux capteurs pour définir un amer ligne 2D. Dans la section 4.8 nous traitons le placage de texture sur les surface planes.

4.1 Notation

On utilise beaucoup de repères dans notre travail, c'est pourquoi il est essentiel d'unifier les notations utilisées. Soient $\mathbf{R}_{1,2}$ et $\mathbf{t}_{1,2}$ la matrice de rotation et le vecteur de translation qui mènent du repère $\mathcal{R}_1$ au repère $\mathcal{R}_2$. Pour un point 3D représenté par $\mathbf{P}_1$ dans le repère $\mathcal{R}_1$ et par $\mathbf{P}_2$ dans le repère $\mathcal{R}_2$, on a :

$$\mathbf{P}_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \tag{4.1}$$

$$\mathbf{P}_1 = \mathbf{R}_{1,2}\,\mathbf{P}_2 + \mathbf{t}_{1,2} \tag{4.2}$$

Les repères utilisés :

- $\mathcal{R}_{sk}$: le repère SICK.
- $\mathcal{R}_c$: le repère Caméra.
- $\mathcal{R}_r$: le repère Robot.
- $\mathcal{R}_w$: le repère Global (monde).
- $\mathcal{R}_\pi$: le repère local lié à l'amer plan.

Les transformations entre ces repères sont données par les matrices et les vecteurs suivants :

- $\mathbf{R}_{r,sk}$ et $\mathbf{t}_{r,sk}$: du repère Robot au repère SICK.
- $\mathbf{R}_{r,c}$ et $\mathbf{t}_{r,c}$: du repère Robot au repère Caméra.
- $\mathbf{R}_{w,r}$ et $\mathbf{t}_{w,r}$: du repère Global au repère Robot.
- $\mathbf{R}_{w,\pi}$ et $\mathbf{t}_{w,\pi}$: du repère Global au repère local de l'amer plan.

La position du robot est définie par $(x_v, y_v, \theta_v)^T$ dans le repère global. Notre robot se déplace sur un sol horizontal, on a donc :

$$\mathbf{R}_{w,r} = \begin{bmatrix} \cos\theta_v & -\sin\theta_v & 0 \\ \sin\theta_v & \cos\theta_v & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

$$\mathbf{t}_{w,r} = \begin{bmatrix} x_v \\ y_v \\ 0 \end{bmatrix} \tag{4.4}$$

4.2 Descriptif du matériel d'acquisition

Notre robot est doté d'un système d'acquisition composé d'un télémètre laser et d'un banc stéréo, et d'autres capteurs que nous n'utilisons pas dans notre travail.

Le télémètre est un LMS 200 *Range Finder* (voir figure 4.1), qui est un laser rotatif balayant un plan dans l'espace. Nous avons choisi une configuration où le plan de balayage est horizontal. De plus, nous avons fixé ce laser sur un axe motorisé par un moteur pas à pas (*stepper*), qui fait des rotations autour d'un axe horizontal. En utilisant les deux balayages, il est possible de capturer une image 3D de l'espace.

Nous avons choisi comme résolution angulaire du balayage par le *scanner* laser la valeur de 0.5° (le résolution angulaire peut prendre les valeurs 0.5° ou 1°), avec un champ de vue de 180° ce qui donne 361 points par *scan*.

Son faisceau laser est dans la bande infrarouge, donc invisible, et sa précision est de l'ordre de 1 *cm* jusqu'à 8 *m* et 2 *cm* au delà. Pour la rotation du scanner autour de l'axe horizontal, nous avons choisi de faire des pas de 0.01 Rad ($\approx 0.57°$) et de pivoter le *scanner* entre -0.3 Rad ($\approx -17°$) et 1.4 Rad ($\approx 80°$), ce qui fait 171 scans. Donc l'image de points 3D est formée de $171 \times 361 = 61731$ points. La figure 4.2 illustre une image de profondeur issue du scanner laser 3D.

FIG. 4.1 – Le scanner laser SICK LMS 200

Le scanner laser a un temps de réponse entre 13 *et* 53 *ms* (selon la résolution angulaire). Mais la rotation autour de l'axe horizontal est réalisée par un moteur pas à pas, ce qui impose un temps de réponse plus important. De plus, il faut assurer que le moteur soit bien arrêté avant de prendre un nouveau scan. En pratique, pour achever un scan 3D complet, il faut environ 60 secondes pour avoir une image de profondeur de 60 mille points. Cette contrainte nous oblige à utiliser la technique "arrêter et tirer" (*Stop and Fire*), et rend pour le moment le temps réel hors de portée.

Le banc stéréo est fixé sur une tourelle *pan/tilt* (voir la figure 2.7 en page 44). Les caméras Flea® sont commercialisées par Ptgrey [7] ; c'est une caméra numérique couleur supportant la norme ieee1394 (voir figure 4.4). Cette caméra possède des pixels carrés permettant de ne pas modifier les proportions des objets perçus. Elle fournit des images au format VGA non compressé. Deux résolutions sont proposées (640 x 480 ou 1024 x 768) et deux modes d'acquisition des images vidéo (640 x 480 à $60FPS$ ou 1024 x 768 à $30FPS$).

Sur notre plate-forme robotique, la caméra Ptgrey Flea est configurée pour fournir des images à la résolution 1024 x 768 pixels.

Nous avons effectué le calibrage de la caméra au moyen de la *Camera Calibration Toolbox for Matlab* [Bouguet, 2007]. Les différents paramètres intrinsèques de la caméra sont donnés au tableau 4.1.

[7] http ://www.ptgrey.com

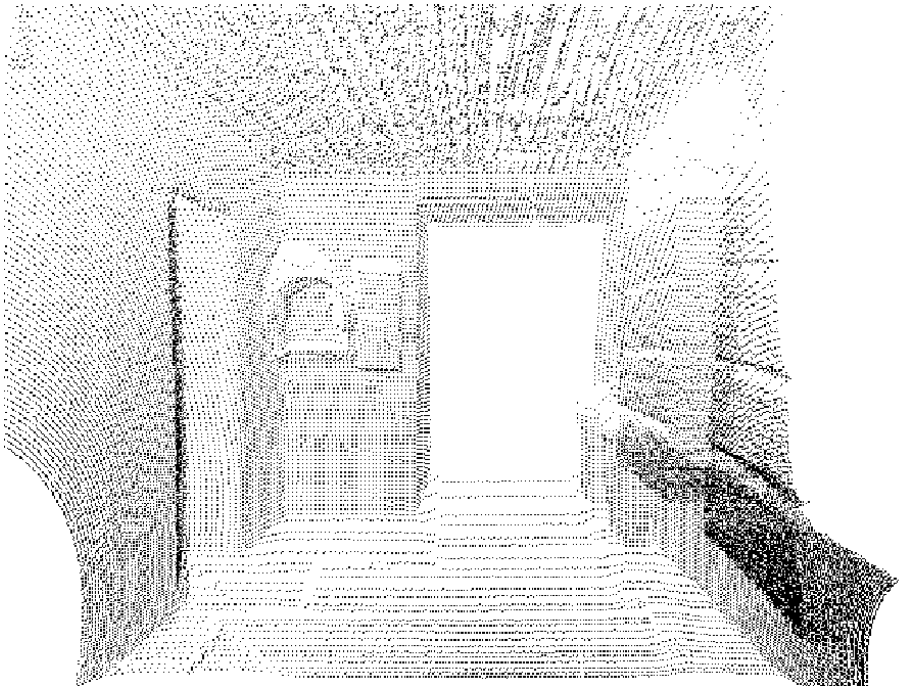


FIG. 4.2 – Image de points 3D provenant du scanner laser

FIG. 4.3 – Image de la même scène prise par la caméra

FIG. 4.4 – La caméra FLEA

u_0	396.278 pixels	v_0	509.582 pixels
α_u	990.015 pixels/m	α_v	990.690 pixels/m

Distorsion radiale : (k_1, k_2, k_3)	0.00164	-0.00568	0

TAB. 4.1 – Paramètres intrinsèques de la caméra

Acquisition des Points 3D

Ce paragraphe détaille les données acquises par le scanner laser 3D, et l'obtention des points 3D avec l'incertitude.

Les mesures rendues par le scanner laser pivotant sont $(Z_\rho, Z_\varphi, Z_\psi)$. Cherchons les coordonnées euclidiennes dans le repère SICK (voir figure 4.5).

$$R_{sick,scan} = \begin{bmatrix} \cos Z_\psi & 0 & \sin Z_\psi \\ 0 & 1 & 0 \\ -\sin Z_\psi & 0 & \cos Z_\psi \end{bmatrix} \tag{4.5}$$

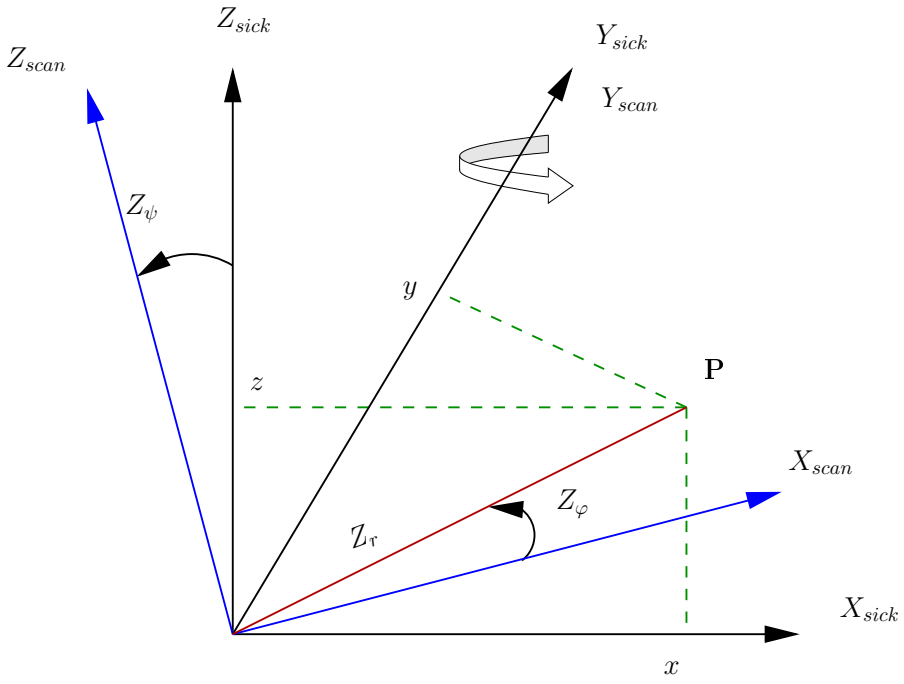


FIG. 4.5 – Le repère SICK et l'acquisition des points 3D

$$\begin{bmatrix} x_{scan} \\ y_{scan} \\ z_{scan} \end{bmatrix} = \begin{bmatrix} Z_\rho \cos Z_\varphi \\ Z_\rho \sin Z_\varphi \\ 0 \end{bmatrix} \tag{4.6}$$

$$\begin{bmatrix} x_{sick} \\ y_{sick} \\ z_{sick} \end{bmatrix} = \begin{bmatrix} \cos Z_\psi & 0 & \sin Z_\psi \\ 0 & 1 & 0 \\ -\sin Z_\psi & 0 & \cos Z_\psi \end{bmatrix} \begin{bmatrix} x_{scan} \\ y_{scan} \\ z_{scan} \end{bmatrix} \tag{4.7}$$

Donc :

$$\begin{bmatrix} x_{sick} \\ y_{sick} \\ z_{sick} \end{bmatrix} = \begin{bmatrix} Z_\rho \cos Z_\varphi \cos Z_\psi \\ Z_\rho \sin Z_\varphi \\ -Z_\rho \cos Z_\varphi \sin Z_\psi \end{bmatrix} \tag{4.8}$$

Propagation de l'incertitude

soit $\Lambda_{\rho\varphi\psi}$ la matrice de covariance de $(Z_\rho, Z_\varphi, Z_\psi)$.

$$J = \begin{bmatrix} \frac{\partial x_{sick}}{\partial Z_\rho} & \frac{\partial x_{sick}}{\partial Z_\varphi} & \frac{\partial x_{sick}}{\partial Z_\psi} \\ \frac{\partial y_{sick}}{\partial Z_\rho} & \frac{\partial y_{sick}}{\partial Z_\varphi} & \frac{\partial y_{sick}}{\partial Z_\psi} \\ \frac{\partial z_{sick}}{\partial Z_\rho} & \frac{\partial z_{sick}}{\partial Z_\varphi} & \frac{\partial z_{sick}}{\partial Z_\psi} \end{bmatrix} \tag{4.9}$$

$$J = \begin{bmatrix} \cos Z_\varphi \cos Z_\psi & -Z_\rho \sin Z_\varphi \cos Z_\psi & -Z_\rho \cos Z_\varphi \sin Z_\psi \\ \sin Z_\varphi & Z_\rho \cos Z_\varphi & 0 \\ -\cos Z_\varphi \sin Z_\psi & Z_\rho \sin Z_\varphi \sin Z_\psi & -Z_\rho \cos Z_\varphi \cos Z_\psi \end{bmatrix} \tag{4.10}$$

donc la matrice de covariance sur le vecteur $(x_{sick}, y_{sick}, z_{sick})^T$ est :

$$\Lambda_{xyz} = J \ \Lambda_{\rho\varphi\psi} \ J^T \tag{4.11}$$

4.3 Architecture Logicielle

Le robot Jido est régi par une instance de l'architecture logicielle LAAS [Alami *et al.*, 1998; Ingrand, 2003]. C'est une architecture hiérarchique à trois niveaux :

- Le niveau décisionnel : Ce plus haut niveau intègre les capacités délibératives de l'agent, par exemple : produire des plans de tâches, reconnaître des situations, détecter des fautes, etc. Dans notre cas, il comprend :
  - ◇ Un exécutif procédural connecté au niveau inférieur auquel il envoie des requêtes qui vont lancer des actions (capteurs/actionneurs) ou démarrer des traitements. Il est responsable de la supervision des actions tout en étant réactif aux événements provenant du niveau inférieur et aux commandes de l'opérateur.
  - ◇ Un planificateur/exécutif temporel chargé de produire et d'exécuter des plans temporels. Ce système doit être réactif et prendre en compte les nouveaux buts ainsi que les échecs d'exécution (échec d'une action et time-out).
- Le niveau fonctionnel : Situé à la base de l'architecture, il est l'interface entre les composants des couches supérieures et la partie physique du système. Il est le siège des fonctions de bases du robot. On y trouve en particulier ses fonctions sensori-motrices, les fonctions de traitement (planificateur de trajectoires, etc) ainsi que les boucles de contrôle (navigation, traitement d'images, capteurs, etc.). Chacune de ces fonctions est encapsulée dans un module. Chaque module offre un ensemble de services, liés à la fonctionnalité du module, accessibles par ses clients via des requêtes.

Nous avons utilisé entre autres les modules suivants : le module SICK, qui gère l'acquisition du scanner laser LMS-200, le module JST, contrôlant la rotation du scanner laser autour de l'axe de rotation, et le module Caméra, pour l'acquisition des images.

De plus, le développement des algorithmes est fait sous Jafar [8] : ("A Framework for Algorithms Development in Robotics") : c'est un environnement de développement C/C++ interactif actuellement utilisé dans les groupes robotiques du LAAS.

4.4 Calibrage Laser-Caméra

Le calibrage des systèmes multi-capteurs est une étape primordiale pour représenter les données provenant des différents capteurs dans un même repère, ce qui est indispensable avant toute analyse ou fusion de ces données. Le calibrage externe d'une caméra et d'un scanner laser est ainsi devenu une nécessité commune pour les systèmes multi-capteurs des robots mobiles actuels. Dans de tels systèmes, une caméra fournit les informations d'intensité sous la forme d'une image, alors que le scanner laser apporte les informations de profondeur sous la forme d'une image de profondeur. Le calibrage externe calcule la transformation rigide entre le repère lié à la caméra et celui lié au scanner laser. La connaissance de cette transformation permet de projeter les points 3D du repère scanner dans le plan image.

Soient $\mathcal{R}_{sk}$ le repère lié au capteur laser et $\mathcal{R}_c$ le repère lié à la caméra. Suivant les mêmes notations utilisées en section 4.1, soient $\mathbf{R}_{c,sk}, \mathbf{t}_{c,sk}$ la matrice de rotation et le vecteur de translation qui amènent du repère caméra au repère laser, de telle sorte que pour un point P de coordonnées $P_c = [x_c, y_c, z_c]^T$ dans $\mathcal{R}_c$, et $P_{sk} = [x_{sk}, y_{sk}, z_{sk}]^T$ dans $\mathcal{R}_{sk}$, on aura :

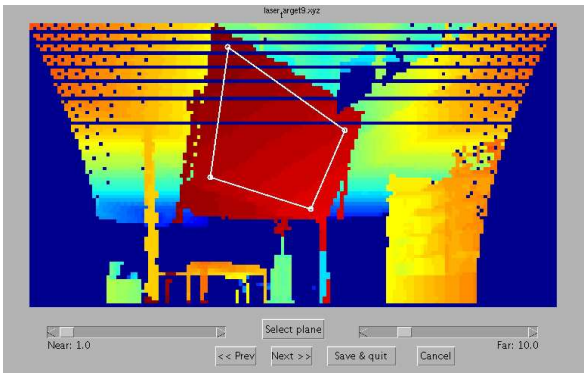
$$P_c = \mathbf{R}_{c,sk}\ P_{sk} + \mathbf{t}_{c,sk} \tag{4.12}$$

L'objectif du calibrage extrinsèque des capteurs est d'estimer la matrice de rotation $\mathbf{R}_{c,sk}$ et le vecteur de translation $\mathbf{t}_{c,sk}$.

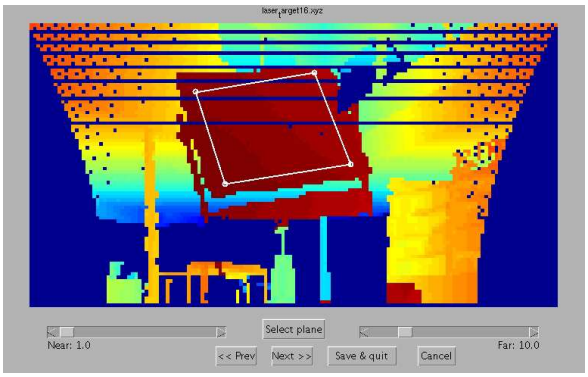
Si le processus de calibrage est bien connu et maîtrisé dans le cas de 2 caméras, le problème est tout autre avec un laser et une caméra. La procédure de calibrage d'une paire laser-caméra est très peu détaillée dans la littérature, gourmande en temps, et reconnue pour être un problème difficile. Pour surmonter ces difficultés, nous avons choisi d'utiliser la boite à outils *Laser-Camera Calibration Toolbox* (LCCT) [Unnikrishnan et Hebert, 2005], qui est un outil graphique sous **Matlab®** permettant de faciliter le calibrage. Le logiciel est disponible gratuitement sur l'internet [9].

Le calibrage démarre par placer en différentes positions une mire pour qu'elle soit perçue à la fois par les deux capteurs. Cette méthode de calibrage utilise une mire de calibrage sous forme d'un échiquier qui est très fréquemment utilisé pour le calibrage intrinsèque d'une caméra. Nous

[8] http ://www.laas.fr/~tlemaire/jafar/
[9] http ://www.cs.cmu.edu/~ranjith/lcct.html

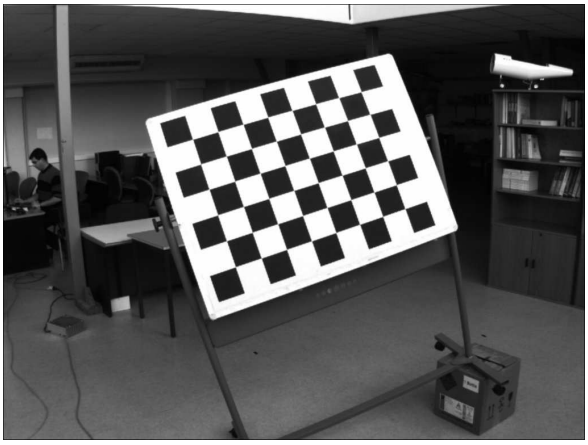


(a) Scène 1 vue par le scanner laser



(b) Scène 2 vue par le scanner laser

(c) Scène 1 : La mire de calibrage vue par la caméra



(d) Scène 2 : La mire de calibrage vue par la caméra

FIG. 4.6 – Outils de calibrage Laser-Caméra

avons utilisé une mire plane de taille aussi grande que possible : dans notre configuration, la taille de l'échiquier est de 100×70 cm^2.

La boite à outils est munie d'une interface interactive avec l'utilisateur permettant de choisir la région de points dans l'image de profondeur qui contient la mire de calibrage. On utilise ensuite une méthode robuste pour trouver les estimations de la normale et la distance à l'origine du plan par rapport au repère du scanner laser. Le calibrage de la caméra est effectuée avec la boite d'outils *Camera Calibration Toolbox for Matlab* [Bouguet, 2007], qui utilise des méthodes de calibrage standard [Faugeras, 1993], [Zhang, 1999]. Elle est aussi disponible sur l'internet [10].

Le principe général est de présenter une mire, ici un échiquier (figure 4.6) devant les deux capteurs laser et caméra. Le télémètre scanne la mire, fournissant un nuage de points 3D, pendant que la caméra capture l'image de l'échiquier. À partir de celle-ci, les paramètres extrinsèques de la caméra sont alors estimés pour déterminer les paramètres $(\mathbf{n_c}, d_c)$ du plan de la mire dans le repère Caméra, tel que $\mathbf{n_c}\ P_c = d_c$ avec d_c, la distance à l'origine du plan de la mire et la caméra

[10] http ://www.vision.caltech.edu/bouguetj/calib_doc/

et $\mathbf{n_c}$ la normale au plan dans R_C.

La figure 4.6 illustre deux prises de vue différentes, et pour chacune on montre la mire vue par les deux capteurs. Une fois la mire repérée dans les deux référentiels pour les différents points de vue, une procédure d'optimisation à deux étapes permet de trouver la transformation entre les deux repères caméra et laser. On trouve dans [Unnikrishnan et Hebert, 2005] plus de détails concernant cet outil et les bases mathématiques utilisées pour son élaboration.

Nous avons choisi de ne pas calibrer la caméra et le laser directement, mais de calibrer la caméra après corrections de la distorsion radiale (voir section 3.2.2). En effet, en faisant le calibrage directement sans correction de la distorsion, l'ajout ultérieure de la distorsion apportera des non-linéarités et compliquera énormément les équations de la fusion des données laser-caméra (voir section 4.7). Ainsi, nous considérons que notre caméra acquiert des images déjà corrigées de la distorsion radiale (voir la section 3.2.2). En effet, nous devons corriger les images avant toute utilisation.

4.5 Importance de la fusion de données

La fusion de données (*Data Fusion*) est un procédé traitant de l'association, la corrélation et la combinaison des données issues de plusieurs capteurs et des informations provenant de plusieurs sources. Elle permet d'aboutir à des estimations affinées des grandeurs physiques et à des conclusions qu'on ne peut obtenir en utilisant une seule source indépendamment des autres. Cette technique est caractérisée par l'amélioration des estimations et des prévisions, et par l'évaluation du besoin en sources additionnelles ou de modification du procédé lui-même, afin d'atteindre des résultats améliorés.

On distingue trois niveaux de fusion [Hall et Llinas, 2001] :

1. La fusion des mesures qui combine directement les mesures de chaque capteur. Elle est aussi appelée fusion amont.

2. La fusion des primitives qui combine les caractéristiques extraites de chaque capteur.

3. La fusion des décisions qui combine l'information après que chaque capteur ait déterminé une sous-décision ou une identité partielle des objets présents. Elle est aussi appelée fusion aval.

Le passage d'une fusion amont à une fusion aval nécessite d'opérer au niveau de chaque capteur une réduction statistique de l'information visant à réduire le débit de transmission (faire au plus tôt des calculs de prétraitement coûteux), tout en préservant l'information pertinente pour le problème à résoudre au niveau central. Cette réduction implique généralement une perte d'information et il convient de s'interroger finement sur l'information à transmettre.

Les données fusionnées ont plusieurs avantages par rapport aux données d'un seul cap-

teur [Hall et Llinas, 2001] :

- Si plusieurs capteurs identiques sont utilisés, combiner les observations provoque des estimations améliorées de la quantité observée. Un avantage statistique est acquis par l'addition de N observations indépendantes. On peut obtenir le même résultat par la combinaison de N observations issues d'un même capteur.

- L'utilisation de l'emplacement relatif des différents capteurs permet d'améliorer le processus d'observation. Par exemple, deux capteurs (caméras) observant le même objet peuvent permettre la détermination de la position 3D de l'objet par une triangulation (stéréovision).

- L'utilisation de plusieurs capteurs améliore l'observabilité. Par exemple, pour un robot équipé par un capteur scanner laser 3D et par une caméra, le scanner laser peut déterminer correctement la distance à un obstacle (un mur par exemple), la caméra peut déterminer les propriétés visuelles de cet obstacle, mais elle ne peut pas déterminer son éloignement. Si les deux observations sont correctement fusionnées, le robot arrive à reconnaître la nature des obstacles dans son environnement. Ainsi, la combinaison des deux capteurs fournit une meilleure localisation par rapport à celle obtenue de chaque capteur agissant tout seul.

Dans notre travail, on a un scanner laser 3D qu'on utilise pour extraire les surfaces planes à partir des images de profondeur (section 4.6). On extrait les segments de lignes 2D dans l'image acquise par une caméra. En combinant les données laser-caméra, on arrive à définir un nouveau type d'amers : ligne 2D attachée à un amer plan. Cet amer ligne 2D peut être vu comme les traits sur une règle : tant que la règle définie le plan dans l'espace 3D, les traits sur elle définissent des informations supplémentaires par rapport au plan de la règle. Utilisant ces deux amers, le robot peut se localiser par rapport au plan et par rapport au traits (amer ligne 2D) sur ce plan. L'importance de tels amers peut être illustrée lorsqu'un robot traverse un couloir long composé de deux murs (et éventuellement par des portes fermées). Quand le robot utilise seulement les amers plans, il sera entre deux plans parallèles ; il ne peut pas se localiser en longitudinal. En utilisant une caméra pour extraire des segments de lignes 2D dans l'image (qui peuvent être les bords d'un poster fixé sur le mur, ou les bords d'une porte), la fusion des données laser-caméra fournit une ligne 2D fixée sur le mur dans une position fixe. Le robot se localise par rapport aux plans et lignes : alors que le plan donne des informations de positionnement latéral, la ligne 2D ajoute des informations de positionnement longitudinal.

4.6 Segmentation de l'Image 3D et Extraction des plans

Les capteurs d'acquisition 3D (télémétrie laser 3D, stéréovision, etc.) fournissent des images qui contiennent des dizaines de milliers de points 3D. L'extraction de plans depuis une image 3D donne un moyen pour réduire la complexité en gardant l'essence des informations contenues dans cette image. Le problème essentiel à résoudre est la segmentation : comment extraire les primitives (*features*) à partir de l'image de profondeur. Segmenter une image est le processus d'étiqueter les points 3D de telle manière que les points appartenant à une même surface plane, portent la même étiquette. Segmenter des images acquises depuis un robot mobile est très difficile, car on

ne connaît pas a priori ce que contient la scène perçue. De plus, le processus de segmentation doit être robuste à la présence d'objets non plans ou d'objets dynamiques (humains partageant le même environnement), aux bruits de mesures.

4.6.1 État de l'art

La segmentation plane a été bien étudiée en infographie dans le but d'afficher en temps réel des modèles complexes, voir Heckbert et Garland [Heckbert et Garland, 1997]. Il existe deux différences majeures entre la robotique et l'infographie. D'une part, les données en robotique sont issues des capteurs et sont donc erronées, alors que les modèles en infographie sont supposés sans erreurs. Ainsi, les algorithmes de simplification en infographie visent à accélérer l'affichage et non à réduire l'erreur. D'autre part, les environnements intérieurs contiennent de grandes surfaces planes verticales (murs) ou horizontales (sol, plafond), alors que les algorithmes en infographie partent généralement de maillages denses.

Horn et Schmidt [Horn et Schmidt, 1995] partent des normales aux plans pour leur algorithme SLAM. Ils extraient les plans en utilisant la Transformée de Hough [Illingworth et Kittler, 1988]. Leur but est d'extraire seulement les plans verticaux, ce qui limite leur méthode. Sequeira et al. [Sequeira *et al.*, 1999] utilisent une méthode hybride de *region-based* et *edge-based* pour la segmentation et alignent les mesures consécutives par un algorithme de *Iterative Closest Point (ICP)*. Liu et al. [Liu *et al.*, 2001] utilisent *Expectation Maximization (EM)* pour créer des cartes 3D de segments plans, mais cette méthode itérative est peu compatible avec les contraintes temps réel propres à la robotique mobile. Kohlhepp et al. [Kohlhepp *et al.*, 2004] extraient des plans en temps réel en utilisant un algorithme de groupement des lignes de balayage. Cet algorithme groupe les segments droits voisins de manière efficace. Mais il exige que les données de chaque ligne de balayage soient segmentées en segments de droite au préalable.

Hähnel et al. [Hähnel *et al.*, 2003] ont proposé un algorithme de simplification adapté au contexte robotique. Dans ce travail, des plans sont extraits en utilisant une approche de type *region-growing* en commençant par un point tiré aléatoirement, et en pilotant le grossissement par les directions des normales. Jan Weingarten [Weingarten, 2006] a proposé une amélioration de cet algorithme en commençant par le grain de région (*region seed*) le plus plat (erreur la plus petite), et en profitant de la structure de données acquises pour éviter l'étape de recherche du plus proche voisin d'une facette en formation. Notre approche est basée sur ces deux travaux, avec une différence dans le paramétrage de plan et la méthode d'estimation de ces paramètres. Récemment, Harati [Harati *et al.*, 2007], a proposé une nouvelle méthode de segmentation de l'image de profondeur basée sur le *bearing angle*, qui est l'angle entre le faisceau laser et la surface réfléchissante.

4.6.2 Représentation d'une surface plane

Dans l'espace euclidien un plan est donné par l'équation :

$$ax + by + cz + d = 0 \tag{4.13}$$

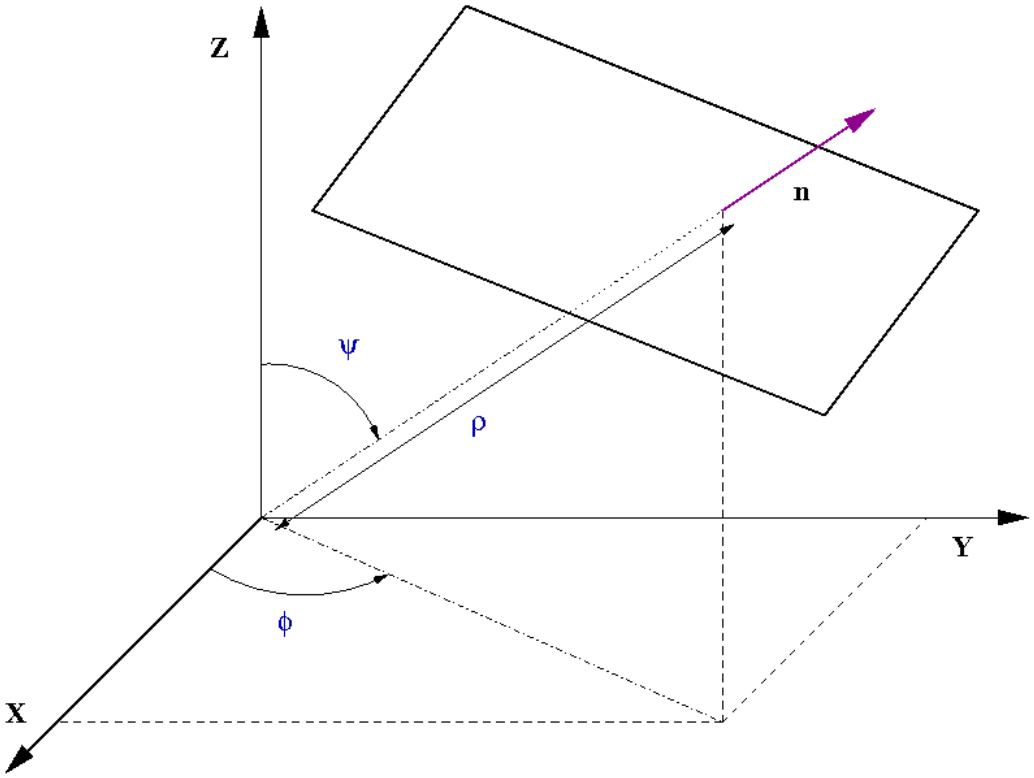


FIG. 4.7 – Représentation du plan

Le vecteur normal est $\vec{n} = (a, b, c)$, et le vecteur normal unitaire $\hat{u} = \frac{\vec{n}}{||\vec{n}||}$. La distance à l'origine est $\rho = \frac{d}{||\vec{n}||}$. La forme dite *Hessian Normal Form* est :

$$\hat{u} \cdot P + \rho = 0 \tag{4.14}$$

où $P = (x, y, z)$ est un point du plan. Cette forme est très utile lorsqu'on va calculer la distance d'un point $P_i = (x_i, y_i, z_i)$ au plan : $D = \hat{u} \cdot P_i + \rho$. Dans ces représentations, le nombre de paramètres est quatre, donc avec une redondance, car un plan peut être exprimé par trois paramètres : la distance à l'origine et deux angles, voir la figure 4.7. Soit φ l'angle que fait la projection de la normale sur le plan OXY avec l'axe $\overrightarrow{OX}$, et soit ψ l'angle que fait la normale avec l'axe $\overrightarrow{OZ}$. L'équation du plan s'écrit :

$$\cos\varphi \, \sin\psi \, x + \sin\varphi \, \sin\psi \, y + \cos\psi \, z + \rho = 0 \tag{4.15}$$

Nous allons utiliser le triplet (ρ, φ, ψ) comme paramétrage du plan, ce qui présente l'avantage d'être une représentation minimale.

4.6.3 Processus d'Estimation

Rappelons que le filtre de Kalman [Kalman, 1960] est un estimateur récursif ; pour estimer l'état courant, seuls l'état précédent et les mesures actuelles sont nécessaires. L'historique des observations et des estimations n'est pas requis. Dans le filtre de Kalman étendu (FKE), les modèles d'évolution et d'observation peuvent être des fonctions non linéaires différentiables. Dans la mesure où ces équations seront aussi exploitées dans le SLAM, nous détaillons ci après les équations pour l'estimation des paramètres d'un plan à partir des coordonnées des points 3D appartenant à ce plan. Le vecteur d'état est composé des paramètres du plan :

$$S_t = \begin{bmatrix} \rho_t \\ \varphi_t \\ \psi_t \end{bmatrix} \tag{4.16}$$

Notons par $P_{t|k}$ la matrice de covariance à l'instant t connaissant toutes les mesures jusqu'à l'instant k.

Prédiction

Dans ce cas simple, nous n'avons pas de modèle de dynamique, ou d'évolution de l'état. La prédiction est donc l'état précédent.

$$\hat{S}_{t|t-1} = \hat{S}_{t-1|t-1} \tag{4.17}$$

$$P_{t|t-1} = P_{t-1|t-1} \tag{4.18}$$

Mesure

On considère chaque point 3D qui appartient au plan comme une observation. La fonction de mesure est implicite, entre état du plan S_t et un point 3D $Z^{'}$ (vecteur de mesures) :

$$\mathbf{h}(S_t, Z^{'}) = \cos\varphi \sin\psi \; x + \sin\varphi \sin\psi \; y + \cos\varphi \; z + \rho = 0 \tag{4.19}$$

Si on note :

- $Z^{'}$: vecteur de mesures (les variables).

- Z_t : vecteur de mesures réelles effectuées à l'instant t.

$$Z_t = \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} \tag{4.20}$$

où (x_t, y_t, z_t) sont les mesures des coordonnées d'un point du plan.

Au voisinage de $(\hat{S}_{t|t-1}, Z_t)$:

$$\begin{aligned} \mathbf{h}(S_t, Z') = \quad & \mathbf{h}(\hat{S}_{t|t-1}, Z_t) + \\ & + \frac{\partial \mathbf{h}}{\partial Z'}\Big|_{(\hat{S}_{t|t-1}, Z_t)} (Z' - Z_t) + \\ & + \frac{\partial \mathbf{h}}{\partial S_t}\Big|_{(\hat{S}_{t|t-1}, Z_t)} (S_t - \hat{S}_{t|t-1}) \end{aligned} \tag{4.21}$$

d'où :

$$\begin{aligned} & -\mathbf{h}(\hat{S}_{t|t-1}, Z_t) + \frac{\partial \mathbf{h}}{\partial S_t}\Big|_{(\hat{S}_{t|t-1}, Z_t)} \hat{S}_{t|t-1} \quad = \\ & \frac{\partial \mathbf{h}}{\partial S_t}\Big|_{(\hat{S}_{t|t-1}, Z_t)} S_t + \frac{\partial \mathbf{h}}{\partial Z'}\Big|_{(\hat{S}_{t|t-1}, Z_t)} (Z' - Z_t) \end{aligned} \tag{4.22}$$

Soient :

$$Y_t = -\mathbf{h}(\hat{S}_{t|t-1}, Z_t) + \frac{\partial \mathbf{h}}{\partial S_t}\Big|_{(\hat{S}_{t|t-1}, Z_t)} \hat{S}_{t|t-1} \tag{4.23}$$

$$J_S = \frac{\partial \mathbf{h}}{\partial S_t}\Big|_{(\hat{S}_{t|t-1}, Z_t)} \tag{4.24}$$

$$J_Z = \frac{\partial \mathbf{h}}{\partial Z'}\Big|_{(\hat{S}_{t|t-1}, Z_t)} \tag{4.25}$$

h est une fonction scalaire. S_t et Z' sont de degré 3, donc J_S et J_Z sont des vecteurs lignes 1 x 3 :

$$\begin{aligned} J_S(1,1) = \quad & -\sin\hat{\varphi}_{t|t-1} \sin\hat{\psi}_{t|t-1}\, x_t + \\ & + \cos\hat{\varphi}_{t|t-1} \sin\hat{\psi}_{t|t-1}\, y_t \end{aligned} \tag{4.26}$$

$$\begin{aligned} J_S(1,2) = \quad & \cos\hat{\varphi}_{t|t-1}\cos\hat{\psi}_{t|t-1}\,x_t + \\ & \sin\hat{\varphi}_{t|t-1}\cos\hat{\psi}_{t|t-1}\,y_t - \\ & \sin\hat{\psi}_{t|t-1}\,z_t \end{aligned} \tag{4.27}$$

$$J_S(1,3) = 1 \tag{4.28}$$

$$J_Z^T = \begin{bmatrix} \cos\hat{\varphi}_{t|t-1}\sin\hat{\psi}_{t|t-1} \\ \sin\hat{\varphi}_{t|t-1}\sin\hat{\psi}_{t|t-1} \\ \cos\hat{\psi}_{t|t-1} \end{bmatrix} \tag{4.29}$$

L'erreur de mesure est $(Z^{'} - Z_t)$ et la matrice de covariance de mesure est notée Λ_{η_t} :

$$\eta_t = Z^{'} - Z_t \tag{4.30}$$

$$\Lambda_{\eta_t} = \mathrm{E}\big(\eta_t \eta_t^T\big) \tag{4.31}$$

Donc :

$$\Lambda_{\xi_t} = J_z\ \Lambda_{\eta_t}\ J_z^T \tag{4.32}$$

Λ_{ξ_t} est un scalaire.

Mise à jour (Update)

L'innovation ν_t

$$\begin{aligned} \nu_t \quad & = Y_t - J_S\ \hat{S}_{t|t-1} \\ & = -\mathbf{h}\left(\hat{S}_{t|t-1}, Z_t\right) \end{aligned} \tag{4.33}$$

Notons K_t le gain du filtre :

$$K_t = P_{t|t-1}\ J_S^T\ \left(J_S\ P_{t|t-1}\ J_S^T + \Lambda_{\xi_t}\right)^{-1} \tag{4.34}$$

donc la mise à jour de l'état :

$$\begin{aligned} \hat{S}_{t|t} &= \hat{S}_{t|t-1} + K_t \left(Y_t - J_S \hat{S}_{t|t-1} \right) \\ &= \hat{S}_{t|t-1} - K_t \, \mathbf{h} \left(\hat{S}_{t|t-1}, Z_t \right) \end{aligned} \tag{4.35}$$

et la mise à jour de la matrice de covariance :

$$P_{t|t} = \left(I - K_t \, J_S \right) P_{t|t-1} \tag{4.36}$$

4.6.4 Segmentation par Transformation de Hough

La transformation de Hough convertit le problème complexe de la reconnaissance de modèle en une simple recherche d'un pic dans l'espace des paramètres. Une étude bien détaillée sur la transformation de Hough et ses utilisations est présentée dans [Illingworth et Kittler, 1988]. Pour la segmentation de l'image de profondeur, l'idée de la transformation de Hough est la suivante : on divise l'espace de paramètres en cellules selon des pas d'échantillonnage, de telle sorte que chaque cellule corresponde à une combinaison des paramètres du modèle recherché. Ensuite, on fait voter les points 3D pour chacune des cellules, et si un point vérifie le modèle définie par la combinaison des paramètres d'une cellule, on ajoute ce point à cette cellule, et ainsi de suite pour tous les points. En accumulant toutes les valeurs pour tous les points dans un espace dit espace de paramètres (Espace de Hough), on peut trouver les plans qui satisfont au mieux les données, ce qui est traduit par les cellules qui ont localement le plus de votes.

Transformation de Hough 3D :

Considérant l'équation 4.15 du plan, les paramètres sont ρ, φ, ψ. Un jeu de paramètres permet de faire un bon choix des valeurs d'échantillonnage $\Delta_\rho, \Delta_\varphi, \Delta_\psi$ qui satisfait un temps de calcul raisonnable et de bonne précision. En balayant toutes les valeurs possibles de φ, ψ ($0 \leq \varphi \leq 360°$ et $0 \leq \psi \leq 180°$), on trouve les valeurs de ρ qui satisfont l'équation 4.15, et on répète cette opération pour tous les points. La cellule qui a accumulé le plus de votes correspond au plan qui passe par le plus grand nombre de points, c'est le plan principal dans la scène. Ainsi, on peut retirer les points qui appartiennent au plan principal et on répète l'algorithme pour trouver le deuxième plan, et ainsi de suite. Mais on voit clairement qu'il y aura une répétition inutile. D'où l'importance de l'algorithme nommé *Progressive Probabilistic Hough Transformation* [Matas *et al.*, 1998]. Dans cet algorithme, on fait voter les points jusqu'à ce qu'une cellule dépasse un seuil prédéfini de votes. Une fois ce seuil est dépassé (ceci correspond à un premier plan), on retire les votes effectués aux autres plans par les points qui ont voté pour lui. De plus, on enlève de l'ensemble des points non encore utilisés tous les points qui satisfont l'équation de ce plan. L'algorithme s'arrête lorsqu'il n'existe plus de points non utilisés. Cet algorithme a une complexité inférieure au premier [Matas *et al.*, 1998], et il permet d'éviter les votes multiples, c'est-à-dire, un point vote pour plusieurs plans, parce que lorsqu'un point appartient à un plan, ses votes pour les autres plans sont retirés.

Nous avons implémenté cette approche. Cette technique a donné de bons résultats pour l'extraction du plan principal (la plus grande surface plane dans la scène). En revanche, elle est incapable d'extraire les petites surfaces. De plus, nous avons rencontré un problème de plusieurs maximums locaux pour le même plan, ce qui conduit l'algorithme à donner plusieurs fois des surfaces très proches du plan principal, et en conséquence de ne pas être capable de suivre les autres plans. Enfin, le choix des pas d'échantillonnage joue un rôle important dans le processus. De petits pas sont à l'origine de la distribution des points 3D d'un même plan sur plusieurs plans très similaires, et donc nous rencontrons une difficulté à trouver les maximums locaux. En revanche, avec de grands pas d'échantillonnage, nous perdons de la précision.

4.6.5 Segmentation par Region-growing

Soient : $\mathrm{V} = \{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_{\mathrm{N_v}}\}$ l'ensemble de points 3D, et $\mathbf{N} = \{\mathbf{n}_1, \mathbf{n}_2, \ldots, \mathbf{n}_{N_\mathrm{v}}\}$ les normales estimées en ces points. Pour chaque point, nous calculons le plan local (en utilisant le point et ses 8 voisins) par la méthode des moindres carrés, et calculons l'erreur correspondante, soit $\mathbf{E} = \{e_1, e_2, \ldots, e_{N_\mathrm{v}}\}$ l'ensemble des erreurs. Dans la boucle principale de l'algorithme 4.1, nous choisissons le point (non encore traité) qui a l'erreur locale la plus petite. Nous utilisons les paramètres du plan local pour initialiser le filtre de Kalman étendu, puis on fait appel à l'algorithme de *region growing*. On utilise une file d'attente pour garder les points pendant l'étape de recherche (*bradth-first*). Dans l'algorithme 4.2, on prend le premier point dans la file d'attente, puis on cherche ses voisins non encore traités. On effectue une série de tests : distance entre les deux points, distance entre le point et le plan estimé, distance entre la normale au plan et la normale au point, auxquels on peut ajouter le test du χ_2 avec la distance de Mahalanobis. Le point qui satisfait tous ces tests est candidat pour rejoindre le plan, et éventuellement ses voisins, que l'on rajoute à la file d'attente. On fait tourner le filtre avec comme mesure, les coordonnées de ce nouveau point. L'algorithme 4.1 donne le pseudo-code de l'algorithme de segmentation, et l'algorithme 4.2 détaille la boucle de grossissement du plan.

Les paramètres de cet algorithme jouent un rôle important dans sa performance :

- δ_{PtPt} : La distance entre un point du plan et un point voisin. Cette distance permet de rejeter les points voisins qui n'appartiennent pas au plan à cause d'une différence de profondeur supérieure à un seuil donné par δ_{PtPt}.

- δ_{PtPln} : La distance entre un point et le plan. Cette distance est calculée très rapidement si l'on utilise la forme Hessian (voir section 4.6.2). Le seuil δ_{PtPln} permet de ne considérer que les points très proches du plan.

- δ_{NN} : L'angle entre la normale au point considéré et la normale au plan. Si l'angle entre ces deux normales est supérieur à un seuil δ_{NN}, c'est que le point avec ses voisins forment localement un plan différent du plan considéré.

De plus, à la fin de la segmentation par *Region-growing*, une étape de traitement est nécessaire pour fusionner les régions supportées par un même plan, et pour supprimer les petits régions. En effet, on peut avoir deux régions planes pour un même plan, cela peut être dû à la présence

Algorithme 4.1 : Segmentation Plane par *Region-growing*

Paramètres :

$\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{N_\mathrm{v}}\}$: L'ensemble des points 3D

$\mathbf{N} = \{\mathbf{n}_1, \mathbf{n}_2, \ldots, \mathbf{n}_{N_\mathrm{v}}\}$: L'ensemble des normales

$\mathbf{E} = \{e_1, e_2, \ldots, e_{N_\mathrm{v}}\}$: L'ensemble des erreurs de *fitting* des plans locaux

$\mathbf{S} = \{s_1, s_2, \ldots, s_{N_\mathrm{S}}\}$: L'ensemble des plans trouvés

Initialisation :

début

- calculer les normales
- estimer les plans locaux
- $S \leftarrow \phi$
- $\mathbf{q} \leftarrow \phi$
- $N_t \leftarrow 0$: Le nombre des points traités

fin

tant que $N_t \leq N_\mathrm{v}$ **faire**

- $\mathrm{v}_{min} \leftarrow$ trouve le point non traité ayant l'erreur locale minimale
- Initialiser le filtre en utilisant le plan local du point v_{min}
- $\mathbf{q} \leftarrow \mathrm{v}_{min}$
- growRegion($\mathbf{q}$)
- $S \leftarrow$ ajouter le plan

fin

Algorithme 4.2 : Region Growing

```
tant que q ≠ φ faire
    v_f ← getFirst(q)
    N_{v_f} : chercher l'ensemble des voisins valides non traités de v_f
    pour v_i ∈ N_{v_f} faire
        δ_PtPt ← calculer la distance entre les deux points v_f et v_i
        si δ_PtPt > τ_PtPt alors
            continue
        fin
        δ_PtPln ← calculer la distance entre le point v_i et le plan
        si δ_PtPln > τ_PtPln alors
            continue
        fin
        δ_NN ← calculer l'angle entre la normale du point et la normale au plan
        si δ_NN > τ_NN alors
            continue
        fin
        Push(q, v_i)
        /* Maintenant on peut fusionner le point avec le plan
        en utilisant le point comme une nouvelle observation du plan
        et en mettant à jour le plan par EKF */
        updateFilter(v_i)
    fin
    PopFirst(q)
fin
```

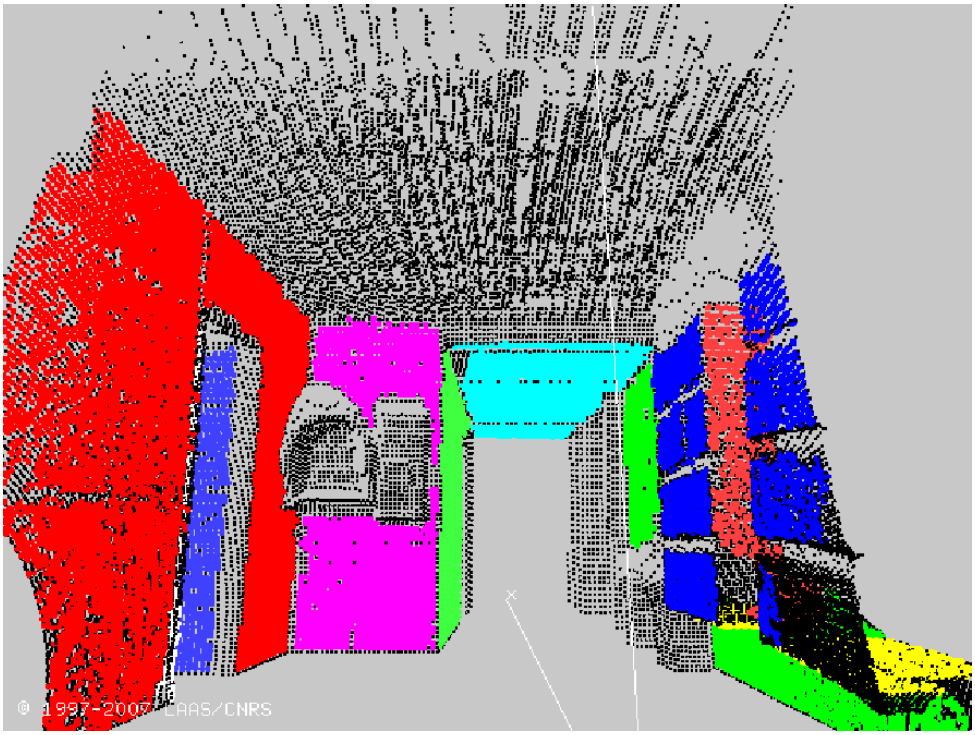


FIG. 4.8 – Segmentation de l'image de profondeur de la scène représentée par les figures 4.2 et 4.3 (page 101) : les points de chaque plan ont la même couleur et les points noirs n'appartiennent à aucun plan

d'un obstacle ou par une rupture causée par une porte ouverte. La taille d'une région plane est proportionnelle au nombre des points qu'elle contient. Ainsi, on peut supprimer les régions planes dont le nombre de points est inférieur à un seuil prédéfini. Par exemple, les résultats de segmentation de la scène représentée par les figures 4.2 et 4.3 (page 101) sont illustrés en figure 4.8. Les façades des boites aux lettres par exemple sont disjointes, mais portées par le même plan ; elles ont ainsi été considérées par l'algorithme de segmentation comme différentes régions du même plan, et en conséquence ont la même couleur (bleue).

Avec un jeu empirique de paramètres, on arrive à segmenter les images de profondeur et à extraire des plans avec une exactitude largement suffisante pour construire une carte stochastique (voir le chapitre suivant). La figure 4.9 illustre un autre résultat de l'algorithme de segmentation. Dans cette figure on peut voir deux plans sur le coté droit, le premier est le mur, alors que le deuxième est le *porte-affiches* qui a une épaisseur de moins de 4 cm et fixé sur le mur.

4.6.6 Choix du repère local lié au plan

Soit un plan $\mathcal{P}$ défini par son vecteur normal $\mathbf{n}_w = \{\varphi_w, \psi_w\}$ et sa distance à l'origine ρ_w dans un repère global $\mathcal{R}_w$. On cherche à définir un repère orthonormé lié à ce plan. Choisissons la projection de l'origine O_w sur le plan $\mathcal{P}$ comme l'origine O_p du repère plan, et l'axe Z_p parallèle au vecteur normal $\mathbf{n}_w$. Il reste à définir l'axe X_p. Soient $\overrightarrow{i_w}, \overrightarrow{j_w}, \overrightarrow{k_w}$ les vecteurs unitaires des axes

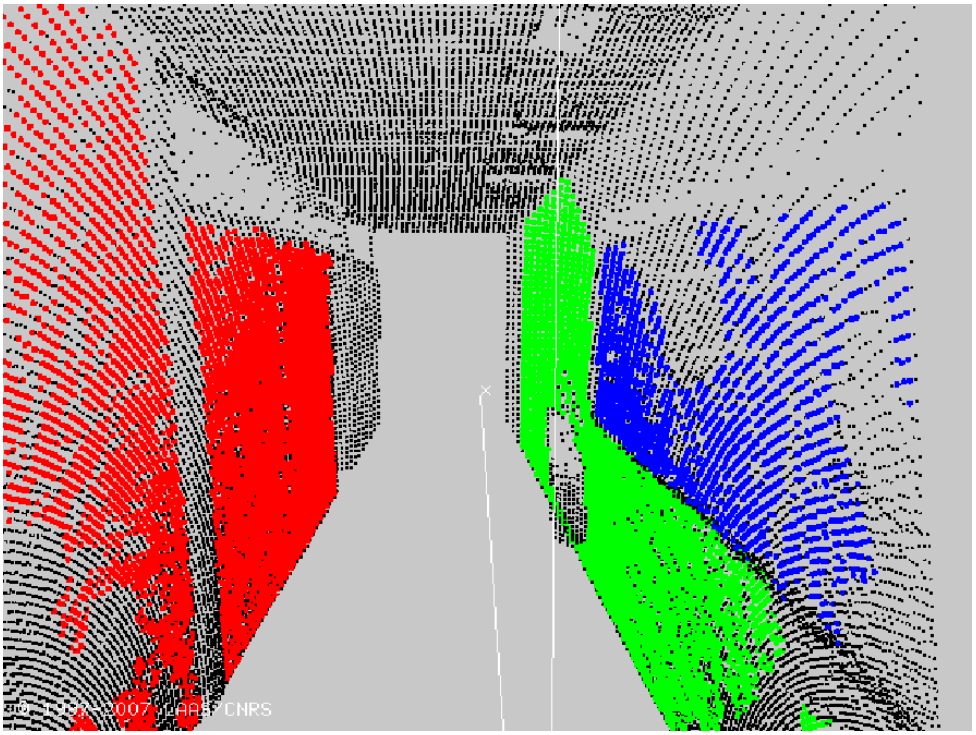


FIG. 4.9 – Segmentation plane d'une image de profondeur : les points de chaque plan ont la même couleur et les points noirs n'appartiennent à aucun plan

FIG. 4.10 – Image de la même scène prise par la caméra : le couloir et le porte-affiches

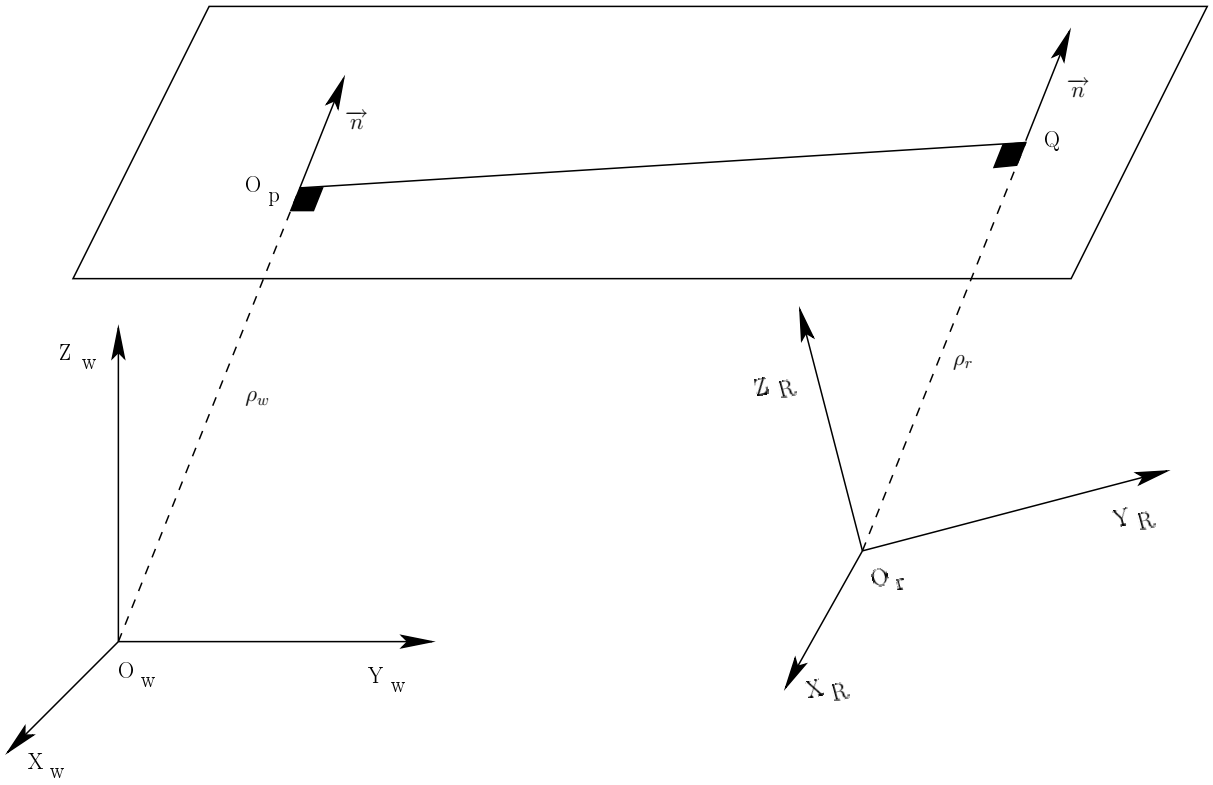


FIG. 4.11 – Un plan et les repères du monde et du robot

O_wX_w, O_wY_w, O_wZ_w respectivement, et $\overrightarrow{i_p}, \overrightarrow{j_p}, \overrightarrow{k_p}$ les vecteurs unitaires des axes recherchés O_pX_p, O_pY_p, O_pZ_p respectivement. Soit

$$\overrightarrow{i_p} = \begin{bmatrix} \sin\varphi_w \\ -\cos\varphi_w \\ 0 \end{bmatrix} \tag{4.37}$$

Ce vecteur peut être interprété comme le vecteur unitaire le long de la ligne d'intersection entre le plan $\mathcal{P}$ et le plan $Z_w = 0$ (s'ils ne sont pas parallèles) :

Or, on connaît l'axe $Z_p = 0$ avec son vecteur unitaire $\overrightarrow{k_p}$:

$$\overrightarrow{k_p} = \begin{bmatrix} \cos\varphi_w \sin\psi_w \\ \sin\varphi_w \sin\psi_w \\ \cos\psi_w \end{bmatrix} \tag{4.38}$$

donc, le vecteur unitaire l'axe OY_p, $\overrightarrow{j_p}$ est :

$$\overrightarrow{j_p} = \overrightarrow{k_p} \wedge \overrightarrow{i_p} = \begin{vmatrix} \cos\varphi_w \sin\psi_w \\ \sin\varphi_w \sin\psi_w \\ \cos\psi_w \end{vmatrix} \wedge \begin{vmatrix} \sin\varphi_w \\ -\cos\varphi_w \\ 0 \end{vmatrix} \tag{4.39}$$

$$\overrightarrow{j_p} = \begin{bmatrix} \cos\varphi_w \cos\psi_w \\ \sin\varphi_w \cos\psi_w \\ -\sin\psi_w \end{bmatrix} \tag{4.40}$$

On peut écrire la matrice de rotation entre le repère plan et repère global :

$$\mathbf{R}_{wp} = \begin{bmatrix} \sin\varphi_w & \cos\varphi_w \cos\psi_w & \cos\varphi_w \sin\psi_w \\ -\cos\varphi_w & \sin\varphi_w \cos\psi_w & \sin\varphi_w \sin\psi_w \\ 0 & -\sin\psi_w & \cos\psi_w \end{bmatrix} \tag{4.41}$$

et le vecteur de translation est donné par :

$$\mathbf{t}_{wp} = \rho_w \begin{bmatrix} \cos\varphi_w \sin\psi_w \\ \sin\varphi_w \sin\psi_w \\ \cos\psi_w \end{bmatrix} \tag{4.42}$$

4.7 Amers lignes 3D par fusion des données Laser et Caméra

On peut extraire les segments de droites 2D dans l'image. Ces segments peuvent être interprétés comme les projections des droites 3D sur le plan image. Afin d'améliorer le modèle de l'environnement construit par l'approche SLAM (voir chapitre suivant), nous voulons utiliser les droites 3D comme amers dans la carte stochastique. Or, pour définir une droite 3D on a besoin de définir deux plans. En utilisant une caméra, on obtient un des deux plans. L'autre plan peut être fourni par la segmentation de l'image 3D issue du scanner laser 3D (section 4.6). En fusionnant les données des deux capteurs on peut donc extraire les droites 3D dans la scène, celles qui sont portées par les plans extraits depuis les données laser. Pour une raison d'optimisation de la représentation, nous considérons les droites 3D comme des droites 2D attachées à un plan qui les contient. Le plan porteur est défini par les données laser, comme expliqué en section 4.6. Cette

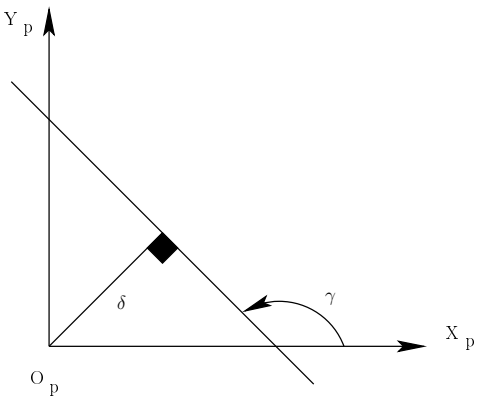


FIG. 4.12 – Représentation minimale d'une droite 2D dans un plan

représentation semble raisonnable, parce que les droites 3D peuvent être des coins (intersections de deux murs) ou des bords d'un poster fixé sur un mur, et dans les deux cas, une droite 2D dans le repère local lié au plan porteur suffit pour définir totalement la droite 3D. On définit ainsi la **ligne 2D attachée à un amer plan** comme un autre type d'amers qui seront utilisés pour construire la carte hétérogène (voir section 5.5).

4.7.1 Extraction des Lignes 2D dans l'Image

Nous utilisons une méthode traditionnelle pour extraire les segments lignes dans l'image. On commence par un filtre de Canny pour extraire les contours, puis on utilise une approximation polynomiale pour estimer la ligne passant par les points contigus des contours. Une étape de traitement supplémentaire est nécessaire afin de fusionner les segments similaires et supprimer les petits segments.

4.7.2 Plan d'Interprétation

Soit l_I un segment 2D dans l'image. Le plan d'interprétation associé est le plan qui passe par le segment 2D et le centre optique de la caméra, comme illustré en figure 4.13. Le vecteur normal de ce plan peut être calculé en utilisant seulement les paramètres intrinsèques de la caméra $(\alpha_u, \alpha_v, u_0, v_0)$, et les données images du segment. En effet, soient (δ_I, γ_I) les paramètres de la ligne 2D infinie qui porte le segment de droite 2D, avec δ_I la distance à l'origine et γ_I l'angle avec l'axe des u, comme illustré en figure 4.12. L'équation de la ligne 2D dans le repère image est donnée par :

$$\cos\gamma_I \ u + \sin\gamma_I \ v - \delta_I = 0 \tag{4.43}$$

et en utilisant les coordonnées caméras :

$$\cos\gamma_I \ \left(\alpha_u \frac{x_c}{z_c} + u_0\right) + \sin\gamma_I \ \left(\alpha_v \frac{y_c}{z_c} + v_0\right) - \delta_I = 0 \tag{4.44}$$

On aura :

$$\alpha_u \cos\gamma_I \ x_c + \alpha_v \sin\gamma_I \ y_c + \left(-\delta_I + u_0 \cos\gamma_I + v_0 \sin\gamma_I\right) z_c = 0 \tag{4.45}$$

Le vecteur normal dans le repère caméra est donc donné par :

$$\mathbf{n}_c = \begin{bmatrix} \alpha_u \cos\gamma_I \\ \\ \alpha_v \sin\gamma_I \\ \\ -\delta_I + u_0 \cos\gamma_I + v_0 \sin\gamma_I \end{bmatrix} \tag{4.46}$$

et bien sûr puisque le plan d'interprétation passe par le centre optique de la caméra, la distance à l'origine est nulle.

$$d_c = 0 \tag{4.47}$$

Notons le plan d'interprétation dans le repère Robot, le repère Global, le repère local de l'amer plan respectivement par : $(\mathbf{n}_r, d_r)$, $(\mathbf{n}_w, d_w)$ et $(\mathbf{n}_p, d_p)$. On note aussi :

$$\begin{cases} \mathbf{n}_c &= [n_{c,x} \quad n_{c,y} \quad n_{c,z}]^T \\ \mathbf{n}_r &= [n_{r,x} \quad n_{r,y} \quad n_{r,z}]^T \\ \mathbf{n}_w &= [n_{w,x} \quad n_{w,y} \quad n_{w,z}]^T \\ \mathbf{n}_p &= [n_{p,x} \quad n_{p,y} \quad n_{p,z}]^T \end{cases} \tag{4.48}$$

4.7.3 La droite 2D dans le repère lié à l'amer plan

Le plan d'interprétation associé au segment l_I, représenté par le vecteur normal et la distance à l'origine n_p, d_p dans le repère local lié à l'amer plan est donné par :

$$\begin{cases} \mathbf{n}_p = & \mathbf{R}_{w,p}^T \ \mathbf{R}_{w,r} \ \mathbf{R}_{r,c} \ \mathbf{n}_c \\ \\ d_p = & d_c - \mathbf{t}_{r,c}^T \mathbf{R}_{r,c} \ \mathbf{n}_c - \mathbf{t}_{w,r}^T \mathbf{R}_{w,r} \mathbf{R}_{r,c} \ \mathbf{n}_c \\ & + \mathbf{t}_{w,p}^T \mathbf{R}_{w,r} \mathbf{R}_{r,c} \ \mathbf{n}_c \end{cases} \tag{4.49}$$

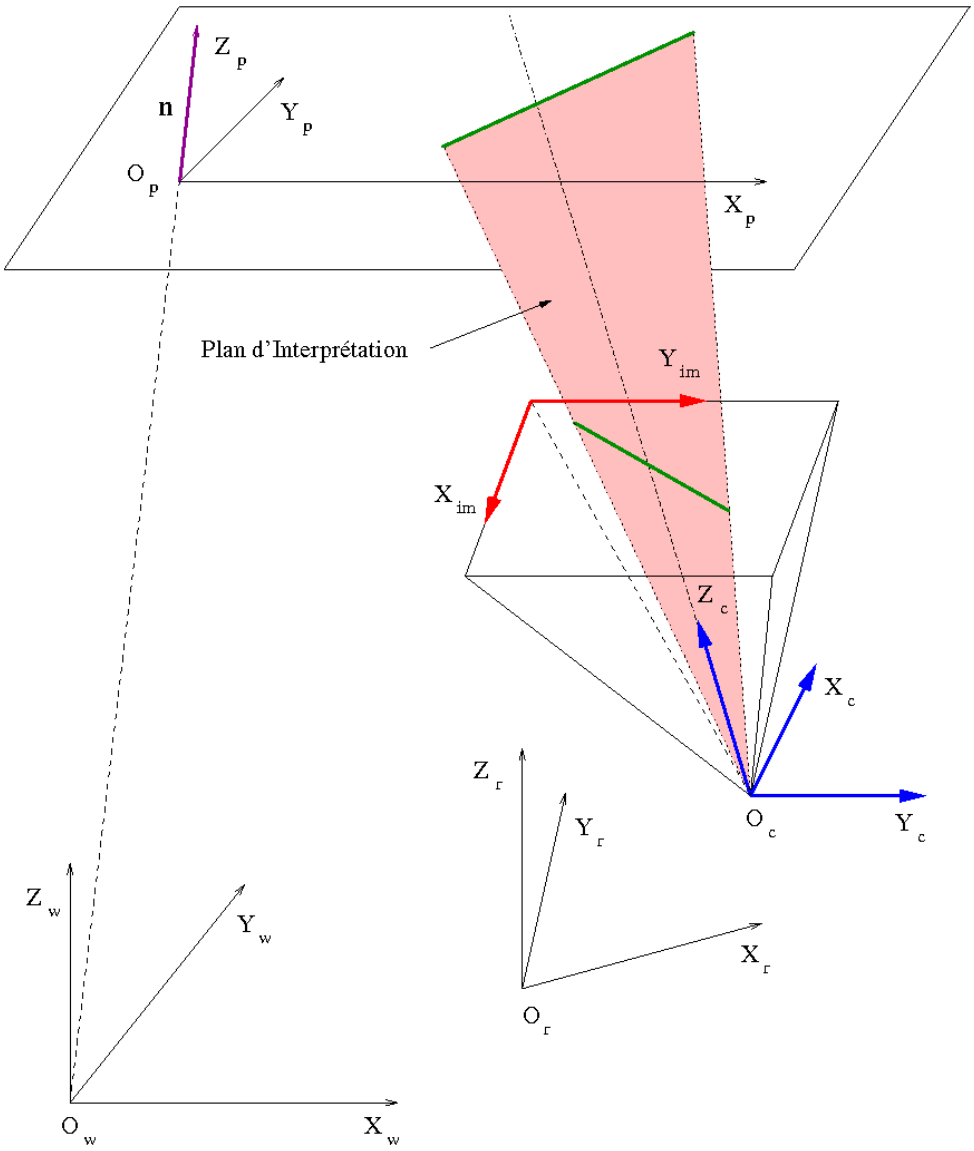


FIG. 4.13 – Segment droit 2D dans l'image et son correspondant amer ligne 2D attaché à l'amer plan

Ainsi, on peut représenter le plan d'interprétation qui passe par un segment 2D de l'image et par le centre optique de la caméra dans le repère local lié à l'amer plan extrait par la méthode de segmentation représentée dans la sections 4.6. La droite 3D formée par l'intersection entre le plan d'interprétation et le plan $z_p = 0$ peut être vu comme une droite 2D dans le plan $O_p X_p Y_p$. L'équation de la droite 2D qui est la projection du segment 2D dans l'image sur l'amer plan est donnée par :

$$n_{x,p}\, x_p + n_{y,p}\, y_p + d_p = 0 \tag{4.50}$$

La figure 4.14 illustre des segments 2D extraits dans l'image. La figure 4.15 donne un exemple

de construction des amers lignes 2D attachés aux amers plans extraits par l'algorithme de segmentation. Dans cet exemple, nous avons construit seulement un amer ligne 2D par plan porteur à titre illustratif.

4.8 Amer Plan texturé

Dans cette section, nous traitons le problème de placage de la texture sur les amers plans extraits par segmentation de l'image de profondeur. Le but est d'avoir des amers plans texturés. La texture sera utilisée pour extraire des points d'intérêt (points de Harris ou SIFT par exemple) qui seront utilisés en phase d'association de données (voir section 5.4).

4.8.1 Homographie

Définition : L'application mathématique reliant les différentes vues d'un motif plan par une ou plusieurs caméras projectives est une homographie [Faugeras *et al.*, 2001].

Une homographie bidimensionnelle **H** est la transformation reliant les projetés $(p, p^{'})$ sur deux plans images d'un même point 3D P situé sur un plan Π.

$$\lambda \begin{pmatrix} u_i^{'} \\ v_i^{'} \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix} \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} \tag{4.51}$$

d'où :

$$\begin{cases} u_i^{'} = \frac{h_{11}\, u_i + h_{12}\, v_i + h_{13}}{h_{31}\, u_i + h_{32}\, v_i + 1} \\ v_i^{'} = \frac{h_{21}\, u_i + h_{22}\, v_i + h_{23}}{h_{31}\, u_i + h_{32}\, v_i + 1} \end{cases} \tag{4.52}$$

Alors pour trouver une homographie 2D, il nous faut au moins quatre correspondances, et dans ce cas, pour trouver la matrice **H**, il suffit de résoudre le système linéaire de la forme $A\,X = B$ suivant :

FIG. 4.14 – Les segments lignes 2D dans l'image.

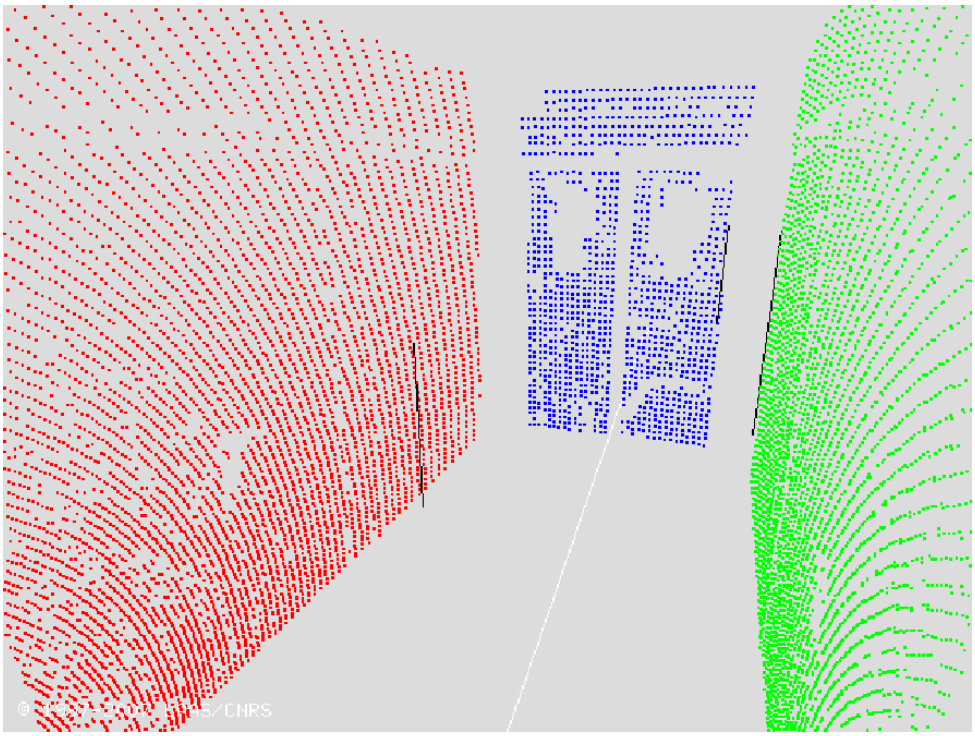


FIG. 4.15 – Amers plans avec quelques Amers lignes 2D attachés à eux.

$$\begin{bmatrix} \vdots & & & & & & & \\ u_i & v_i & 1 & 0 & 0 & 0 & -u_i\,u_i^{'} & -v_i\,u_i^{'} \\ 0 & 0 & 0 & u_i & v_i & 1 & -u_i\,v_i^{'} & -v_i\,v_i^{'} \\ \vdots & & & & & & & \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} \vdots \\ u_i^{'} \\ v_i^{'} \\ \vdots \end{bmatrix} \tag{4.53}$$

Dans le cas où nous avons plus de quatre correspondances, le système linéaire est surdéterminé : il peut être résolu par pseudo-inverse de la matrice A.

4.8.2 Procédure de placage de la texture sur le plan 3D

Afin d'ajouter des informations visuelles aux amers plans 3D trouvés par la segmentation de l'image de profondeur, nous suivons les étapes suivantes :

- Tout d'abord, et parce que les deux capteurs laser et caméra n'ont pas le même champ de vue, nous devons trouver pour chaque plan issu de la segmentation plane (cf. section 4.6) les points 3D qui sont aussi perçus par la caméra. Notons :

  - π_i un plan issu de la segmentation plane.
  - $\mathcal{P}_{\pi_i} = \{P_j \in \pi_i \mid j = 1 \dots N_{\pi_i}\}$ l'ensemble des points 3D laser appartenant au plan π_i.
  - $\mathcal{P}_{c,\pi_i} = \{P_j \in \mathcal{P}_{\pi_i} \ \wedge \ P_j$ est perçu par la caméra$\}$: l'ensemble des points du plan π_i qui sont aussi perçus par la caméra.

  Pour trouver l'ensemble $\mathcal{P}_{c,\pi_i}$, il suffit de projeter les points de l'ensemble $\mathcal{P}_{\pi_i}$ dans le repère image (en utilisant les matrices de calibrage trouvées en section 4.4), et de tester si la projection se trouve dans l'image.

- L'ensemble de points $\mathcal{P}_{c,\pi_i}$ sera projeté sur le plan π_i, pour trouver $\mathcal{P}^{\pi}_{c,\pi_i}$ un ensemble de points 2D dans le repère local lié au plan.

- Nous cherchons les points de contour convexe de l'ensemble $\mathcal{P}^{\pi}_{c,\pi_i}$. L'enveloppe convexe (*Convex Hull*) d'un ensemble de points est le plus petit ensemble convexe qui enveloppe tous les points. Dans le cas bidimensionnel, et pour un ensemble fini de points, l'enveloppe convexe est un polygone convexe. Plusieurs algorithmes ont été inventés pour résoudre ce problème. Nous avons au début choisi l'algorithme de parcours de Graham qui est un algorithme déterminant l'enveloppe convexe d'un ensemble de points. Son principal intérêt est sa complexité algorithmique en O(n log n). Cet algorithme doit son nom à Ronald Graham, qui a publié l'algorithme original en 1972 [Graham, 1972].
  Le problème du contour convexe est qu'il relie les régions non contiguës. Pour cette raison,

nous avons choisi d'extraire les contours de chaque région à part. Ensuite, nous utilisons tous les points des différents contours pour le calcul de l'homographie.
Soient $\mathcal{P}^{\pi}_{contour,\pi_i}$ l'ensemble des points des contours.

- Nous projetons les points de l'enveloppe convexe dans le repère image pour trouver l'ensemble $\mathcal{P}^{im}_{contour,\pi_i}$.

- Nous calculons l'homographie qui relie les deux ensembles de points $\mathcal{P}^{im}_{contour,\pi_i}$ et $\mathcal{P}^{\pi}_{contour,\pi_i}$, voir la section 4.8.1.

- Le calcul de l'homographie nous permet de placer la partie de l'image correspondant à la zone formée par la projection du contour sur le plan 3D. La figure 4.28 illustre un plan 3D sur lequel nous avons projeté la texture délivrée par la méthode présentée ci-dessus.

Les figures suivantes illustrent les différentes étapes du calcul de l'homographie et la création de la texture d'un plan. La figure 4.16 illustre une image des points 3D laser, sous forme de niveau de gris. Plus un point 3D est éloigné, plus son niveau de gris s'approche du blanc. La figure 4.17 montre l'image des points 3D après segmentation. Dans cette figure, on ne garde que les points qui appartiennent à un plan, et les points du même plan ont la même couleur (pour l'illustration). La figure 4.18 est basée sur l'image précédente, mais ne contient que les points 3D qui sont aperçus aussi par la caméra. Donc, ces points sont perçus par les deux capteurs, et on a pour chaque point les coordonnées cartésiennes issues du laser et l'information d'intensité provenant de la caméra.

Les figures 4.19 et 4.21 montre les points de deux facettes planes, ces points sont aperçus par les deux capteurs. Et les figures 4.20 et 4.22 illustrent les contours des régions dans chaque plan. Ainsi, nous constatons que le plan peut contenir plusieurs régions non contiguës.

Dans les figures 4.23 et 4.24, nous illustrons les textures pour chacune des facettes planes avec ou sans les points de contours projetés sur elles.

La figure 4.25 illustre l'image prise par la caméra, qui sera plaquée sur les plans extraits par l'algorithme de segmentation. Les plans texturés sont illustrés par la figure 4.26. Dans cette dernière image, nous constatons le problème de différence des champs de vue des deux capteurs : il existe des points appartenant au plan et perçus par le scanner laser, mais qui sont occultés par d'autres objets dans l'image. Par exemple, nous pouvons constater que les bords de la porte ouverte dans l'image 4.25 seront utilisés pour texturer le plan même si elles ne sont pas dans le même plan. Ce problème est provoqué à cause de deux raisons : La première est que les champs de vue des deux capteurs ne sont pas identiques. En effet, le capteur laser a un champ de vue de 180 degrés, alors que celui de la caméra n'est que d'environ 45 degrés. La deuxième cause est que les deux capteurs sont éloignés l'un de l'autre. En fait, vu la configuration du robot, nous utilisons la caméra gauche du banc stéréo situé sur le mât, et le scanner laser est proche du sol. En tenant compte des valeurs données par le calibrage (voir la section 4.4), la meilleure solution est d'avoir les deux capteurs très proches l'un de l'autre, comme par exemple, de fixer la caméra juste au dessus du scanner laser, une solution triviale, mais qui n'est pas facile à appliquer sur notre robot vu les contraintes matérielles.

FIG. 4.16 – Image 3D issue du Laser 3D

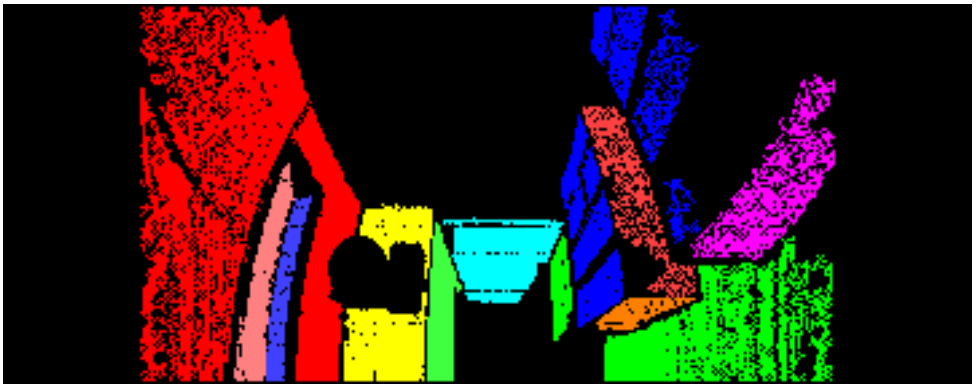


FIG. 4.17 – Image 3D après la segmentation plane : les points de chaque plan ont la même couleur

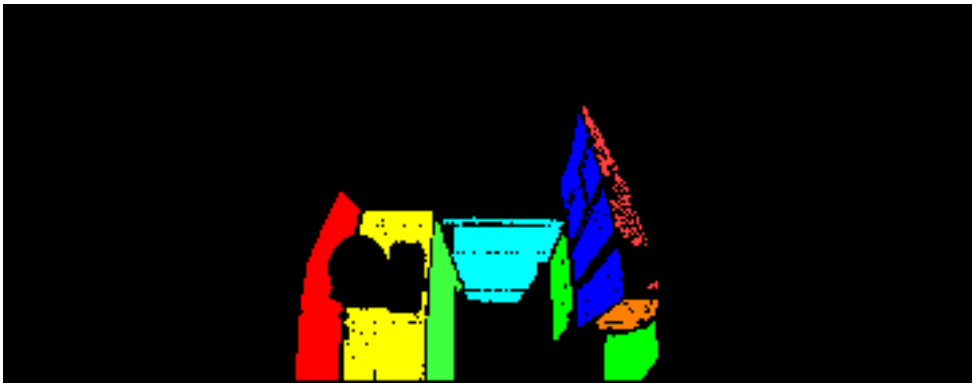


FIG. 4.18 – Image des points 3D vus aussi par la caméra après la segmentation plane : les points de chaque plan ont la même couleur

FIG. 4.19 – Image des points 3D d'un seul plan vus aussi par la caméra

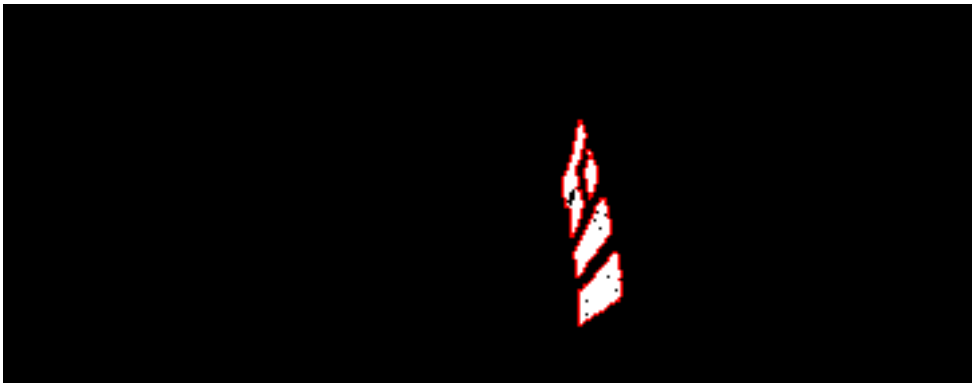


FIG. 4.20 – Image des points 3D d'un seul plan vus aussi par la caméra, et le contour (en rouge) de ces régions

4.9 Conclusion

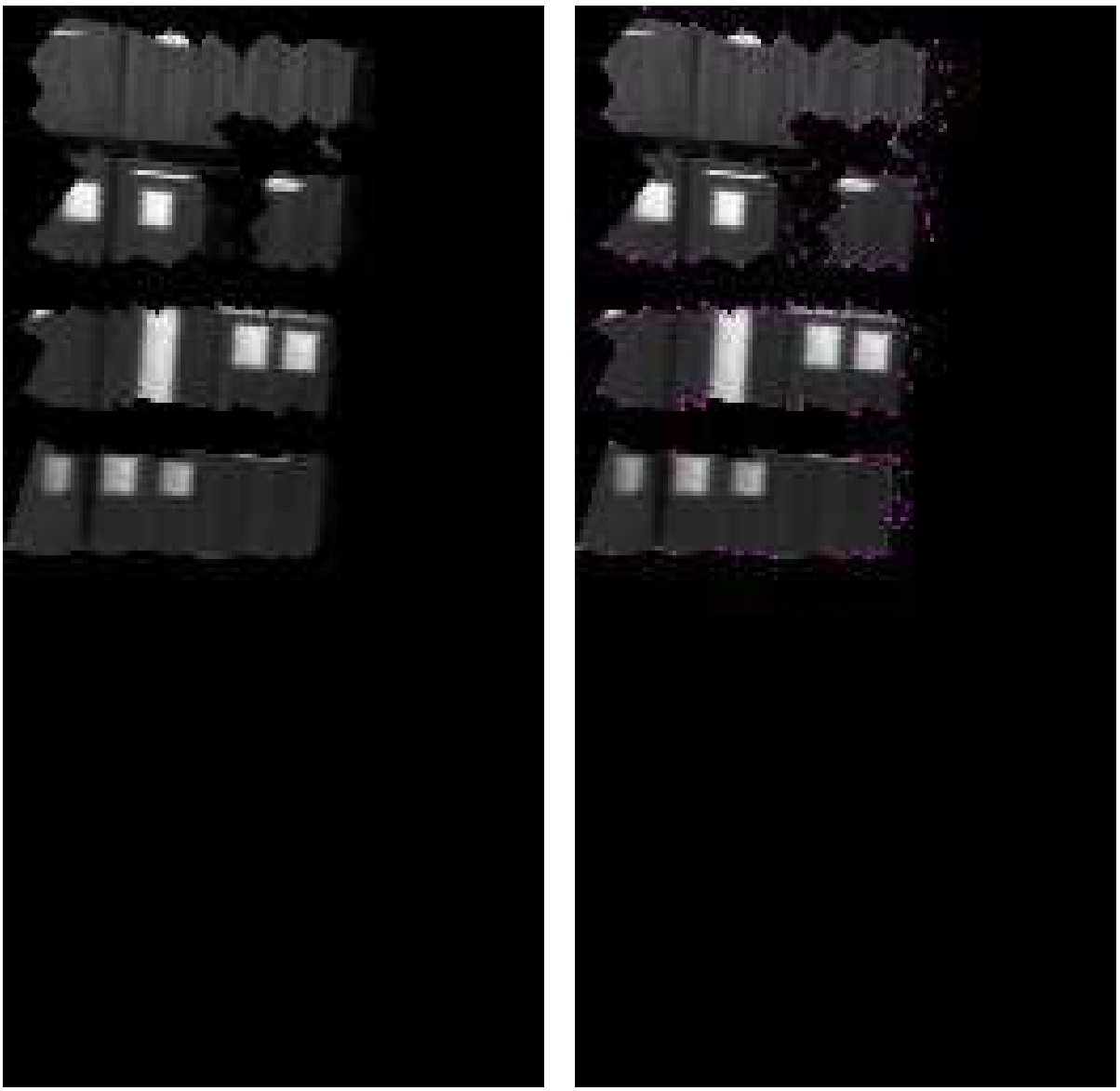
Nous avons présenté dans ce chapitre les matériels d'acquisition de notre robot démonstrateur. Il s'agit d'un scanner laser 3D et d'une caméra. Le scanner laser fournit une image de profondeur composée de dizaines de milliers de points 3D. Nous avons présenté un algorithme de segmentation d'une image de points 3D acquise par ce capteur, par une approche de type *region-growing*. Cette segmentation donne un ensemble de plans. Le calibrage des deux capteurs (laser et caméra) est réalisée par un toolbox Matlab utilisant une mire sous forme d'échiquier. Ce calibrage permet de trouver la transformation entre les deux repères liés aux capteurs, une étape essentielle avant de pouvoir fusionner les données des deux capteurs. Le calibrage permet d'exprimer entièrement les données photométriques et les données télémétriques dans un même référentiel. Les données issues de la caméra sont utilisées pour texturer les plans provenant de la segmentation de l'image de profondeur, ainsi que pour extraire les segments de droite 3D comme des amers lignes 2D attachés aux plans porteurs. Ce chapitre a traité seulement la partie *Perception* de l'algorithme SLAM ; la partie *Estimation* sera détaillée dans le prochain chapitre.

L'acquisition par le scanner laser pivotant prend un temps considérable, et empêche de travailler en ligne. Le robot doit s'arrêter avant chaque acquisition d'une image de profondeur. Nous

FIG. 4.21 – Image des points 3D d'un seul plan vus aussi par la caméra

FIG. 4.22 – Image des points 3D d'un seul plan vus aussi par la caméra, et le contour (en rouge) de ces régions

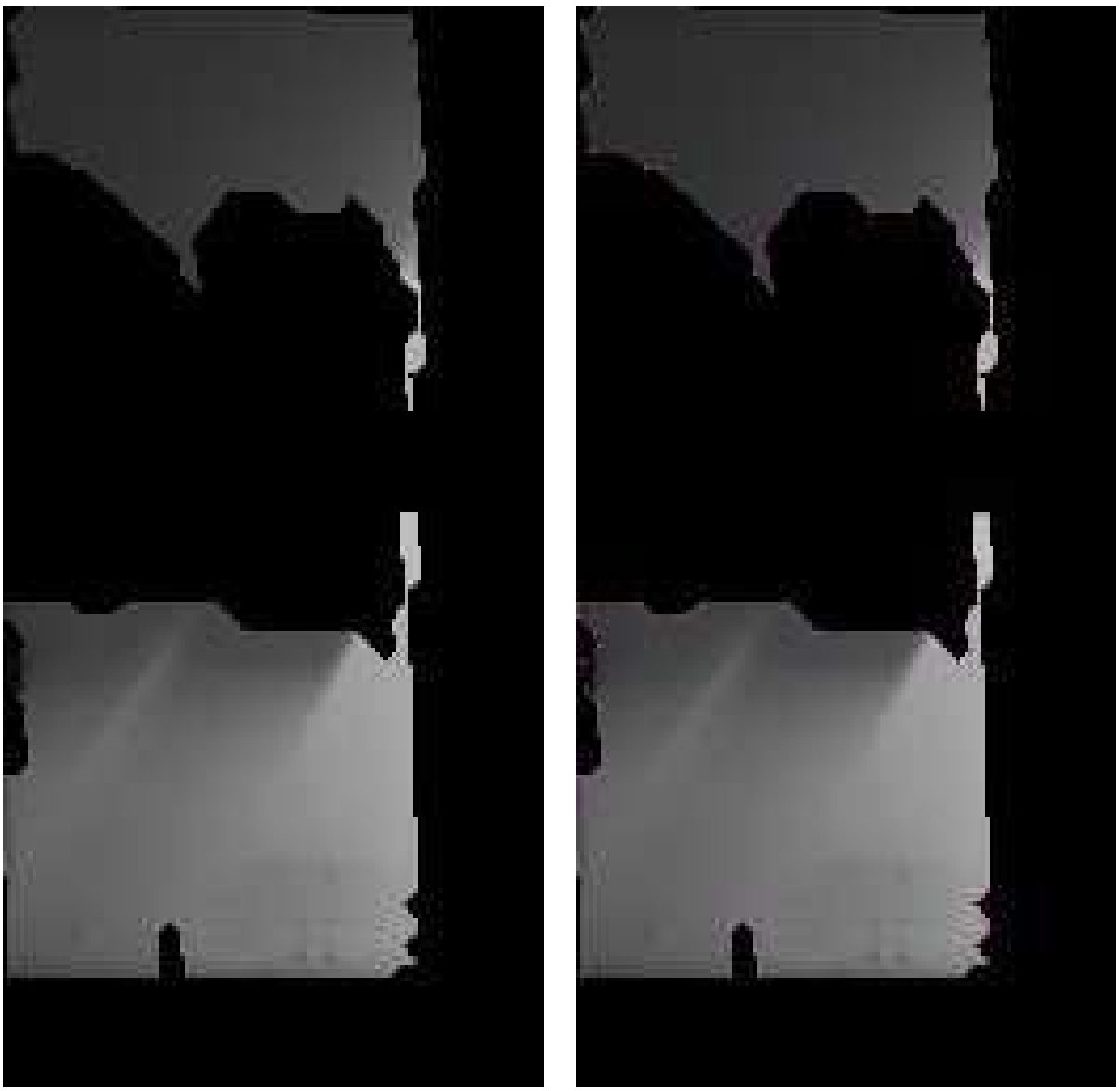
envisageons de remplacer le télémètre laser SICK pivotant, par une caméra optique à temps de vol de type Swiss Ranger SR3000, figure 4.29, déjà montée sur notre robot Jido. Ce capteur est une "caméra de profondeur" : il délivre des mesures d'images de profondeur en temps réel. De plus, il est conçu pour fonctionner dans les conditions normales d'éclairage de l'environnement intérieur. Il fournit des images de taille 176x144 pixels, et pour chaque pixel, les coordonnées (x, y, z) et l'intensité I sont fournies. Mais, il a un point faible, c'est sa porté $(7.5m)$, et la précision qui se dégrade très rapidement avec la profondeur. Malgré tout, l'exploitation de ce nouveau capteur devrait permettre d'acquérir les données sensorielles en mouvement, et d'accélérer le temps nécessaire pour l'exploration.



(a) Image de la texture calculée par l'homographie

(b) Image de la texture calculée par l'homographie, avec les points des contours

FIG. 4.23 – Image de la texture calculée par l'homographie pour une facette plane



(a) Image de la texture calculée par l'homographie

(b) Image de la texture calculée par l'homographie, avec les points des contours

FIG. 4.24 – Image de la texture calculée par l'homographie pour une facette plane

FIG. 4.25 – Placage de la texture : l'image d'origine

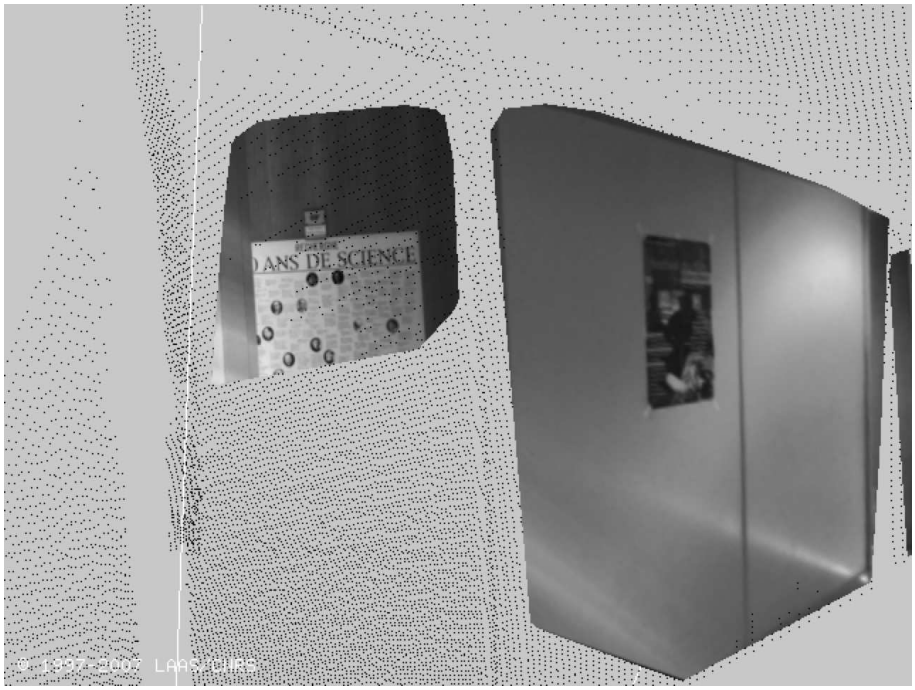


FIG. 4.26 – Les plans texturés

FIG. 4.27 – Placage de la texture : l'image d'origine

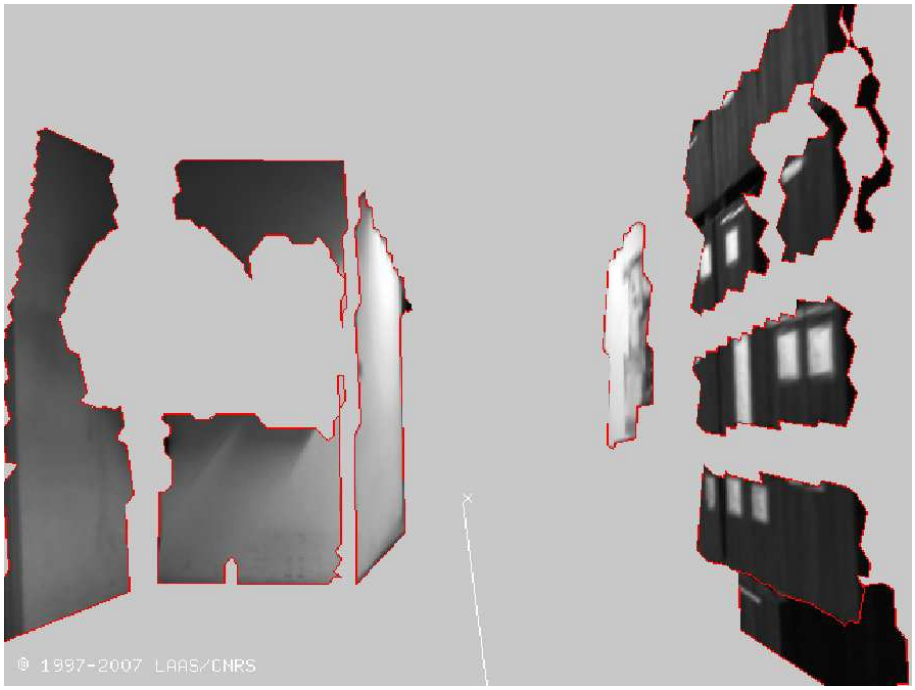


FIG. 4.28 – Les plans texturés : un plan texturé est composé de plusieurs régions, comme par exemple le plan passant par les façades des boîtes aux lettres

FIG. 4.29 – Le SwissRanger SR-3000

Chapitre 5

Construction de la Carte Hétérogène

Sommaire

5.1 Introduction

Rappelons que la Cartographie et la Localisation Simultanées (SLAM) est le processus qui permet à un robot ou un véhicule mobile, de construire une carte de l'environnement et d'utiliser cette carte pour estimer sa propre position. Lorsqu'il exécute la fonction SLAM, un véhicule réalise un processus complexe, combinant l'exécution de déplacements dans l'espace libre, la construction incrémentale d'une carte de l'environnement et l'exploitation de cette carte afin de détecter l'espace libre, puis générer le prochain déplacement dans cet espace libre vers la meilleure position pour compléter la carte (problématique du choix de la *Next Best View*, non abordée dans nos travaux).

Essentiellement, la robustesse de cette fonction dépend des capacités du véhicule à extraire, à partir des données sensorielles, des informations utiles pour la navigation (appelées **amers** ou *Landmarks*). Le véhicule démarre d'une position inconnue sans aucune connaissance *a priori* des emplacements des amers. En utilisant les mesures relatives des amers, le véhicule calcule des estimations de sa position et des positions des amers. En se déplaçant, le véhicule construit une carte d'amers et l'utilise pour fournir une estimation continue de sa position. La localisation du véhicule est faite par rapport à un repère absolu, défini généralement par sa position initiale au début de l'exploration. En suivant tout au long de la séquence des données sensorielles, la position relative entre le véhicule et les amers identifiables de l'environnement, la position du véhicule et les positions des amers peuvent être estimées simultanément.

Nous avons vu en section 2.3 que l'algorithme de SLAM est très bien étudié depuis les travaux fondateurs de Chatila, Durrant-White, Ayache, Smith et Cheesemann à partir de 1985 : une grande partie des travaux sur le SLAM ont eu pour but de construire des cartes 2D à partir de coupes-laser. Utiliser de nouveaux capteurs 3D, exploiter des données visuelles sur l'apparence afin d'extraire et apparier de manière robuste des *primitives*, restent des sujets ouverts. Fusionner les données de plusieurs capteurs est une approche intéressante pour surmonter les insuffisances de chaque capteur et pour obtenir des résultats plus précis et sophistiqués. Nous construisons une carte métrique hétérogène multi-primitives.

Dans ce chapitre, nous allons d'abord présenter l'algorithme SLAM de type *feature-based*, utilisé pour produire des estimations de la position du robot et des amers en utilisant les mesures relatives à la position du robot. Nous utilisons le formalisme du filtre de Kalman étendu (EKF-SLAM). Nous commençons par une présentation de l'état du véhicule et de l'environnement. La section suivante présente les modèles du véhicule et des amers utilisés pour décrire le modèle dynamique du robot et le modèle d'observation des amers. Ensuite, nous présentons

un cadre général pour le SLAM basé sur le filtre de Kalman étendu. Dans cet algorithme, nous distinguons trois étapes : prédiction, observation et mise à jour (ou *update*). Les outils mathématiques nécessaires pour accomplir chaque étape sont présentés. En section 5.3, nous donnons les équations nécessaires pour ajouter les amers de type lignes 2D attachés aux amers plans. Et dans la section suivante, le problème d'association de données est détaillé. Finalement, les différentes étapes pour la construction de la carte hétérogène sont regroupées en section 5.5. Enfin, nous montrons les implémentations et les résultats expérimentaux.

5.2 Construction d'une Carte Stochastique à base de Plans

La figure 5.1 illustre les différentes étapes de l'algorithme de SLAM. Dans le chapitre précédent, nous avons présenté l'étape de la perception et de l'extraction des amers à partir des données capteurs (laser et caméra). Dans ce chapitre, nous traitons les techniques adoptées pour créer et mettre à jour la représentation *carte stochastique*, ainsi que l'appariement entre les primitives perçues dans la scène courante et celles déjà inclues dans la carte (association de données).

L'algorithme SLAM construit et maintient une représentation de l'état de l'environnement et de l'état du robot qui s'y déplace, comme illustré dans la figure 5.2. Pendant ses mouvements, le robot utilise ses capteurs pour observer les amers dans son entourage. L'état du système à l'instant k, $\mathbf{X}(k)$, est composé du vecteur $\mathbf{X}_v$ représentant l'état du robot, et de n_f vecteurs décrivant les amers observés, $\mathbf{X}_i(k)$, $i = 1, \ldots, n_f$.

$$\mathbf{X}(k) = \begin{bmatrix} \mathbf{X}_v^W \\ \mathbf{X}_1^W \\ \vdots \\ \mathbf{X}_{n_f}^W \end{bmatrix} \tag{5.1}$$

où $\mathbf{X}_i^W$ est l'état d'un amer par rapport au repère global $\mathcal{R}_W$. Dans la suite (sauf indication contraire), tous les états sont exprimés par rapport au repère global, donc la référence du repère global sera omise.

De plus, le vecteur d'état du système peut être écrit en groupant tous les états des amers sous un seul terme $\mathbf{X}_m(k)$:

$$\mathbf{X}(k) = \begin{bmatrix} \mathbf{X}_v^W \\ \mathbf{X}_m^W \end{bmatrix} \tag{5.2}$$

Notre robot JIDO se déplace dans un environnement d'intérieur supposé inconnu, composé (d'une façon simplifiée) par des surfaces planes que nous allons prendre comme amers dans

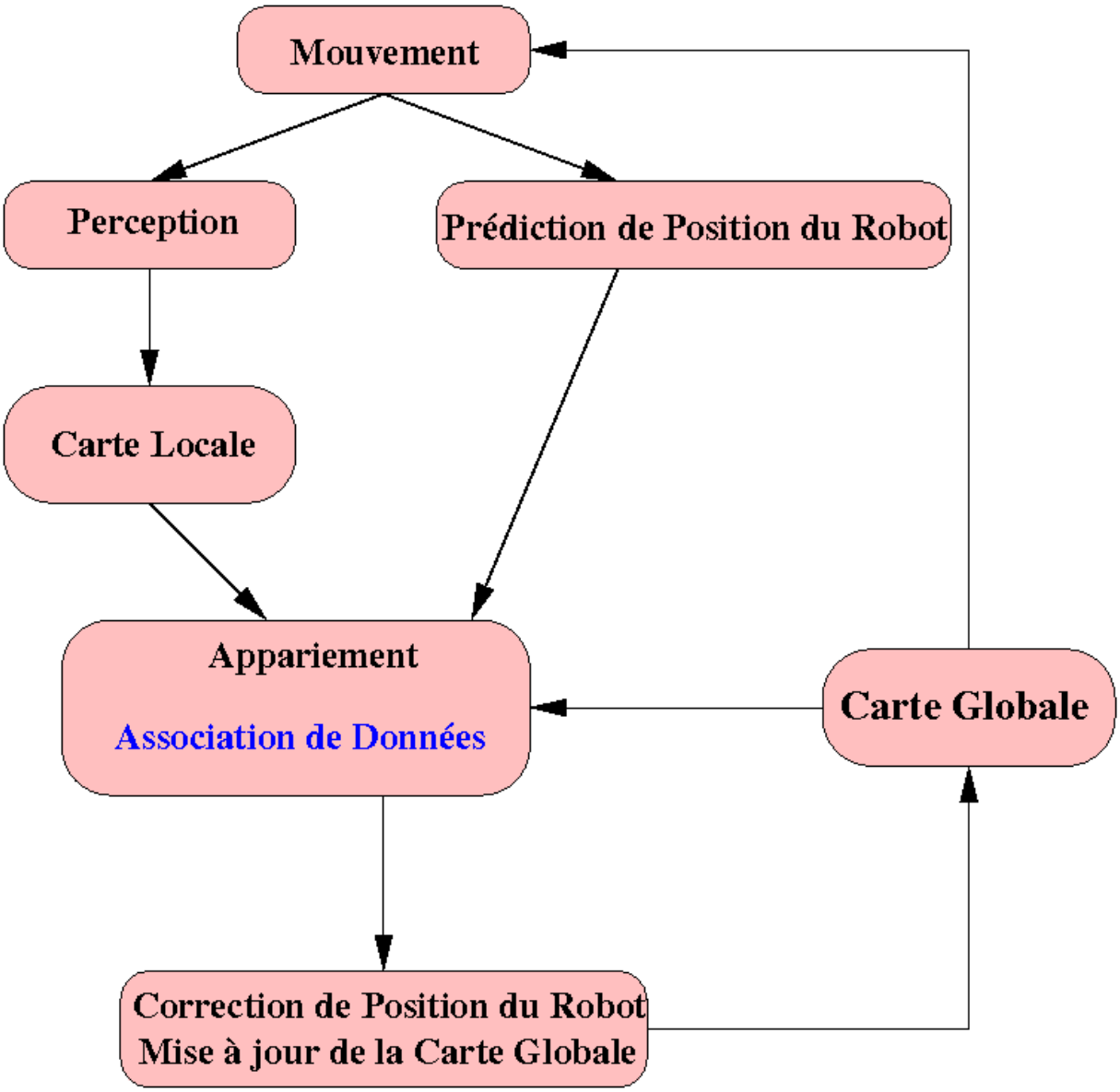


FIG. 5.1 – L'algorithme de SLAM

l'algorithme de SLAM. Considérons le scénario présenté dans la figure 5.2. L'état du robot à l'instant k peut être déterminé par sa position et son orientation dans l'espace. Le vecteur d'état du robot est défini par :

$$\mathbf{X}_v(k) = \begin{bmatrix} x_v(k) \\ y_v(k) \\ \theta_v(k) \end{bmatrix} \tag{5.3}$$

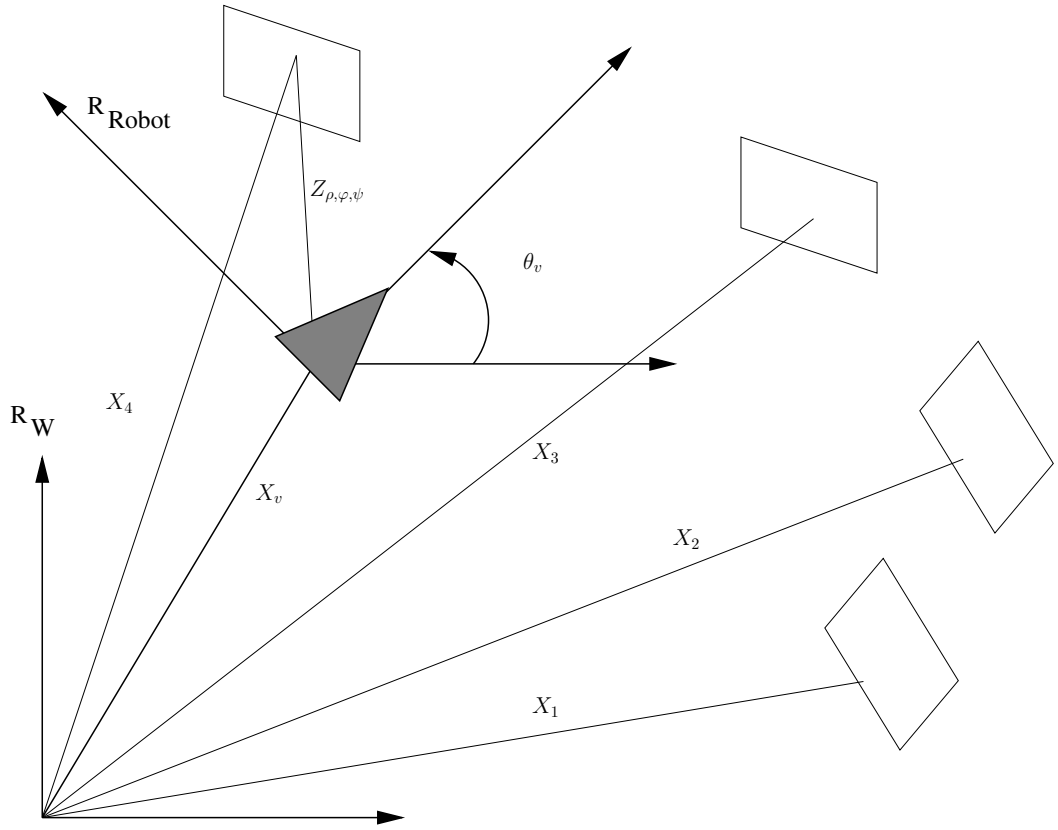


FIG. 5.2 – L'état du système

Chaque surface plane (mur, plafond, sol, etc.) est considérée comme un plan infini défini par trois paramètres, voir la section 4.6.2 pour le choix de ces paramètres :

$$\mathbf{X}_i(k) = \begin{bmatrix} \rho_i(k) \\ \varphi_i(k) \\ \psi_i(k) \end{bmatrix} \tag{5.4}$$

et le vecteur d'état des amers dans la carte est la concaténation des vecteurs décrivant chacun d'eux :

$$\mathbf{X}_m(k) = \begin{bmatrix} \mathbf{X}_1(k) \\ \vdots \\ \mathbf{X}_{n_f}(k) \end{bmatrix} \tag{5.5}$$

5.2.1 Le modèle du Véhicule

Un modèle du véhicule essaie de traduire la relation entre l'état précédent du véhicule, $\mathbf{X}_v(k-1)$, et son état courant, $\mathbf{X}_v(k)$, étant donné le vecteur de contrôle $u(k)$:

$$\mathbf{X}_v(k) = \mathbf{f}\Big(\mathbf{X}_v(k-1), u(k)\Big) + \nu_v(k) \tag{5.6}$$

où $\mathbf{X}_v(k) \in \mathcal{R}^{n_v}$ est le vecteur d'état du véhicule à l'instant k et $\nu_v(k)$ est un vecteur aléatoire décrivant le bruit dans le modèle dynamique du véhicule. $f(.,.)$ est le modèle (ou fonction) du mouvement du robot.

Un modèle précis du véhicule est essentiel dans tous les systèmes de navigation. Ce modèle peut être plus au moins complexe. Selon la fiabilité obtenue pour estimer les déplacements élémentaires du véhicule à partir des capteurs proprioceptifs disponibles (odométrie, gyromètre, compas, inclinomètre en terrain non plat...), des modèles plus ou moins précis peuvent être générés. Un modèle *Idéal* du véhicule pourrait prévoir le mouvement sans erreur et pour toute transition de l'état. C'est évidement impossible, parce que les modèles choisis sont toujours imparfaits et sujets à des bruits qui endommagent la précision de l'état estimé.

5.2.2 Le modèle des Amers

Dans le contexte SLAM, un amer (*landmark*) est une *entité* (primitive perceptuelle : point, segment, plan, etc., ou un objet) de l'environnement qui peut être observée de manière fiable par les capteurs du véhicule. Les amers doivent être représentés par un modèle paramétrique pour pouvoir être inclus dans le modèle d'état du système. Points, coins, lignes et polylignes sont des amers rapportés dans la littérature [Castellanos *et al.*, 1999; Leonard et Durrant-Whyte, 1991]. Pour l'algorithme SLAM, les amers sont supposés fixes ; de ce fait, la partie dynamique du modèle du système considère seulement le modèle du véhicule. Donc le modèle simple des amers est :

$$\hat{\mathbf{X}}_m(k) = \hat{\mathbf{X}}_m(k-1) \tag{5.7}$$

5.2.3 Le modèle de la mesure

Le processus d'observation de l'état (mesure) peut être modélisé dans l'espace d'état par une fonction vectorielle non linéaire de la forme :

$$Z(t) = \mathbf{h}\Big(\mathbf{X}(t), t\Big) + w(t) \tag{5.8}$$

où $Z(t) \in \mathcal{R}^m$ est l'observation (mesure) à l'instant t, $h(.,.,.)$ est le modèle (ou fonction)

d'observation, et $w(t)$ est un vecteur aléatoire qui représente le bruit de mesure. Dans le cas discret, le modèle devient :

$$Z(k) = \mathbf{h}\Big(\mathbf{X}(k)\Big) + w(k) \tag{5.9}$$

5.2.4 Prédiction

L'étape de prédiction du filtre de Kalman étendu utilise le modèle de mouvement du robot pour produire une estimation de la position du robot $\hat{\mathbf{X}}_v(k|k-1)$ à l'instant k sachant les informations disponibles jusqu'à l'instant $k-1$ comme :

$$\hat{\mathbf{X}}_v(k|k-1) = \mathbf{f}\Big(\hat{\mathbf{X}}_v(k-1|k-1), u(k)\Big) \tag{5.10}$$

Le modèle de l'amer de l'équation 5.7 donne la prédiction des amers :

$$\hat{\mathbf{X}}_m(k|k-1) = \hat{\mathbf{X}}_m(k-1|k-1) \tag{5.11}$$

L'étape de prédiction du filtre peut s'écrire :

$$\begin{bmatrix} \hat{\mathbf{X}}_v(k|k-1) \\ \\ \hat{\mathbf{X}}_m(k|k-1) \end{bmatrix} = \begin{bmatrix} \mathbf{f}\Big(\hat{\mathbf{X}}_v(k-1|k-1), u(k)\Big) \\ \\ \hat{\mathbf{X}}_m(k-1|k-1) \end{bmatrix} \tag{5.12}$$

L'erreur sur l'estimation du déplacement du robot u, doit se propager sur l'état, afin de mettre à jour la matrice de covariance sur l'état à travers le modèle de son mouvement. Le filtre de Kalman étendu linéarise la propagation de l'incertitude autour de l'estimation courante $\hat{\mathbf{X}}(k-1|k-1)$ en utilisant le Jacobien $\nabla_X \mathbf{f}(k)$ de $\mathbf{f}$ évalué en $\hat{\mathbf{X}}(k-1|k-1)$.

$$\mathbf{P}(k|k-1) = \nabla_X \mathbf{f}(k)\ \mathbf{P}(k-1|k-1)\ \nabla_X \mathbf{f}^T(k)\ + \mathbf{Q}(k) \tag{5.13}$$

L'incertitude dans l'entrée de contrôle, $u(k)$, peut aussi être prise en compte, en utilisant l'incertitude du contrôle $U(k)$ et le Jacobien $\nabla_u \mathbf{f}(k)$ de $\mathbf{f}$ évalué autour de l'entrée de contrôle courante :

$$\begin{aligned} \mathbf{P}(k|k-1) = \quad & \nabla_X \mathbf{f}(k)\ \mathbf{P}(k-1|k-1)\ \nabla_X \mathbf{f}^T(k) + \\ & + \nabla_u \mathbf{f}(k)\ U(k)\ \nabla_u \mathbf{f}^T(k)\ + \mathbf{Q}(k) \end{aligned} \tag{5.14}$$

Pour l'algorithme SLAM, cette étape de filtrage peut être simplifiée grâce à l'hypothèse que les amers sont fixes. Ceci permet de réduire la complexité du calcul de la matrice de covariance sur l'état, au calcul seulement de la matrice de variance associée à l'état du robot et aux matrices de covariances entre le robot et chaque amer [Williams, 2001].

$$\begin{bmatrix} \mathbf{P}_{vv}(k|k-1) & \mathbf{P}_{vm}(k|k-1) \\ \mathbf{P}_{vm}^T(k|k-1) & \mathbf{P}_{mm}(k|k-1) \end{bmatrix} = \begin{bmatrix} \nabla_v \mathbf{f}(k)\, \mathbf{P}_{vv}(k-1|k-1)\, \nabla_v \mathbf{f}^T(k) + \mathbf{Q}(k) & \nabla_v \mathbf{f}(k)\, \mathbf{P}_{vm}(k-1|k-1) \\ (\nabla_v \mathbf{f}(k)\, \mathbf{P}_{vm}(k-1|k-1))^T & \mathbf{P}_{mm}(k-1|k-1) \end{bmatrix} \tag{5.15}$$

5.2.5 Observation

La fusion de l'observation avec l'état estimé est accomplie d'abord par le calcul de la prédiction de l'observation, $\hat{Z}(k|k-1)$, utilisant le modèle de l'observation $\mathbf{h}$:

$$\hat{Z}(k|k-1) = \mathbf{h}\Big(\hat{\mathbf{X}}(k|k-1)\Big) \tag{5.16}$$

Lorsque les observations sont reçues depuis les capteurs du robot, elles doivent être associées à des amers déjà perçus, donc déjà intégrés dans la carte de l'environnement : c'est la phase dite de *Data association*. La différence entre les observations actuelles, $Z(k)$, et les observation prédites $\hat{Z}(k|k-1)$ est connue comme l'innovation ν,

$$\nu = Z(k) - \hat{Z}(k|k-1) \tag{5.17}$$

La matrice de covariance de l'innovation, $\mathbf{S}(k)$, est calculée en utilisant l'estimation courante, la covariance de l'état, $\mathbf{P}(k|k-1)$, le Jacobien du modèle de l'observation, $\nabla_X \mathbf{h}(k)$, et la matrice de covariance de l'observation $\mathbf{R}(k)$.

$$\mathbf{S}(k) = \nabla_X \mathbf{h}(k)\, \mathbf{P}(k|k-1)\, \nabla_X \mathbf{h}^T(k) + \mathbf{R}(k) \tag{5.18}$$

L'innovation et sa matrice de covariance peuvent être utilisées pour valider les mesures avant d'être intégrées dans le calcul des états estimés [Bar-Shalom *et al.*, 2001]. Le calcul de la matrice de covariance de l'innovation peut aussi être simplifié en notant que chaque observation est seulement dépendante de l'amer observé. Le Jacobien de la fonction d'observation de l'amer i, $\nabla_X \mathbf{h}(k)$, est ainsi une matrice éparse de la forme :

$$\nabla_X \mathbf{h}(k) = [\ \nabla_v \mathbf{h}(k) \quad 0 \quad \dots \quad \nabla_i \mathbf{h}(k) \quad 0 \quad \dots\ ] \tag{5.19}$$

En développant le produit dans l'équation 5.18, et en utilisant la dernière forme du Jacobien, on obtient la matrice de covariance de l'innovation provenant de l'observation de l'amer i :

$$\begin{array}{rcl} \mathbf{S}(k) & = & \nabla_v \mathbf{h}(k)\, \mathbf{P}_{vv}(k|k-1)\, \nabla_v \mathbf{h}^T(k) + \\ & & \nabla_v \mathbf{h}(k)\, \mathbf{P}_{vi}(k|k-1)\, \nabla_i \mathbf{h}^T(k) + \\ & & \nabla_i \mathbf{h}(k)\, \mathbf{P}_{vi}^T(k|k-1)\, \nabla_v \mathbf{h}^T(k) + \\ & & \nabla_i \mathbf{h}(k)\, \mathbf{P}_{ii}(k|k-1)\, \nabla_i \mathbf{h}^T(k) + \mathbf{R}(k) \end{array} \tag{5.20}$$

Dans le cas de la fusion plan/plan : après une étape de segmentation des données 3D issues de laser 3D, on obtient des plans 3D dans le repère SICK lié au capteur. Soient $(\rho_s, \varphi_s, \psi_s)$ les paramètres d'un plan dans le repère SICK, nous avons :

$$R_{r,sk} = I_3 \tag{5.21}$$

$$t_{r,sk} = \begin{bmatrix} x_{sk} \\ \\ y_{sk} \\ \\ z_{sk} \end{bmatrix} \tag{5.22}$$

donc, le robot perçoit un amer plan par la fonction d'observation suivante exprimée dans le repère robot $\mathcal{R}_r$:

$$\begin{cases} \rho_r = & \rho_s + \cos\varphi_s \sin\psi_s \;\; x_{sk} \\ & + \sin\varphi_s \sin\psi_s \;\; y_{sk} \\ & + \cos\psi_s \;\; z_{sk} \\ \\ \varphi_r = & \varphi_s \\ \\ \psi_r = & \psi_s \end{cases} \tag{5.23}$$

Relation entre les paramètres du plan dans les repères Monde et Robot

Cherchons la relation entre les équations du plan dans les deux repères global et robot. Soit $(\rho_w, \varphi_w, \psi_w)$ et $(\rho_r, \varphi_r, \psi_r)$ les paramètres du plan dans le repère global et robot respectivement. Dans la figure 5.3, soit O_p et Q la projection de O_w et O_r respectivement sur le plan. On a

$$\overrightarrow{O_w O_p} = \rho_w \overrightarrow{n} \tag{5.24}$$

$$\overrightarrow{O_r Q} = \rho_r \overrightarrow{n} \tag{5.25}$$

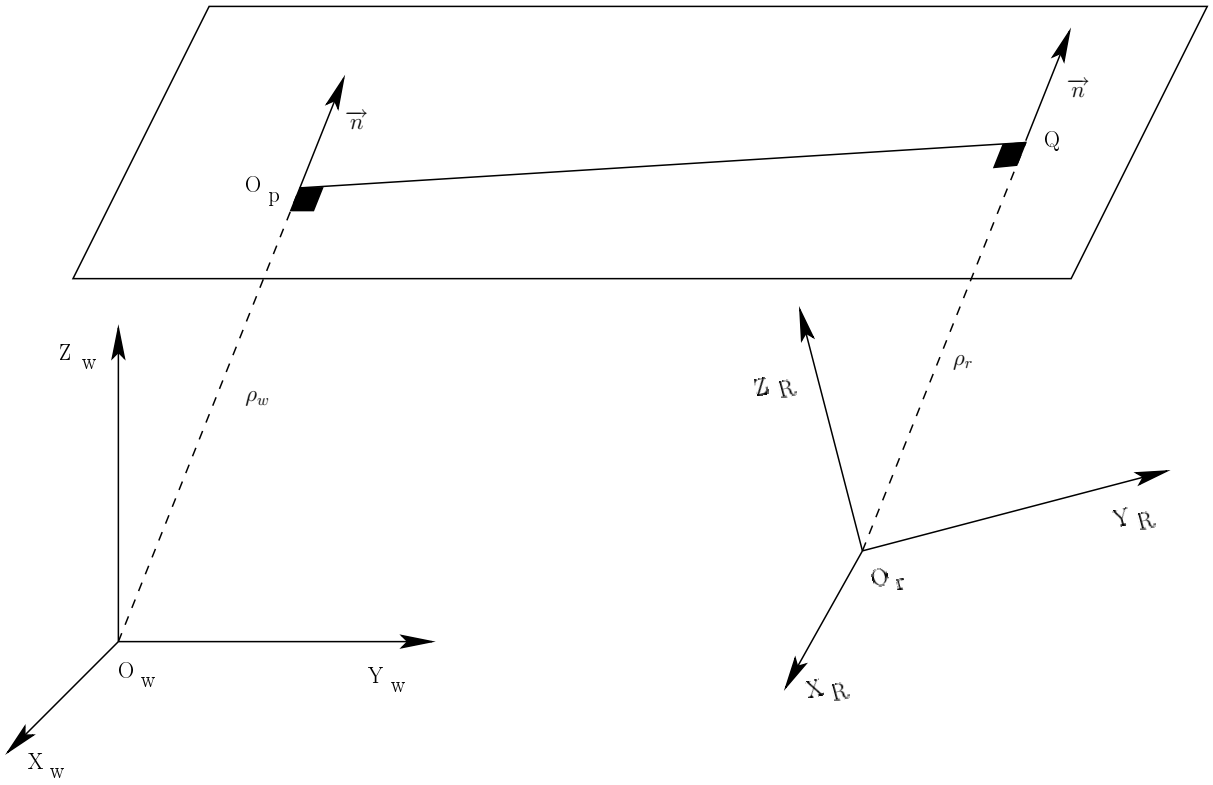


FIG. 5.3 – Un plan et les repères du monde (W) et du robot (R)

$$\overrightarrow{O_wQ} = \overrightarrow{O_wO_r} + \overrightarrow{O_rQ} = \overrightarrow{O_wO_p} + \overrightarrow{O_pQ} \tag{5.26}$$

$$\overrightarrow{O_wO_r} + \rho_r \overrightarrow{n} = \rho_w \overrightarrow{n} + \overrightarrow{O_pQ} \tag{5.27}$$

Si on multiplie les deux termes par $\overrightarrow{n}$, et en profitant du fait que $\overrightarrow{n}$ est perpendiculaire à $\overrightarrow{O_pQ}$, on aura :

$$\rho_r = \rho_w - \overrightarrow{O_wO_r}.\overrightarrow{n} \tag{5.28}$$

Or $\overrightarrow{O_wO_r}$ est le déplacement du robot par rapport au repère global. Si le robot se déplace sur un plan horizontal, on a :

$$\rho_r = \rho_w - (\cos\varphi_w \sin\psi_w \; x_v + \sin\varphi_w \sin\psi_w \; y_v) \tag{5.29}$$

De même, la relation entre la normale exprimée dans les deux repères nous donne :

$$\overrightarrow{n}_r = R_{r,w}\,\overrightarrow{n}_w = R_{w,r}^T\,\overrightarrow{n}_w \tag{5.30}$$

$$\begin{bmatrix} \varphi_r \\ \\ \psi_r \end{bmatrix} = \begin{bmatrix} \varphi_w - \theta_v \\ \\ \psi_w \end{bmatrix} \tag{5.31}$$

Donc pour un robot se déplaçant sur un plan horizontal, la relation entre les paramètres du plan dans les repères global et robot :

$$\begin{cases} \rho_r &= \rho_w - \cos\varphi_w \sin\psi_w\ x_v + \\ & \quad - \sin\varphi_w \sin\psi_w\ y_v \\ \varphi_r &= \varphi_w - \theta_v \\ \psi_r &= \psi_w \end{cases} \tag{5.32}$$

ou inversement

$$\begin{cases} \rho_w &= \rho_r + \cos(\varphi_r + \theta_v) \sin\psi_r\ x_v \\ & \quad + \sin(\varphi_r + \theta_v) \sin\psi_r\ y_v \\ \varphi_w &= \varphi_r + \theta_v \\ \psi_w &= \psi_r \end{cases} \tag{5.33}$$

calculons les Jacobiennes :

$$\nabla_v \mathbf{h} = \frac{\partial \mathbf{h}}{\partial X_v} = \begin{bmatrix} \frac{\partial \rho_r}{\partial x_v} & \frac{\partial \rho_r}{\partial y_v} & \frac{\partial \rho_r}{\partial \theta_v} \\ \\ \frac{\partial \varphi_r}{\partial x_v} & \frac{\partial \varphi_r}{\partial y_v} & \frac{\partial \varphi_r}{\partial \theta_v} \\ \\ \frac{\partial \psi_r}{\partial x_v} & \frac{\partial \psi_r}{\partial y_v} & \frac{\partial \psi_r}{\partial \theta_v} \end{bmatrix} \tag{5.34}$$

$$\nabla_v \mathbf{h} = \begin{bmatrix} -\cos\varphi_w \sin\psi_w & \sin\varphi_w \sin\psi_w & 0 \\ \\ 0 & 0 & -1 \\ \\ 0 & 0 & 0 \end{bmatrix} \tag{5.35}$$

$$\nabla_i \mathbf{h} = \frac{\partial \mathbf{h}}{\partial X_i} = \begin{bmatrix} \frac{\partial \rho_r}{\partial \rho_w} & \frac{\partial \rho_r}{\partial \varphi_w} & \frac{\partial \rho_r}{\partial \psi_w} \\ \frac{\partial \varphi_r}{\partial \rho_w} & \frac{\partial \varphi_r}{\partial \varphi_w} & \frac{\partial \varphi_r}{\partial \psi_w} \\ \frac{\partial \psi_r}{\partial \rho_w} & \frac{\partial \psi_r}{\partial \varphi_w} & \frac{\partial \psi_r}{\partial \psi_w} \end{bmatrix} \tag{5.36}$$

$$\nabla_i \mathbf{h} = \begin{bmatrix} 1 & \alpha_1 & \alpha_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.37}$$

où :
$\alpha_1 = \sin\varphi_w \sin\psi_w x_v - \cos\varphi_w \sin\psi_w y_v$
$\alpha_2 = -\cos\varphi_w \cos\psi_w x_v - \sin\varphi_w \cos\psi_w y_v$

Calculons la prédiction de l'observation d'un plan relativement à la position du robot :

$$\hat{Z}(k|k-1) = \begin{bmatrix} \hat{\rho}_r(k|k-1) \\ \hat{\varphi}_r(k|k-1) \\ \hat{\psi}_r(k|k-1) \end{bmatrix} \tag{5.38}$$

Donc

$$\begin{cases} \hat{\rho}_r(k|k-1) = & \hat{\rho}_w(k|k-1) \\ & -\cos(\hat{\varphi}_w(k|k-1))\sin(\hat{\psi}_w(k|k-1))\hat{x}_v(k|k-1) \\ & -\sin(\hat{\varphi}_w(k|k-1))\sin(\hat{\psi}_w(k|k-1))\hat{y}_v(k|k-1) \\ \hat{\varphi}_r(k|k-1) = & \hat{\varphi}_w(k|k-1) - \hat{\theta}_v(k|k-1) \\ \hat{\psi}_r(k|k-1) = & \hat{\psi}_w(k|k-1) \end{cases} \tag{5.39}$$

5.2.6 Mise à jour

L'association de données est le processus de rechercher les appariements entre les amers perçus dans la scène courante et les amers déjà enregistrés dans la carte. Nous détaillons en section 5.4 comment réaliser cette association. Dans cette section, nous considérons que l'association de données est faite.

Une fois que les observations sont associées à un amer particulier dans la carte, l'estimation de l'état peut être mise à jour en utilisant la matrice de gain $\mathbf{W}(k)$. La matrice de gain fournit une somme pondérée de la prédiction et de l'observation. Elle est calculée en utilisant la matrice de covariance de l'innovation $\mathbf{S}(k)$, et la prédiction de la matrice de covariance, $\mathbf{P}(k|k-1)$. Le facteur

de pondération est proportionnel à $\mathbf{P}(k|k-1)$ et inversement proportionnel à la covariance de l'innovation [Smith *et al.*, 1990]. Ceci peut est utilisé pour mettre à jour l'estimation du vecteur d'état du système $\hat{\mathbf{X}}(k|k)$ et sa matrice de covariance $\mathbf{P}(k|k)$. $\nu(k)$ est la fonction d'innovation.

$$\hat{\mathbf{X}}(k|k) = \hat{\mathbf{X}}(k|k-1) + \mathbf{W}(k)\,\nu(k) \tag{5.40}$$
$$\mathbf{P}(k|k) = \mathbf{P}(k|k-1) - \mathbf{W}(k)\,\mathbf{S}(k)\,\mathbf{W}^T(k) \tag{5.41}$$

où

$$\mathbf{W}(k) = \mathbf{P}(k|k-1)\,\nabla_{\mathbf{X}}\mathbf{h}^T\,\mathbf{S}^{-1}(k) \tag{5.42}$$

On peut accélérer le calcul en profitant de l'équation 5.19 (voir la figure 5.4) :

$$\mathbf{W}(k) = \begin{bmatrix} \mathbf{P}_{vv}(k|k-1) & \mathbf{P}_{vm}(k|k-1) \\ \mathbf{P}_{vm}^T(k|k-1) & \mathbf{P}_{mm}(k|k-1) \end{bmatrix} \begin{bmatrix} \nabla_v\mathbf{h}^T \\ 0 \\ \vdots \\ 0 \\ \nabla_i\mathbf{h}^T \\ 0 \\ \vdots \end{bmatrix} \mathbf{S}^{-1} \tag{5.43}$$

donc :

$$\mathbf{W}(k) = \begin{bmatrix} \mathbf{P}_{vv}(k|k-1)\nabla_v\mathbf{h}^T + \mathbf{P}_{vi}(k|k-1)\nabla_i\mathbf{h}^T \\ \mathbf{P}_{vm}^T(k|k-1)\nabla_v\mathbf{h}^T + \mathbf{P}_{mi}(k|k-1)\nabla_i\mathbf{h}^T \end{bmatrix} \mathbf{S}^{-1} \tag{5.44}$$

5.2.7 Le Processus d'Estimation :

Fondé sur le modèle du véhicule et sur le modèle des observations, modèles développés dans les sections précédentes, le processus de Localisation et Cartographie consiste à générer les meilleures estimations pour l'état du système en utilisant les informations disponibles. Ceci peut être accompli par une procédure récursive à trois étapes, prédiction, observation et mise à jour connue comme le Filtre de Kalman Étendu (EKF) [Dissanayake *et al.*, 2000]. Le filtre de Kalman est un estimateur récursif et produit à l'instant i l'estimation optimale (qui minimise la somme des erreurs quadratiques) $\hat{x}(i|j)$ de l'état $x(i)$ étant donné les observations jusqu'à l'instant j, $Z^j = \{z(1)\ldots z(j)\}$:

$$\hat{x}(i|j) = E\Big[x(i)|Z^j\Big] \tag{5.45}$$

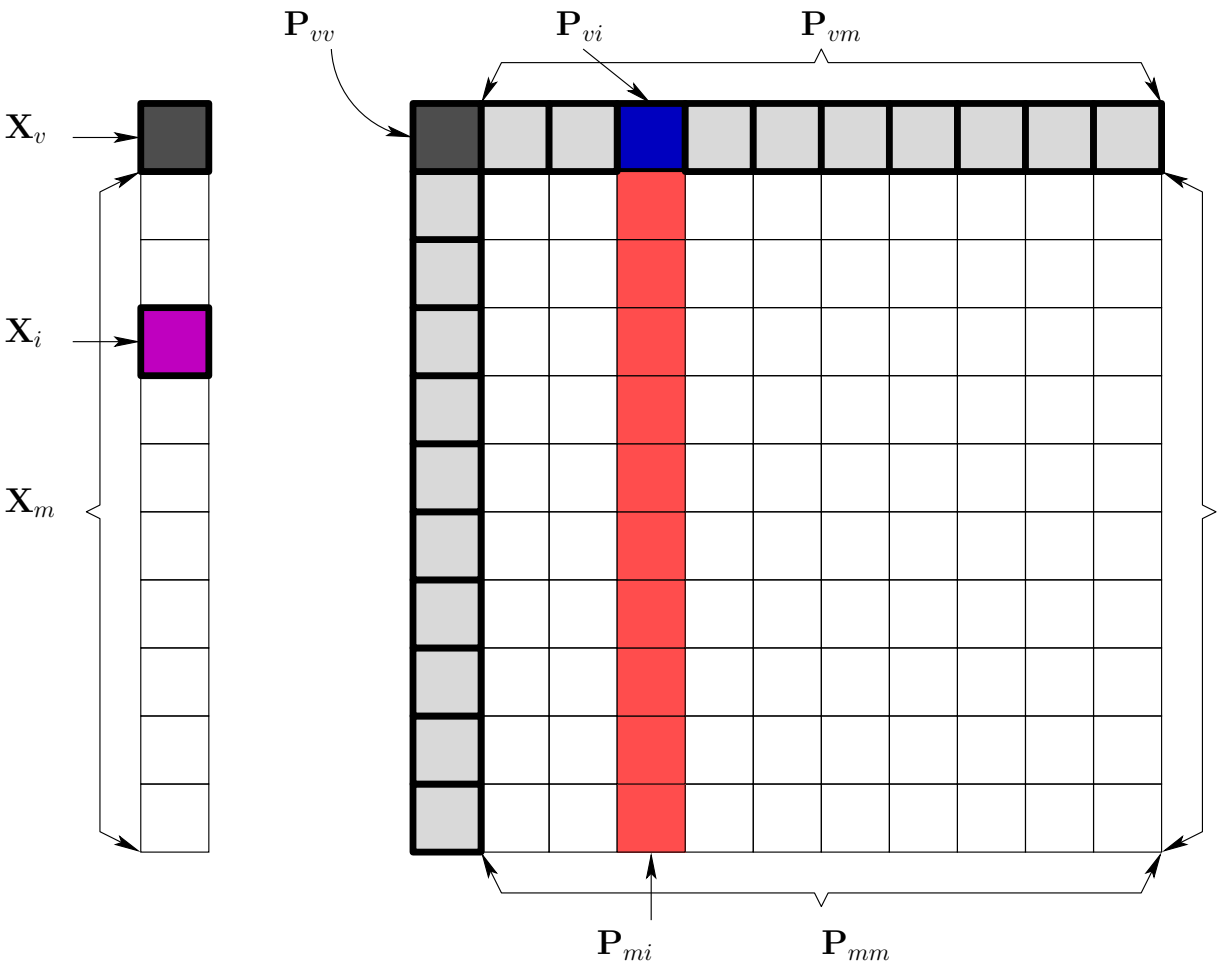


FIG. 5.4 – La matrice de covariance

Les détails sur le développement du filtre de Kalman étendu peuvent être trouvés dans de nombreux livres [Maybeck, 1982; Miller et Leskiw, 1987].

Le filtre de Kalman fait l'hypothèse simplicatrice que le bruit de processus $v(k)$ et le bruit d'observation $w(k)$ sont non corrélés et de moyennes nulles :

$$E\Big[v(k)\Big] = E\Big[w(k)\Big] = 0, \forall k \tag{5.46}$$

et de matrices de covariances $\mathbf{Q}(k)$ et $\mathbf{R}(k)$ respectivement :

$$\begin{aligned} E\Big[v(k)v^T(k)\Big] &= \mathbf{Q}(k) \\ E\Big[w(k)w^T(k)\Big] &= \mathbf{R}(k) \end{aligned} \tag{5.47}$$

De plus, il faut modéliser l'inexactitude de l'entrée du système, $u(k)$. Elle peut être modélisée comme un bruit non corrélé à moyenne nulle :

$$E\Big[u(k)\Big] = 0, \forall k \tag{5.48}$$

avec une matrice de covariance $U(k)$

$$E\Big[u(k)u^T(k)\Big] = U(k) \tag{5.49}$$

Pour l'algorithme SLAM, le EKF est utilisé pour estimer la position du véhicule $\hat{X}_v(k|k)$ avec les paramètres de n_f amers observés $\hat{X}_i(k|k), i = 1 \dots n_f$. Le vecteur d'état augmenté (équation 5.1) montre que ce vecteur est composé de l'état estimé du véhicule et des estimations des amers observés :

$$\hat{\mathbf{X}}(k|k) = \begin{bmatrix} \hat{\mathbf{X}}_v(k|k) \\ \hat{\mathbf{X}}_1(k|k) \\ \vdots \\ \hat{\mathbf{X}}_{n_f}(k|k) \end{bmatrix} \tag{5.50}$$

et la matrice de covariance est définie comme

$$\mathbf{P}(k|k) = E\left[\Big(\mathbf{X}(k) - \hat{\mathbf{X}}(k|k)\Big)\Big(\mathbf{X}(k) - \hat{\mathbf{X}}(k|k)\Big)^T \Big| Z^k\right] \tag{5.51}$$

Cette matrice de covariance peut être développée sous la forme

$$\mathbf{P}(k|k) = \begin{bmatrix} \mathbf{P}_{vv}(k|k) & \mathbf{P}_{v1}(k|k) & \dots & \mathbf{P}_{vn_f}(k|k) \\ \mathbf{P}^T_{v1}(k|k) & \mathbf{P}_{11}(k|k) & \dots & \mathbf{P}_{1n_f}(k|k) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}^T_{vn_f}(k|k) & \mathbf{P}^T_{1n_f}(k|k) & \dots & \mathbf{P}_{n_fn_f}(k|k) \end{bmatrix} \tag{5.52}$$

où

$$\mathbf{P}_{vv}(k|k) = E\left[\left(\mathbf{X}_v(k) - \hat{\mathbf{X}}_v(k|k)\right)\left(\mathbf{X}_v(k) - \hat{\mathbf{X}}_v(k|k)\right)^T \Big| Z^k\right] \tag{5.53}$$

représente la matrice de covariance sur la position du robot, et

$$\mathbf{P}_{ii}(k|k) = E\left[\left(\mathbf{X}_i(k) - \hat{\mathbf{X}}_i(k|k)\right)\left(\mathbf{X}_i(k) - \hat{\mathbf{X}}_i(k|k)\right)^T \Big| Z^k\right] \tag{5.54}$$

représente la matrice de covariance sur les paramètres de l'amer, alors que

$$\mathbf{P}_{vi}(k|k) = E\left[\left(\mathbf{X}_v(k) - \hat{\mathbf{X}}_v(k|k)\right)\left(\mathbf{X}_i(k) - \hat{\mathbf{X}}_i(k|k)\right)^T \Big| Z^k\right] \tag{5.55}$$

représente la matrice de covariance croisée entre la position du robot et les paramètres de l'amer i, et enfin

$$\mathbf{P}_{ij}(k|k) = E\left[\left(\mathbf{X}_i(k) - \hat{\mathbf{X}}_i(k|k)\right)\left(\mathbf{X}_j(k) - \hat{\mathbf{X}}_j(k|k)\right)^T \Big| Z^k\right] \tag{5.56}$$

représente la matrice de covariance croisée entre les amers i et j. La matrice de covariance globale peut être écrite en notant $\mathbf{P}mm(k|k)$ la matrice de covariance de la carte, et $\mathbf{P}_{vm}(k|k)$ la matrice de covariance entre le véhicule et la carte :

$$\mathbf{P}(k|k) = \begin{bmatrix} \mathbf{P}_{vv}(k|k) & \mathbf{P}_{vm}(k|k) \\ \mathbf{P}_{vm}^T(k|k) & \mathbf{P}_{mm}(k|k) \end{bmatrix} \tag{5.57}$$

5.2.8 Initialisation d'un amer

Quand un amer est observé pour la première fois, son estimation doit être initialisée proprement et ajoutée au vecteur d'état. Étant donné l'estimation courante de l'état $\hat{\mathbf{X}}(k|k-1)$, composé de l'état de véhicule, $\hat{\mathbf{X}}_v(k|k-1)$, et de l'état de la carte, $\hat{\mathbf{X}}_m(k|k-1)$, une observation relative entre le véhicule et un nouvel amer, $Z(k)$, et soit le modèle d'initialisation d'amer, $\mathbf{g}_i(.,.)$, qui relie l'état estimé courant du véhicule et l'observation du nouvel amer, l'estimation initiale de l'état du nouvel amer est :

$$\hat{\mathbf{X}}_i(k|k) = \mathbf{g}_i\Big(\hat{\mathbf{X}}_v(k|k-1), Z(k)\Big) \tag{5.58}$$

L'estimation du nouvel amer est ajoutée au vecteur d'état comme un nouvel élément de la carte.

La covariance de l'estimation du nouvel amer doit être initialisée proprement. Cette estimation initiale dépend de l'estimation courante de l'état du véhicule, et donc sera corrélée à la fois à l'état du véhicule et aux estimations des autres amers dans la carte. Ignorer la corrélation de ce nouvel amer aux autres éléments de la carte peut provoquer une incohérence dans le processus du filtrage [Csorba, 1997]. La matrice de covariance doit au début être augmentée par la variance du nouvel amer observé et par les covariances croisées entre les éléments existants de la carte et ce nouvel amer. Supposons que la matrice de covariance est :

$$\mathbf{P}(k|k-1) = \begin{bmatrix} \mathbf{P}_{vv}(k|k-1) & \mathbf{P}_{vm}(k|k-1) \\ \mathbf{P}_{vm}^T(k|k-1) & \mathbf{P}_{mm}(k|k-1) \end{bmatrix} \tag{5.59}$$

La matrice de covariance est augmentée par la covariance de l'observation, $\mathbf{R}(k)$

$$\mathbf{P}^*(k|k-1) = \begin{bmatrix} \mathbf{P}_{vv}(k|k-1) & \mathbf{P}_{vm}(k|k-1) & 0 \\ \mathbf{P}_{vm}^T(k|k-1) & \mathbf{P}_{mm}(k|k-1) & 0 \\ 0 & 0 & \mathbf{R}(k) \end{bmatrix} \tag{5.60}$$

La covariance finale est calculée par projection de la matrice augmentée de covariance à travers le Jacobien $\nabla_X \mathbf{g}(k)$ de la fonction d'initialisation, $\mathbf{g}$, par rapport à l'état augmenté

$$\mathbf{P}(k|k) = \nabla_X \mathbf{g}(k)\ \mathbf{P}^*(k|k-1)\ \nabla_X \mathbf{g}^T(k) \tag{5.61}$$

où

$$\nabla_X \mathbf{g}(k) = \begin{bmatrix} I_v & 0 & 0 \\ 0 & I_m & 0 \\ \nabla_v \mathbf{g}(k) & 0 & \nabla_Z \mathbf{g}(k) \end{bmatrix} \tag{5.62}$$

cette opération est $O(n^3)$, et peut être simplifiée en exploitant la nature éparse du Jacobien $\nabla_X \mathbf{g}(k)$:

$$\mathbf{P}(k|k) =$$

$$\begin{bmatrix} \mathbf{P}_{vv}(k|k-1) & \mathbf{P}_{vm}(k|k-1) & \mathbf{P}_{vv}(k|k-1)\nabla_v \mathbf{g}^T(k) \\ \mathbf{P}_{vm}^T(k|k-1) & \mathbf{P}_{mm}(k|k-1) & \mathbf{P}_{vm}^T(k|k-1)\nabla_v \mathbf{g}^T(k) \\ \nabla_v \mathbf{g}(k)\mathbf{P}_{vv}^T(k|k-1) & \nabla_v \mathbf{g}(k)\mathbf{P}_{vm}(k|k-1) & \nabla_v \mathbf{g}(k)\mathbf{P}_{vv}(k|k-1)\nabla_v^T \mathbf{g}(k) + \nabla_Z \mathbf{g}(k)\mathbf{R}(k)\nabla_Z \mathbf{g}(k)^T \end{bmatrix} \tag{5.63}$$

Dans le cas d'amer plan, la fonction d'initialisation, $\mathbf{g}$ est donnée par :

$$\begin{cases} \rho_w &= \rho_r + \cos(\varphi_r + \theta_v)\sin\psi_r \; x_v \\ & \quad + \sin(\varphi_r + \theta_v)\sin\psi_r \; y_v \\ \varphi_w &= \varphi_r + \theta_v \\ \psi_w &= \psi_r \end{cases} \tag{5.64}$$

C'est une fonction $\mathbf{g}(\mathbf{X}_v, Z(k))$ avec $Z(k) = [\rho_r, \varphi_r, \psi_r]^T$. Calculons les Jacobiens :

$$\nabla_v \mathbf{g}(k) = \frac{\partial \mathbf{g}}{\partial \mathbf{X}_v} = \begin{bmatrix} \frac{\partial \rho_w}{\partial x_v} & \frac{\partial \rho_w}{\partial y_v} & \frac{\partial \rho_w}{\partial \theta_v} \\ \frac{\partial \varphi_w}{\partial x_v} & \frac{\partial \varphi_w}{\partial y_v} & \frac{\partial \varphi_w}{\partial \theta_v} \\ \frac{\partial \psi_w}{\partial x_v} & \frac{\partial \psi_w}{\partial y_v} & \frac{\partial \psi_w}{\partial \theta_v} \end{bmatrix} \tag{5.65}$$

donc :

$$\nabla_v \mathbf{g}(k) = \begin{bmatrix} \cos(\varphi_r + \theta_v)\sin\psi_r & \sin(\varphi_r + \theta_v)\sin\psi_r & \alpha_1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.66}$$

avec :
$\alpha_1 = -\sin(\varphi_r + \theta_v)\sin\psi_r \; x_v + \cos(\varphi_r + \theta_v)\sin\psi_r \; y_v$

et :

$$\nabla_z \mathbf{g}(k) = \frac{\partial \mathbf{g}}{\partial Z} = \begin{bmatrix} \frac{\partial \rho_w}{\partial \rho_r} & \frac{\partial \rho_w}{\partial \varphi_r} & \frac{\partial \rho_w}{\partial \psi_r} \\ \frac{\partial \varphi_w}{\partial \rho_r} & \frac{\partial \varphi_w}{\partial \varphi_r} & \frac{\partial \varphi_w}{\partial \psi_r} \\ \frac{\partial \psi_w}{\partial \rho_r} & \frac{\partial \psi_w}{\partial \varphi_r} & \frac{\partial \psi_w}{\partial \psi_r} \end{bmatrix} \tag{5.67}$$

alors :

$$\nabla_z \mathbf{g}(k) = \begin{bmatrix} 1 & \beta_1 & \beta_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.68}$$

avec :

$$\beta_1 = -\sin(\varphi_r + \theta_v)\sin\psi_r\, x_v + \cos(\varphi_r + \theta_v)\sin\psi_r\, y_v$$

$$\beta_2 = \cos(\varphi_r + \theta_v)\cos\psi_r\, x_v + \sin(\varphi_r + \theta_v)\cos\psi_r\, y_v$$

5.3 Amers lignes 2D dans la Carte

Afin d'enrichir la représentation de l'environnement d'intérieur, nous prenons en compte les segments de droites qui sont aussi omniprésents : coins, bords des portes, bords des posters, etc. Pour les inclure dans la carte stochastique, nous avons choisi comme expliqué en section 4.7, d'utiliser les segments de droites 3D comme des segments de droite 2D portés par un plan (déjà inclus dans la carte). L'état d'un amer plan (voir section 5.2) qui est considéré comme un plan infini est donné par $\mathbf{X}_{\pi,j}(k) = [\rho_j(k), \varphi_j(k), \psi_j(k)]^T$. Un segment 2D attaché à un plan peut être considéré comme une droite infinie 2D, et peut être représenté de manière optimale par deux paramètres : δ la distance à l'origine et γ l'angle avec l'axe des abscisses. L'état d'un amer segment 2D dans la carte est donné par $\mathbf{X}_{L,i}(k) = [\delta_i(k), \gamma_i(k)]^T$. Bien sûr un amer plan peut porter plusieurs amers lignes 2D, mais un amer ligne 2D ne peut pas être défini sans un amer plan porteur. La carte stochastique devient donc hétérogène : elle contient deux types d'amers.

Notons que dans nos perspectives proches, nous souhaitons aussi introduire des points d'intérêt 2D, également portés par des amers plans, dans cette carte hétérogène.

5.3.1 Observation des amers lignes 2D

Le processus de mise à jour de la carte est le suivant : d'abord on met à jour les amers plans en utilisant les données laser 3D seulement, ensuite on met à jour les amers lignes 2D sur chaque plan. Ceci peut être traduit par le fait que lors de la mise à jour des amers lignes 2D, les plans porteurs sont déjà à jour dans la carte.

Fonction d'innovation

Nous définissons la fonction d'innovation de l'amer ligne 2D par :

$$\nu = \begin{bmatrix} \delta_i^{\pi,M} - \delta_i^{\pi} \\ \\ \gamma_i^{\pi,M} - \gamma_i^{\pi} \end{bmatrix} \tag{5.69}$$

où $(\delta_i^{\pi,M},\ \gamma_i^{\pi,M})$ sont les paramètres du segment 2D dans le repère amer plan (qui fait partie de la carte), et $(\delta_i^{\pi},\ \gamma_i^{\pi})$ sont les paramètres du segment 2D dans le repère plan (perçu dans la scène courante).

Dans l'algorithme SLAM "traditionnel" (voir par exemple SLAM avec des amers plans), la fonction d'innovation s'écrit :

$$\nu(k) = \mathbf{Z}(k) - \hat{\mathbf{Z}}(k|k-1) \tag{5.70}$$

avec :

- $\mathbf{Z}(k)$: la mesure courante, c'est-à-dire les plans extraits à partir des données laser 3D dans le repère robot.
- $\hat{\mathbf{Z}}(k|k-1)$: l'estimation des mesures, c'est-à-dire, comment sont placés dans le repère robot, les plans de la carte stochastique, c'est-à-dire la position relative au robot des amers déjà inclus dans la carte. Ainsi l'estimation des mesures est une fonction de l'état prédit :

$$\hat{\mathbf{Z}}(k|k-1) = h(\hat{\mathbf{X}}(k|k-1)) \tag{5.71}$$

En revanche, dans notre approche de SLAM avec amers lignes 2D collés sur amers plans, la fonction d'innovation s'écrit toujours :

$$\nu = \mathbf{Z}(k) - \hat{\mathbf{Z}}(k|k-1) \tag{5.72}$$

mais avec :

- $\mathbf{Z}(k)$: la mesure courante de l'amer ligne 2D collé sur un amer plan. En fait, on obtient cette mesure à partir des données issues de la caméra ($\mathbf{X}_I(k)$) fusionnées avec les données de l'amer plan porteur (voir section 4.7). Ainsi la mesure courante est une fonction de l'état courant prédit du système et des données caméra. Cette mesure est représentée dans le repère amer plan.

$$\mathbf{Z}(k) = h\left(\hat{\mathbf{X}}(k|k-1), \mathbf{X}_I(k)\right) \tag{5.73}$$

– $\hat{\mathbf{Z}}(k|k-1)$: l'estimation des mesures, c'est-à-dire, comment sont placés les amers lignes de la carte stochastique par rapport au repère amer plan porteur. En fait comme les amers lignes sont déjà attachés au plan porteur, on peut considérer :

$$\hat{\mathbf{Z}}(k|k-1) = \hat{\mathbf{Z}}(k-1|k-1) \tag{5.74}$$

Matrice de Covariance de l'innovation

$$\mathbf{S}(k) = \mathbf{P}_{ii}(k|k-1) + \mathbf{R}(k) \tag{5.75}$$

avec $\mathbf{P}_{ii}(k|k-1)$: la matrice de covariance de l'amer ligne 2D numéro i, et $\mathbf{R}(k)$ est la matrice de covariance de la mesure courante. Or, comme on a détaillé auparavant, on peut écrire :

$$\mathbf{Z}(k) = h\left(\hat{\mathbf{X}}_v(k|k-1),\, \hat{\mathbf{X}}_j(k|k-1),\, \mathbf{X}_I(k)\right) \tag{5.76}$$

où $\hat{\mathbf{X}}_j(k|k-1)$ est l'état prédit de l'amer plan porteur, et $\mathbf{X}_I(k)$ représente les données caméra (les paramètres de la ligne 2D dans l'image) dont la matrice de covariance est Λ_I.

Soient $\nabla_v h$, $\nabla_j h$ et $\nabla_I h$ les Jacobiennes de la fonction h par rapport à l'état du robot, du plan amer porteur j et de la ligne 2D dans l'image l_I respectivement, on écrit :

$$\begin{aligned} \mathbf{R}(k) = \quad & \nabla_v h\ \mathbf{P}_{vv}\ \nabla_v h^T \\ & +\nabla_v h\ \mathbf{P}_{vj}\ \nabla_j h^T \\ & +\nabla_j h\ \mathbf{P}_{jj}\ \nabla_j h^T \\ & +\nabla_j h\ \mathbf{P}_{vj}^T\ \nabla_v h^T \\ & +\nabla_I h\ \Lambda_I\ \nabla_I h^T \end{aligned} \tag{5.77}$$

Dans cette dernière équation, nous identifions clairement le rôle des données de la caméra qui ne sont pas corrélées ni avec les données laser ni avec les données d'odomètre. Ainsi, nous pouvons justifier notre choix des amers lignes 2D attachés aux amers plans porteurs. Le fait qu'un amer ligne 2D est attaché à un amer plan porteur peut faire croire que ces deux amers sont corrélés. En effet, l'équation 5.77 démontre qu'un amer plan n'est pas corrélé avec les amers lignes 2D attachés à lui. Les données caméra qui participent à définir un amer ligne 2D sont totalement non corrélées avec les données laser (qui identifient l'amer plan porteur). Ceci donne une preuve théorique que les deux amers ne sont pas corrélés. Avec cette technique, nous arrivons à fusionner les données des deux capteurs, et à les intégrer de manière cohérente dans une carte hétérogène (plan + ligne 2D).

5.4 Association de Données

L'association de données est le processus de rechercher des correspondances entre les amers perçus et les amers déjà enregistrés dans la carte. En d'autres termes, de pouvoir décider si les amers trouvés dans la scène courante sont des amers déjà vus dans les scènes précédentes. L'association de données est cruciale pour garantir la cohérence de la carte. En fait une correspondance erronée peut provoquer la divergence du filtre, et ainsi la non cohérence de la carte stochastique [Neira et Tardós, 2001].

5.4.1 L'association de données fondée sur la distance de Mahalanobis

Rappelons tout d'abord les définitions de la distribution de χ^2 et de la distance de Mahalanobis.

Définition de la Distribution χ^2

La distribution du Chi-2 (notée χ^2) est la somme des carrés des observations issues de la distribution $N(0,1)$. Soient $X_1, X_2, \ldots, X_n$: n variables indépendantes et toutes $\sim N(0,1)$. Alors la distribution χ^2_n est définie comme celle de la somme $X_1^2 + X_2^2 + \ldots + X_n^2$

$$\left(X_1^2 + X_2^2 + \ldots + X_n^2\right) \sim \chi^2_n \tag{5.78}$$

Il n'y a donc pas une distribution du χ^2, mais une famille de distributions indexée par le paramètre entier n. Ce paramètre est appelé "nombre de degrés de liberté" de la distribution. La distribution "Chi-2 à n degrés de liberté" est donc définie comme celle de la somme des carrés de n variables indépendantes toutes $\sim N(0,1)$.

Dans le cas où X est une variable normale quelconque : $X \sim N(\mu, \sigma^2)$, la variable centrée et normalisée $\frac{X-\mu}{\sigma}$ est une variable normale standard $\sim N(0,1)$. Dans ce cas, la somme des carrés des variables standardisées est de distribution χ^2.

$$\left((\frac{X_1-\mu_1}{\sigma_1})^2 + (\frac{X_2-\mu_2}{\sigma_2})^2 + \ldots + (\frac{X_n-\mu_n}{\sigma_n})^2\right) \sim \chi^2_n \tag{5.79}$$

La Distance de Mahalanobis

Soient $\mathbf{z}$ une variable aléatoire de moyenne et covariance $(\hat{\mathbf{z}}, \Lambda_{\mathbf{z}})$, et $\mathbf{x}$ une variable aléatoire de moyenne et covariance $(\hat{\mathbf{x}}, \Lambda_{\mathbf{x}})$. Comment déterminer d'une façon plausible si ces deux variables vérifient une relation $\mathbf{f}(\mathbf{z}, \mathbf{x}) = 0$. Si on suppose que les $\mathbf{x}$ et $\mathbf{z}$ sont des gaussiennes centrées

indépendantes, et que la relation est vérifiée, donc $\mathbf{f}(\hat{\mathbf{z}}, \hat{\mathbf{x}})$ est une variable gaussienne centrée de matrice de covariance :

$$\begin{aligned} \mathbf{Q} &= E\left[\mathbf{f}(\hat{\mathbf{z}}, \hat{\mathbf{x}})\, \mathbf{f}(\hat{\mathbf{z}}, \hat{\mathbf{x}})^T\right] \\ &= \nabla_z \mathbf{f}\ \Lambda_z\ \nabla_z \mathbf{f}^T + \nabla_x \mathbf{f}\ \Lambda_x\ \nabla_x \mathbf{f}^T \end{aligned} \tag{5.80}$$

La **distance de Mahalanobis généralisée** est donnée par :

$$d(\hat{\mathbf{z}}, \hat{\mathbf{x}}) = \mathbf{f}(\hat{\mathbf{z}}, \hat{\mathbf{x}})^T\ \mathbf{Q}\ \mathbf{f}(\hat{\mathbf{z}}, \hat{\mathbf{x}}) \tag{5.81}$$

C'est une variable aléatoire ayant la distribution de probabilité d'un χ^2. Donc, en consultant les tables de χ^2, on peut déterminer un seuil de confiance qui correspond à une probabilité 95% (par exemple) que la relation $\mathbf{f}(\mathbf{z}, \mathbf{x}) = 0$ soit vérifiée. Par exemple :

$$P\left(\chi^2_3 \geq 7.8147\right) = 0.05 \tag{5.82}$$

Donc :

$$P\left(0 \leq \chi^2_3 \leq 7.8147\right) = 0.95 \tag{5.83}$$

Dans le cas des primitives planes, notons $[\rho_{map}, \varphi_{map}, \psi_{map}]^T$ les paramètres d'un plan dans la carte stochastique, et $[\rho_w, \varphi_w, \psi_w]^T$ une nouvelle observation du plan exprimée dans le repère monde. La fonction $\mathbf{f}$ est tout simplement la différence entre ces deux vecteurs. Soient Λ_{map} et Λ_w respectivement les matrices de covariance.

$$\Lambda_f = \Lambda_{map} + \Lambda_w \tag{5.84}$$

$$d(\mathbf{x}_{map}, \mathbf{z}_w) = \begin{bmatrix} \rho_{map} - \rho_w \\ \varphi_{map} - \varphi_w \\ \psi_{map} - \psi_w \end{bmatrix}^T \quad \Lambda_f \quad \begin{bmatrix} \rho_{map} - \rho_w \\ \varphi_{map} - \varphi_w \\ \psi_{map} - \psi_w \end{bmatrix} \tag{5.85}$$

L'algorithme de *Nearest Neighbour*

Un algorithme classique d'association de données permet d'apparier les amers en utilisant leurs représentations géométriques paramétrisées. Ces représentations sont sauvegardées dans la carte et mises à jour après chaque observation. Typiquement, on utilise : les points (3 coordonnées cartésiennes), les lignes 3D infinies (4 paramètres), les plans (3 paramètres), les positions des objets (6 paramètres donnant les transformations rigides), etc. Soit par exemple L un vecteur

représentant les paramètres pour définir un amer. Soient L_r les paramètres définissant un amer L observé dans le repère robot $\mathcal{R}_r$, et L_w les paramètres d'un amer enregistré dans la carte représentée dans le repère monde $\mathcal{R}_w$. Soit $\mathbf{X}_v$ le vecteur d'état du robot dans le repère monde. Associer le vecteur L_r au vecteur L_w nécessite l'existence d'une fonction différentiable F qui exprime le vecteur L_w en fonction du vecteur L_r et de $\mathbf{X}_v$.

$$L_w = F(L_r, \mathbf{X}_v) \tag{5.86}$$

La procédure d'appariement basée sur les représentations paramétrisées des amers est composée de deux étapes [Neira et Tardós, 2001] :

- D'abord, le carré de l'innovation normalisée ($||L_w - F(L_r, \mathbf{X}_v)||$) est calculé pour déterminer la compatibilité entre les paramètres de L_r et de L_w qui décrivent un appariement potentiel entre l'amer observé et l'amer enregistré, en se basant sur la prédiction de l'état du robot $\mathbf{X}_v$ (qui est prédite en utilisant les données odométriques).

- Ensuite, l'algorithme du plus proche voisin (*Nearest Neighbour* NN) est utilisé pour choisir l'appariement. En effet, l'algorithme de *Nearest Neighbour* est un test pour déterminer la distance de Mahalanobis la plus petite.

Les incertitudes des amers observés (*sensor model*) et de la prédiction de l'état du robot (*robot model*) sont prises en compte par les matrices de covariances des vecteurs L_r et $\mathbf{X}_v$ en utilisant les Jacobiens de la fonction F pour propager les erreurs d'observation et de prédiction.

Le test d'appariement basé seulement sur les mesures géométriques est non robuste. D'autres informations rendant cet appariement plus évident sont sollicitées. Ainsi, une étape supplémentaire permettant de confirmer l'hypothèse d'appariement donné par l'algorithme de *Nearest Neighbour* sera directement recommandée. Ces vérifications peuvent être faites en utilisant les informations (non formelles) connues pour chaque amer. Typiquement, des informations d'apparence extraites de l'image : couleur d'un point, l'intensité moyenne le long d'un segment, etc. De manière générale, les extrémités d'un segment et les contours d'une facette plane sont difficilement exploitable à cause de leur sensibilité à l'occultation, en particulier pour les segments longs (segment formé par le sol et un mur par exemple) ou les facettes larges (murs).

5.4.2 L'association de données fondée sur l'apparence

L'apparence est utilisée en SLAM visuel (comme par exemple dans les travaux de Davison [Davison, 2003], Lemaire [Lemaire *et al.*, 2005], Montiel, Sola [Sola *et al.*, 2005], Hayet [Hayet *et al.*, 2003] etc.) ou dans SLAM multi-capteurs combinant des données laser 2D et les données visuelles d'une caméra (comme par exemple les travaux de Borges, Tardos, etc.). Newman et al. [Posner *et al.*, 2007],[Ho et Newman, 2007] utilisent des primitives basées sur l'apparence pour détecter la fermeture de boucle. Angeli et al. [Angeli *et al.*, 2008] utilisent une approche inspirée de la méthode de catalogue de mots visuels (*bag of visual words*) connue dans l'indexation d'images pour consolider la fermeture de boucle.

[Gil *et al.*, 2007] construisent une carte des points 3D extraits par vision monoculaire. Ils ajoutent pour chaque point tridimensionnel un descripteur visuel discriminant (vecteur SIFT, pour *Scale Invariant Feature Transform*). Une approche retardée (*delayed*) est utilisée pour l'initialisation des amers : ils suivent (tracking visuel) les points d'intérêt sur plusieurs images de la scène avant de les ajouter à la carte. Ils traitent le problème d'association de données comme un problème de classification de patterns, en considérant que chaque amer visuel i est une classe C_i, et que les différentes observations de cet amer sont des éléments dans la classe C_i. Ils utilisent ces descripteurs pour calculer la valeur moyenne et la matrice de covariance du descripteur de l'amer visuel.

Les amers plans ont deux représentations : d'abord, dans la carte stochastique, un amer plan est représenté par une représentation géométrique (3 paramètres, voir la section 4.6.2). Ensuite, les informations d'apparence sont enregistrées en utilisant des primitives visuelles locales (segments ou points) ou par une image virtuelle :

- Si la facette plane a une couleur uniforme (typiquement un mur avec peinture uniforme), l'image virtuelle n'est pas construite, seules les primitives visuelles (les bords des portes, les contours des posters, etc.) sont mémorisées par leurs caractéristiques géométriques 2D. La couleur de la facette est apprise comme un modèle Gaussien (moyenne, variance). La couleur est utilisée pour initialiser et générer des appariements potentiels entre les amers enregistrés et observés. Ensuite, pour chaque appariement potentiel, les primitives visuelles sont retro-projetées sur la facette enregistrée dans la carte en utilisant la prédiction de l'état du robot, et l'association de données entre les amers visuels observés et enregistrés est effectuée sur le plan porteur de la facette.

- Si la facette n'est pas uniforme (à cause de la texture des papiers peints par exemple), l'image virtuelle est construite pour cette facette. Voir la section 4.8 pour plus de détails concernant le calcul de l'homographie qui permet de plaquer l'image de la caméra sur un plan extrait par l'algorithme de segmentation de l'image de profondeur 3D issue du scanner laser. Des mesures de corrélations sont calculées pour calculer la vraisemblance entre les textures de l'amer observé et de l'amer enregistré.

- De manière générale, un appariement potentiel entre une facette observée et une facette enregistrée dans la carte est évalué non seulement par la distance de Mahalanobis entre les représentations géométriques, mais aussi en appariant les primitives visuelles qui sont portées par ces amers. En effet, une mesure globale de vraisemblance est utilisée pour rendre l'appariement plus robuste. Cette mesure globale est fondée soit sur les couleurs uniformes soit sur le score de corrélation des textures des facettes observées et enregistrées. Ces deux critères peuvent être combinées ensemble, comme le mur peut être en général uniforme, mais, texturé dans certaines régions (par exemple : un poster est fixé sur un mur). La fusion de ces critères est gérée en prenant en compte des coefficients adaptatifs, qui peuvent être calculés en ligne, en profitant des probabilités d'observation des scènes similaires dans l'environnement.

5.4.3 Les informations d'apparence

Pour chaque plan 3D π_i, les informations inscrites dans la carte sont les paramètres $(\rho_i, \varphi_i, \psi_i)$. Les informations supplémentaires sont inscrites dans une structure de donnée noté $PlaneInf$. Ces structures de données sont liées à la carte stochastique, et doivent être mises à jour après la fusion de chaque nouvelle observation dans la carte. La structure de donnée $PlaneInf$ contient pour le moment :

- Une image discrète notée π_{img}, nous avons choisi la taille de chaque cellule de 2x2 cm. L'image π_{img} est basée sur le repère local de l'amer plan $\mathcal{R}_\pi$, (voir la section 4.6.6 pour le choix de ce repère). La taille de l'image π_{img} associée au plan π_i est adaptée en ligne. Dans l'image π_{img}, l'axe de u (colonne) est parallèle à l'axe O_pX_p, et l'axe de v (ligne) est parallèle à O_pY_p. L'image π_{img} doit être étendue pour recouvrir toute la région incluant des points 3D inclus dans le plan π_i et perçus par la caméra (voir la section 4.8 pour le calcul de l'homographie qui est à l'origine de cette image).

- Dans le cas où le plan est presque totalement perçu (comme le cas d'une porte par exemple), les bords l'image π_{img} de la zone perçue peuvent être interprétés comme les frontières ou les limites du plan.

- Dans le cas où une partie de ces bords de l'image π_{img} est supportée par des lignes droites définies dans la carte stochastique comme des amers lignes 2D (voir la section 4.7), alors cette ligne peut être la frontière entre deux plans voisins ou les bords d'un poster fixé sur le plan porteur.

- La texture d'une facette plane perçue est calculée à l'intérieur de ces bords. De plus, des attributs globaux sont associés à l'image π_{img} : approximation gaussienne de l'histogramme de couleur (moyenne et covariance), et attributs globaux de la texture calculés à partir des histogrammes des sommes et des différences (représentation paramétrique de la texture proposée par Unser [Unser, 1995] : elle donne des résultats similaires aux matrices de cooccurences de Haralick, tout en étant plus rapide à calculer).

- Les amers lignes 2D sont inscrits dans la carte stochastique. Ses extrémités et des attributs globaux (la couleur moyenne, texture sur les cotés gauche et droit) sont enregistrés. Notons que ces attributs pourraient être transférés et représentés dans l'image π_{img}, mais nous avons préféré les garder dans l'espace continu du plan porteur de l'amer plan.

- De plus, dans nos perspectives, les points d'intérêt (Harris ou SIFT) seront intégrés dans la carte stochastique. Par exemple, pour un point d'intérêt, il suffit de garder dans la carte ses coordonnées (x et y) dans le repère local de l'amer plan qui le contient. Les attributs globaux de ces points sont enregistrés : par exemple, un vecteur de 128 éléments pour un point ou "patch" SIFT. Ces points peuvent aussi être représentés dans l'image π_{img}.

5.5 Étapes de construction de la carte hétérogène

Nous détaillons dans cette partie les différentes étapes pour la construction de la carte hétérogène introduite auparavant dans la section 5.3. Tout d'abord, les amers plans sont mis à jour comme expliqué en section 5.2. Dans cette étape, les informations liées à la facette plane sont utilisées pour garantir un appariement robuste. Une fois les amers plans à jour, leurs informations supplémentaires sont mises à jour aussi. Même si les informations d'apparence liées à chaque amer plan ne sont pas enregistrées dans la carte stochastique, elles y sont fortement reliées.

5.5.1 L'étape d'observation

Décrivons le scénario d'observation : à l'instant $t+1$, nous avons une nouvelle mesure par le scanner laser 3D (l'image de profondeur) et une image prise par la caméra.

- D'abord, l'image de profondeur est segmentée par l'algorithme détaillé en section 4.6. La segmentation donne un ensemble de plans dans le repère robot. Nous calculons pour chacun d'eux les informations d'apparence ($PlaneInf$).

- Ensuite, les segments de droites sont extraits de l'image, et on construit les amers lignes 2D attachées aux plans, comme détaillé en section 4.7. Les extrémités de chacun des segments sont projetés dans le repère local du plan, et la couleur moyenne et les attributs de la texture sur les deux cotés sont calculés.

- Enfin, les points d'intérêt sont extraits dans la facette. Chaque point d'intérêt (détecté par un détecteur de Harris ou SIFT) est initialisé dans le repère local du plan porteur. Leurs attributs sont calculés en utilisant les données de l'image autour du point d'intérêt. A terme, ces points seront aussi inclus dans la carte sous la forme de points 2D portés par un amer plan.

5.5.2 L'étape d'appariement

Les données odométriques sont utilisées pour prédire l'état du robot, comme cela a été détaillé auparavant en section 5.2. Des travaux préliminaires sur le SLAM avec des amers plans ont montré que les attributs 3D (l'équation du plan, sa surface et les bords de la facette) ne sont pas suffisants pour obtenir un bon appariement entre les plans [Nashashibi et Devy, 1993]. Nous cherchons un appariement plus robuste entre les primitives perçues et celles déjà inclues dans la carte, en utilisant en plus des données géométriques, les données d'apparence de chaque plan.

La procédure d'association de données contient les étapes suivantes :

- D'abord, les appariements potentiels sont recherchés par la méthode classique de la distance

de Mahalanobis (comme expliqué en section 5.4.1). Ceci donne pour chaque facette plane perçue, la liste des facettes planes déjà inclues dans la carte qui peuvent être fusionnées avec.

- Pour un appariement potentiel de deux plans, les opérations suivantes sont effectuées :

  ◇ Générer l'ensemble des appariements possibles entre les segments lignes 2D sur les deux plans. Un appariement entre deux lignes est vérifié en utilisant en plus des données géométriques qui les définissent, la similarité entre les informations d'apparence des deux segments.

  ◇ Générer l'ensemble des appariements possibles entre les points d'intérêt portés par les deux plans. Ces appariements sont d'abord trouvés grâce aux positions relatives des points d'intérêt sur les deux plans, et de plus grâce aux vecteurs d'attributs liés aux points d'intérêt. La procédure d'appariement proposée par [Jung et Lacroix, 2001] est utilisée pour générer un ensemble cohérent d'appariements entre les points d'intérêt (Harris, SIFT ou SURF).

  ◇ Enfin, un score global est calculé pour chaque appariement potentiel entre un plan perçu et un plan de la carte. Pour le moment, c'est un score heuristique, qui mélange la distance normalisée de l'innovation avec le score donné par les attributs d'apparence.

Les figures 5.5 et 5.6 illustrent les π_{img} du même plan vu dans deux scènes successives. La construction de ces π_{img} était détaillée en section 4.8. Notons que nous choisissons la taille de l'image π_{img} de manière automatique pour qu'elle soit sous la forme $2^M * 2^N$ pour des raisons d'accélération d'affichage sous OpenGL, en gardant toujours le même rapport cm/pixel pour les images, mais en ajoutant des régions noires non utilisées. La figure 5.7 illustre l'utilisation d'une fonction de mise en correspondance de points SIFT (ici, celle proposée par David Lowe), pour apparier des points d'intérêts extraits dans ces deux π_{img}. Nous constatons que malgré la mauvaise résolution des ces images, nous arrivons à apparier (correctement) 13 points.

5.5.3 L'étape de Fusion et la gestion du modèle

Cette étape est classique dans l'algorithme de SLAM pour la mise à jour des amers.

- La carte est mise à jour en utilisant les appariements des facettes planes comme détaillé en section 5.2.

- Pour chaque amer plan dans la carte stochastique, l'image π_{img} est mise à jour. L'image π_{img} est corrigée en utilisant des opérateurs logiques simples entre l'image π_{img} de la facette perçue et celle de la facette dans la carte. Ensuite, les autres informations sont calculées :

  – Si la facette est uniforme, seuls les attributs globaux de la couleur et de la texture sont calculés.

FIG. 5.5 – Pimage de l'amer plan dans une première scène

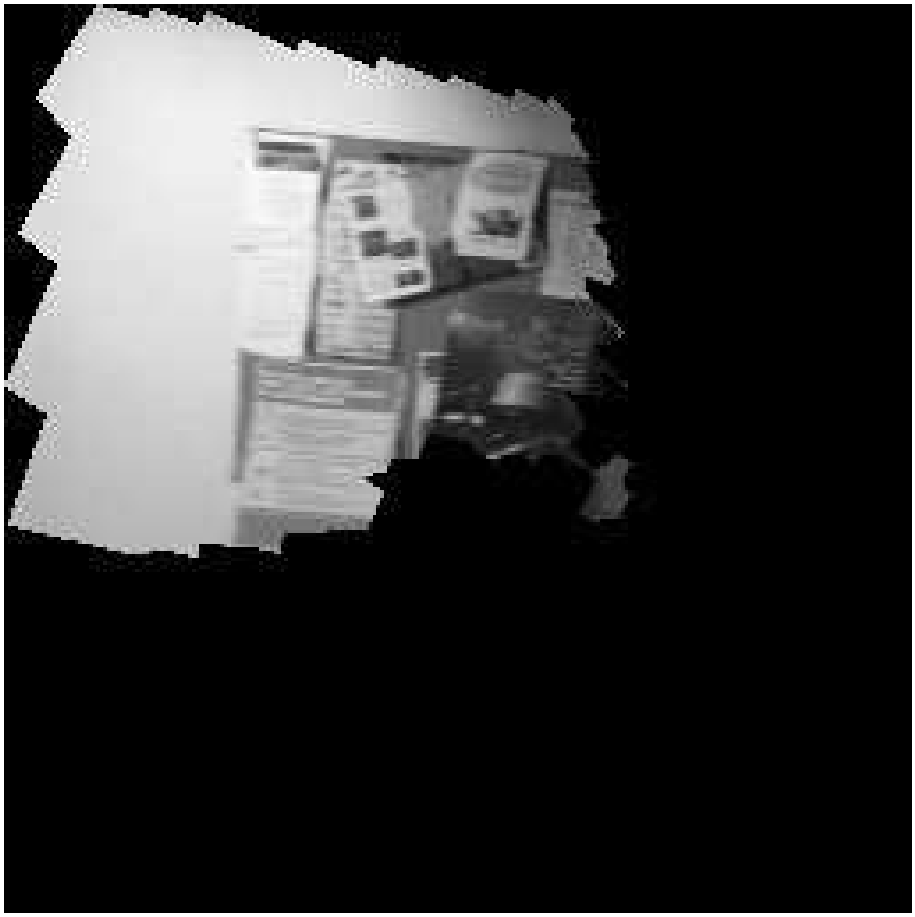


FIG. 5.6 – Pimage du même amer plan dans une autre scène (à la même résolution : l'image est grossie à l'affichage)

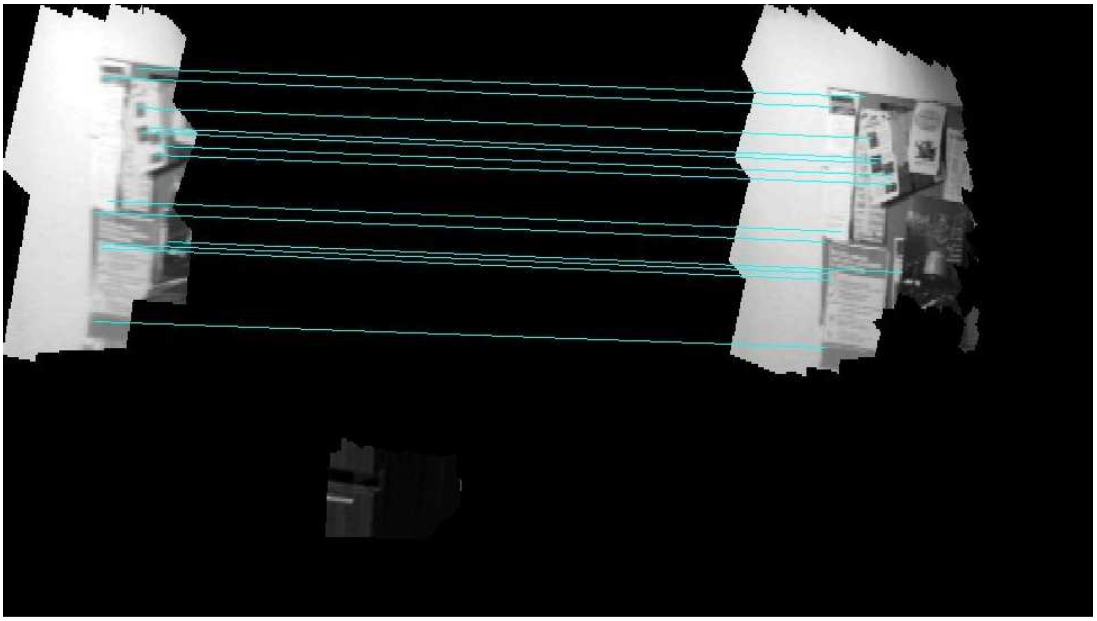


FIG. 5.7 – Appariement de points de SIFT sur deux Pimages d'un même amer plan dans deux scènes successives

  - Sinon, la couleur de chaque cellule dans l'image π_{img} est calculée.

- Créer les nouvelles primitives (amers lignes 2D et points d'intérêt) sur les plans existants dans la carte : toutes les lignes et tous les points d'intérêt non appariés sont ajoutés à la carte stochastique, et leurs attributs d'apparence sont calculés.

- Ajouter les plans non appariés à la carte, et créer leurs attributs associés : l'image π_{img}, les lignes 2D, les points d'intérêt, couleur, etc.

- Enfin, une procédure de ménage peut supprimer les primitives qui ne sont jamais appariées. Pour le moment, nous ne considérons pas cette opération, car nous supposons que l'environnement est statique pendant l'exploration du robot.

5.6 Implémentation et Résultats

Nous allons dans un premier temps présenter les résultats de la construction de la carte stochastiques des amers plans seulement. Les résultats sur la carte hétérogène et texturée sont encore trop partiels pour être présentés dans ce document.

L'implémentation est faite sous l'environnement de développement Jafar [11]. Plusieurs modules ont été intégrés, puis exploités pour réaliser ensemble la construction de la carte stochastique. Un premier module traite de la segmentation de l'image de profondeur. Un second module

[11] http ://www.laas.fr/∼tlemaire/jafar/

se charge de la fusion de données et du calcul de π_{img} pour les facettes planes. Le filtre de Kalman étendu est géré par le module filtre. La gestion de la carte stochastique par l'algorithme SLAM est réalisée via un module SLAM. Enfin un module basé sur OpenGL assure l'affichage 3D des amers plans texturés et des positions successives du robot.

Nous avons fait des mesures dans notre laboratoire, le robot se déplace et s'arrête, prend des mesures, ensuite il avance de nouveau. Il a fait un parcours dans un couloir et a fait un demi tour pour revenir au point de départ. L'algorithme de segmentation plane appliqué sur chaque image de points 3D, a donné des résultats très convaincants. La construction incrémentale de la carte de couloir est illustrée (une partie) dans la figure 5.8, dont nous voyons les points des facettes planes issues de l'algorithme SLAM ; les triangles au sol montrent les positions successives du robot lors de cette séquence d'exploration dans ce large carrefour de couloirs (en rouge, les positions prédites par l'odométrie, et en bleu, les positions estimées par le SLAM). La figure 5.9 représente la scène vue par une caméra pour mieux interpréter les résultats.

5.7 Conclusion et perspectives

Dans ce chapitre, nous avons adapté l'algorithme de *Simultaneous Localization and Mapping* (SLAM) par filtre de Kalman étendu à notre représentation, pour construire une carte des amers plans. Nous avons également défini un algorithme permettant d'extraire à partir des données laser et caméra les segments 3D dans la scène. Pour des raisons d'optimisation des représentations, nous avons choisi de modéliser ces segments 3D comme des lignes 2D sur des amers plans qui les contiennent, par la technique décrite au chapitre précédent. La fonction d'observation de ces amers est exploitée par le formalisme de SLAM de la carte hétérogène : elle contient deux types différents d'amers, les plans et les lignes 2D portées par ces plans. Un seul filtre est utilisé pour gérer la carte hétérogène.

La fusion de données des deux capteurs nous a permis d'ajouter des informations d'apparence aux amers plans. Ces informations d'apparence sont utilisées pour renforcer l'association de données, et par conséquence, assurer une bonne cohérence de la carte obtenue. L'image virtuelle de chaque amer plan est mise à jour par des opérations logiques, et permet une meilleure visualisation du modèle construit par le robot, par un opérateur humain chargé d'interpréter ce modèle.

Nous sommes conscients que ce travail sur le SLAM n'est pas achevé, nous allons donc donner quelques pistes pour les travaux futurs à court terme. Dans une première extension simple, nous étudions l'ajout des points d'intérêt portés sur les amers plans. En fait, un des problèmes majeurs du SLAM visuel est l'initialisation des amers, c'est-à-dire, ou placer l'amer visuel perçu par la caméra dans l'espace. Plusieurs solutions ont été proposées : l'approche non retardé (*undelayed*) [Sola *et al.*, 2005], dans laquelle chaque amer est directement introduit dans la carte avec plusieurs hypothèses, ou l'approche avec retard (*delayed*) [Lemaire *et al.*, 2005], dans laquelle l'amer visuel n'est ajouté à la carte qu'après être perçu plusieurs fois jusqu'à ce que la triangulation soit possible. Le même processus d'initialisation a été proposé pour les segments 3D. En revanche, dans notre approche, nous n'avons pas ce problème d'initialisation d'amer. En fait, dire que le point d'intérêt est fixé sur un amer plan résout le problème d'initialisation. Bien

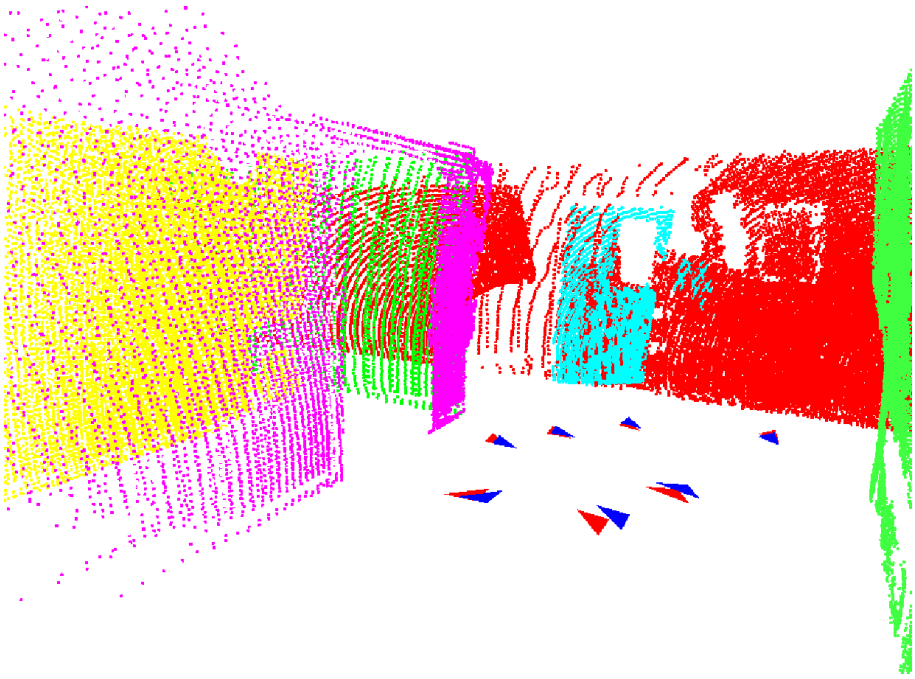


FIG. 5.8 – Résultats expérimentaux de l'algorithme SLAM avec amers plans : les points de chaque plan ont la même couleur. Les positions du robot prédites par l'odométrie sont en rouge, et celles estimées par l'algorithme SLAM sont en bleu

FIG. 5.9 – Image de la scène vue par une caméra.

que cette idée est un peu restrictive, c'est-à-dire, est limité aux points d'intérêt qui se trouvent sur des facettes planes, leur existence en milieu intérieur justifie ce choix. Ainsi, pour initialiser un amer visuel (point d'intérêt), nous le considérons comme un point 2D dans le repère local lié à l'amer plan qui le porte. Nous avons commencé à explorer cette piste, mais beaucoup de travail reste à faire. De plus, les différentes catégories de points d'intérêt pourraient être utilisés : Harris, SIFT, etc.

D'autres perspectives envisageables sont bien évidemment les améliorations au cadre classique du SLAM par filtrage de Kalman, en particulier un découpage en sous-cartes et une réduction des erreurs de linéarisation. Les techniques de FastSLAM figurent également parmi les perspectives envisageables. En fait, dans le FastSLAM, les positions successives du robot seraient modélisées par des particules et les paramètres définissant un plan pourraient être traitées via des filtres de Kalman individuels.

Chapitre 6

Conclusion et Perspectives

Conclusion

Dans les travaux développés au cours de cette thèse, nous nous sommes intéressés à la Modélisation 3D d'un environnement intérieur. Nous avons commencé par définir la problématique de la modélisation et son importance pour l'autonomie du robot mobile. Ensuite, nous avons comparé les différents types de représentations 2D afin de caractériser leurs avantages et limitations pour les applications robotiques, en particulier dans des environnements intérieurs. Nous avons ensuite présenté l'intérêt et la nécessité de construire un modèle 3D. Une étude des différents travaux effectués dans la modélisation 3D nous a permis de clarifier deux points : Le premier est le type de carte à construire, qui est une carte métrique hétérogène incluant des facettes planes, des segments 3D et des points d'intérêt. Le deuxième est de mettre l'accent sur l'importance des capteurs utilisés. En fait, il ressort de cette étude, la nécessité d'avoir un système sensoriel qui donne à la fois des données métriques précises et des informations sur l'apparence des scènes perçues (données photométriques ou chromatiques). L'importance de ce constat nous a encouragé à exploiter dans un premier temps, un capteur stéréoscopique qui donne ces deux types

d'informations.

En conséquence, nous avons commencé par exploiter l'algorithme de stéréovision passive disponible sur nos robots ; il s'est révélé insuffisant. Nous avons donc étudié la stéréovision (chapitre 3), méthode qui vise à estimer la position des points dans la scène à partir de deux images de la même scène prises sous différents angles de vue. Nous avons exposé les différentes techniques pour la mise en correspondance stéréoscopique, en particulier les méthodes globales fondées sur la coupure de graphe (*graph cuts*). L'approche de coupure de graphe transforme le problème de la mise en correspondance stéréoscopique en un problème de minimisation d'une fonction d'énergie globale. Le minimum de cette fonction est trouvé par une coupure minimale dans un graphe. Nous avons implémenté en premier temps un algorithme de coupure de graphe inspiré des travaux de Roy et Cox [Roy et Cox, 1998]. Ensuite, pour limiter la complexité de cette approche, nous avons proposé une autre formalisation du problème, qui passe par la construction d'un graphe réduit. Notre algorithme combine une méthode locale et une méthode globale fondée sur la coupure du graphe réduit. Notre approche nous a permis de réaliser deux avancées : améliorer sensiblement la qualité des images de disparité obtenues uniquement par une méthode locale et éviter l'explosion combinatoire des méthodes de coupure de graphe exécutées sans élagage préalable du graphe. Bien que nous n'ayons pas pu utiliser ces résultats dans la suite de notre travail, cela reste une contribution exploitable dans d'autres applications pour lesquelles les contraintes temps réel sont moins critiques (par exemple, le contrôle non destructif par stéréovision passive).

Nous avons ensuite présenté en chapitre 4, les méthodes que nous proposons pour la perception et la fusion de données multi-capteurs. En fait, pour percevoir l'environnement, notre robot est équipé d'un télémètre laser pivotant autour d'un axe horizontal et d'une caméra. Nous avons dans un premier temps calibré les deux capteurs afin de pouvoir exprimer leurs données dans un même repère. En outre, nous avons implémenté un algorithme de segmentation de l'image de profondeur provenant du scanner laser 3D. Le but de cette segmentation est l'extraction des facettes planes de la scène. Puis, nous avons présenté un algorithme de fusion de données laser et caméra afin d'extraire les segments 3D. Enfin, nous traitons le problème du placage de la texture sur les facettes planes. Cela nécessite le calcul de l'homographie qui relie une image virtuelle de la facette plane avec l'image provenant de la caméra.

Dans le chapitre 5, nous avons détaillé la chaîne algorithmique permettant de construire de manière incrémentale une carte hétérogène, par l'algorithme de Cartographie et Localisation Simultanées fondé sur le filtre de Kalman étendu (EKF-SLAM). Nous avons implémenté cette algorithme pour les amers plans 3D, tandis que la finalisation de la construction de la carte hétérogène reste à achever dans les travaux futurs.

Les différents mécanismes développés sont illustrés et validés par des résultats expérimentaux. Plusieurs séries d'acquisitions de données réelles ont permis de mettre en évidence le besoin d'un modèle hétérogène texturé. En effet, après une grande boucle, et à cause de l'accumulation des incertitudes, l'association de données devient une tâche difficile, voire impossible, si l'on se base seulement sur les propriétés géométriques des amers plans et sur les tests probabilistes (distance de Mahalanobis) pour les apparier. Les informations d'apparence jouent un rôle considérable pour robustifier l'appariement des amers. En outre, la structuration et la richesse du modèle fusionné permettent, d'une part, d'améliorer la robustesse de la phase d'association de données, et d'autre part, de fournir des informations utiles pour l'interaction avec un opérateur humain ou pour les raisonnements spatiaux de plus haut niveau.

Enfin, l'utilisation de plusieurs capteurs nous a permis de surmonter les lacunes intrinsèques de chacun d'eux. Cette étude nous a donné les moyens de présenter la fusion de données, si elle est bien faite, comme un outil puissant pour les applications robotiques.

Perspectives

Les travaux développés ont permis d'ouvrir un certain nombre de perspectives intéressantes. Nous proposons quelques directions envisageables pour les travaux futurs.

- En premier lieu, plusieurs améliorations de l'algorithme stéréovision par coupure de graphe réduit sont possibles. D'une part, le code a été écrit avec l'esprit d'aboutir à des résultats, et non en vue d'optimiser le temps d'exécution, ce qui pourrait être réalisé pour accélérer l'algorithme. D'autre part, nous produisons une image de disparité entière, alors qu'en principe, le graphe réduit est capable de gérer des disparités réelles (interpolation sous-pixellique de la disparité), la mise en oeuvre de cette technique reste à faire pour améliorer la précision de l'image de disparité.

- Ensuite, achever la construction de la carte hétérogène sera une première priorité. Ajouter des amers objets (des objets reconnaissables) serait un but à long terme. De plus, des campagnes d'acquisitions et des séries de validations expérimentales seraient indispensables afin de rendre les algorithmes proposés plus fiables.

- Par ailleurs, pour obtenir des modèles opérationnels et performants de l'environnement, qui répondent à des spécifications de besoin précises, il nous paraît indispensable de pouvoir évaluer correctement la qualité des représentations construites, à divers stades de leur développement. Il s'agit d'élaborer des méthodologies d'évaluation quantitative.

- De plus, améliorer le cadre classique de SLAM par filtre de Kalman, en particulier un découpage en sous-cartes et une réduction des erreurs de linéarisation. Les techniques de FastSLAM figurent également parmi les perspectives envisageables. En fait, dans le FastSLAM, les positions successives du robot seraient modélisées par des particules et les paramètres définissant un amer plan pourraient être estimées via des filtres de Kalman individuels.

- En outre, passer de la représentation géométrique vers une représentation plus symbolique serait aussi envisageable, via une coopération lors de la phase d'apprentissage, avec un opérateur humain. Seul l'homme peut annoter un modèle avec des symboles qui lui sont connus (par exemple, nom de lieux : cuisine... ou d'objets : tasse, bouteille...). Avec une telle représentation symbolique, le robot peut mettre en oeuvre un haut niveau d'interaction avec l'homme.

- Enfin, les techniques proposées dans le cadre de cette thèse ont été développées pour les environnements statiques. Toutefois, dans certaines applications concrètes, il serait également pertinent de s'affranchir de cette hypothèse et de modéliser des environnements évolutifs

(certains objets peuvent y être déplacés, comme des chaises) ou dynamiques (d'autres objets mobiles s'y déplacent, comme des humains ou d'autres robots).

Bibliographie

Abuhadrous, I., Ammoun, S., Nashashibi, F., Goulette, F. et Laurgeau, C. (2004). Digitizing and 3d modelling of urban environments using vehicle-borne laser scanner system. *In Proceedings of the IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*. 2.5.2

Alami, R., Chatila, R., Fleury, S., Ghallab, M. et Ingrand, F. (1998). An architecture for autonomy. *International Journal of Robotics Research, Special Issue on Integrated Architectures for Robot Control and Programming*, 17(4):315–337. 4.3

Angeli, A., Filliat, D., Doncieux, S. et Meyer, J.-A. (2008). Real-time visual loop-closure detection. *In Proceedings of the International Conference on Robotics and Automation (ICRA)*. 5.4.2

Ayache, N. et Faugeras, O. (1988). Maintaining representations of the environment of a mobile robot. *In The fourth international symposium*, pages 337–350. MIT Press. 2.3

Baba, A. (2007). *Cartographie de l'Environnement et Suivi Simultané de Cibles Dynamiques par Un Robot Mobile*. Thèse de doctorat, Université de Toulouse - Paul Sabatier. (document), 2.4.3, 2.5

Bar-Shalom, Y., Li, X. R. et Kirubarajan, T. (2001). *Estimation with Applications to Tracking and Navigation*. Wiley-Interscience, 1 édition. 5.2.5

Belhumeur, P. N. (1996). A bayesian approach to binocular stereopsis. *International Journal of Computer Vision*, 19(3):237–260. 3.3.2

Bellman, R. (1957). *Dynamic programming*. Princeton University Press. 3.3.2

Biber, P., Andreasson, H., Duckett, T. et Schilling, A. (2004). 3d modeling of indoor environments by a mobile robot with a laser scanner and panoramic camera. *In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2.5.2

Biber, P., Fleck, S. et Duckett, T. (2005). 3d modeling of indoor environments for a robotic security guard. *In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 2.5.2

Bobick, A. F. et Intille, S. S. (1999). Large occlusion stereo. *International Journal of Computer Vision (IJCV)*, 33(3):181–200. 3.3.2

Bosse, M., Newman, P., Leonard, J., Soika, M., Feiten, W. et Teller, S. (2003). An atlas framework for scalable mapping. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1899–1906. 2.4.5

Bouguet, J.-Y. (2007). Camera calibration toolbox for matlab. (consulté en avril 2008). 4.2, 4.4

Boykov, Y. et Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1124–1137. 3.4.8

Boykov, Y., Veksler, O. et Zabih, R. (1998). Markov random fields with efficient approximations. *In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 648–655. IEEE Computer Society. 3.3.2, 3.3.2

Boykov, Y., Veksler, O. et Zabih, R. (1999). Fast approximate energy minimization via graph cuts. *In Proceedings of International Conference on Computer Vision (ICCV)*, volume 1, pages 377–384. 3.3.2

Castellanos, J., Montiel, J., Neira, J. et Tardos, J. (1999). The spmap : a probabilistic framework for simultaneous localizationand map building. *In IEEE Transactions on Robotics and Automation*, pages 948–952. 5.2.2

Castellanos, J. A., Montiel, J. M. M., Neira, J. et Tardós, J. D. (2000). Sensor influence in the performance of simultaneous mobile robot localization and map building. *In The Sixth International Symposium on Experimental Robotics VI*, pages 287–296. Springer-Verlag. 2.3

Chambon, S. (2005). *Mise en correspondance stéréoscopique d'images couleur en présence d'occultations.* Thèse de doctorat, Université Paul Sabatier, Toulouse, France. 3.3

Chatila, R. et Laumond, J. P. (1985). Position referencing and consistent world modeling for mobile robots. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 138–145. 2.3

Chen, Q. et Medioni, G. (1999). A volumetric stereo matching method : Application to image-based modeling. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).* 3.3.1

Chong, K. S. et Kleeman, L. (1997). Large scale sonarray mapping using multiple connected local maps. *In In International Conference on Field and Service Robotics*, pages 538–545, ANU, Canberra, Australia. 2.4.5

Cormen, T. H., Leiserson, C. E., Rivest, R. L. et Stein, C. (2001). *Introduction to Algorithms, Second Edition.* The MIT Press. 3.4.8, 3.4.8

Cox, I. J. (1992). Stereo without disparity gradient smoothing : a bayesian sensor fusion solution. *In In proceedings of British Machine Vision Conference (BMVC)*, pages 337–346. 3.3.2

Cox, I. J., Hingorani, S. L., Rao, S. B. et Maggs, B. M. (1996). A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3):542–567. 3.3.1, 3.3.2

Csorba, M. (1997). *Simultaneous Localisation and Map Building.* Thèse de doctorat, University of Oxford. 5.2.8

Dahlhaus, E., Johnson, D. S., Papadimitriou, C. H., Seymour, P. D. et Yannakakis, M. (1994). The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864–894. 3.3.2

Davison, A. (2003). Real-time simultaneous localisation and mapping with a single camera. *In IEEE International Conference on Computer Vision (ICCV)*, Nice, France. 5.4.2

Davison, A. J., Reid, I. D., Molton, N. D. et Stasse, O. (2007). MonoSLAM : Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067. 2.5.2

Diebel, J., Reuterswärd, K., Thrun, S., Davis, J. et Gupta, R. (2004). Simultaneous localization and mapping with active stereo vision. *In Proceedings of the IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, pages 3436–2443, Sendai, Japan. 2.5.2

Dima, C. (2002). Using multiple disparity hypotheses for improved indoor stereo. *In International Conference on Robotics and Automation.* IEEE. 3.3.1

Dissanayake, G., Newman, P., Durrant-Whyte, H. F., Clark, S. et Csorba, M. (2000). An experimental and theoretical investigation into simultaneous localisation and map building. *In The Sixth International Symposium on Experimental Robotics VI*, pages 265–274. Springer-Verlag. 2.3, 5.2.7

Duckett, T. et Saffiotti, A. (2000). Building globally consistent gridmaps from topologies. *In Proceedings of the International Proc. IFAC Symposium on Robot Control*, pages 357–361, Wien, Austria. 2.4.5

Dufourd, D. (2005). *Des Cartes Combinatoires Pour La Construction Automatique De Modèles D'Environnement Par Un Robot Mobile.* Thèse de doctorat, Institut National Polytechnique de Toulouse. 2.4.2

Durrant-Whyte, H. et Bailey, T. (2006). Simultaneous Localization and Mapping (SLAM) : Part I & II. *IEEE Robotics & Automation Magazine.* 2.3

Fabrizi, E. et Saffiotti, A. (2002). Augmenting topology-based maps with geometric information. *Robotics and Autonomous Systems*, 40(2–3):91–97. 2.4.2

Faugeras, O. (1993). *Three-Dimensional Computer Vision : a Geometric Viewpoint.* MIT press. 3.2.3, 3.2.4, 3.3, 4.4

Faugeras, O., Luong, Q.-T. et Papadopoulou, T. (2001). *The Geometry of Multiple Images : The Laws That Govern The Formation of Images of A Scene and Some of Their Applications.* MIT Press. 4.8.1

Faugeras, O., Vieville, T. et al. (1993). Real-time correlation-based stereo : algorithm, implementations and applications. Rapport technique RR-2013, INRIA-Sophia Antipolis. 3.3

Feder, H. J. S., Leonard, J. J. et Smith, C. M. (1999). Adaptive mobile robot navigation and mapping. *The International Journal of Robotics Research*, 18(7):650–668. 2.3

Ford, L. et Fulkerson, D. (1962). *Flows in Networks*. Princeton University Press. 3.4.7, 3.4.8, 3.4.8, 3.5.3

Früh, C. et Zakhor, A. (2004). An automated method for large-scale, ground-based city model acquisition. *International Journal of Computer Vision*, 60(1):5–24. 2.5.2

Gasós, J. a. (1999). Integrating fuzzy geometric maps and topological maps for robot navigation. *In roc. of the 3rd International Symposium on Soft Computing*. 2.4.5

Gil, A., Reinoso, O., Payá, L., Ballesta, M. et Pedrero, J. M. (2007). Managing data association in visual slam using sift features. *International Journal of Factory Automation, Robotics and Soft Computing*, pages 179–184. 5.4.2

Goldberg, A. V. (1985). A new max-flow algorithm. Rapport technique MIT/LCS/TM-291, Massachusetts Institute of Technology, Cambridge, MA. 3.4.8, 3.6

Goldberg, A. V. et Tarjan, R. E. (1988). A new approach to the maximum-flow problem. *Journal of the Association for Computing Machinery (JACM)*, 35(4):921–940. 3.4.8

Graham, R. (1972). An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133. 4.8.2

Grandjean, P. et Lasserre, P. (1995). Stereo Vision Improvments. *In IEEE International Conference on Advanced Robotics*, Barcelona (Spain). 3.3

Guan, C., Hassebrook, L. G., Lau, D. L. et Yalla, V. (2007). Near-infrared composite pattern projection for continuous motion hand-computer interaction. *Journal of Visual Communication and Image Representation*, 18(2):141–150. 2.6.1

Hähnel, D., Burgard, W. et Thrun, S. (2003). Learning compact 3d models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*. 2.5.2, 4.6.1

Hall, D. L. et Llinas, J., éditeurs (2001). *Handbook of Multisensor Data Fusion*. CRC Press LLC. 4.5

Harati, A., Gachter, S. et Siegwart, R. (2007). Fast range image segmentation for indoor 3d-slam. *In The 6th IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*. 4.6.1

Harris, C. et Stevens, M. (1988). A combined corner and edge detector. *In Proc. of 4th Alvey Vision Conference*, pages 147–151. 3.3

Hayet, J., Lerasle, F. et Devy, M. (2003). Visual landmarks detection and recognition for mobile robot navigation. *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 313–318, Madison (USA). 5.4.2

Heckbert, P. S. et Garland, M. (1997). Survey of polygonal surface simplification algorithms. Rapport technique, Carnegie-Mellon Univ. 4.6.1

Hirschmüller, H. (2001). Improvements in real-time correlation-based stereo vision. *In Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV)*. 3.3.1

Ho, K. et Newman, P. (2007). Detecting loop closure with scene sequences. *International Journal of Computer Vision and International Journal of Robotics Research. Joint issue on computer vision and robotics*, 74(3):261–286. 5.4.2

Horaud, R. et Monga, O. (1995). *Vision par ordinateur : Outils fondamentaux.* Editions Hermès, 2 édition. 3.2.1, 3.2.3

Horn, B. (1986). *Robot Vision.* MIT Press. 3.3.1

Horn, J. et Schmidt, G. (1995). Continuous localization for long-range indoor navigation of mobile robots. *In IEEE International Conference on Robotics and Automation*, pages 387–394. 4.6.1

Illingworth, J. et Kittler, J. (1988). A survey of the hough transform. *Computer Vision, Graphics, and Image Processing*, 44(1):87–116. 4.6.1, 4.6.4

Ingrand, F. (2003). Architectures logicielles pour la robotique autonome. *In Journées Nationales de Rechercher en Robotique (JNRR).* 4.3

Iocchi, L., Konolige, K. et Bajracharya, M. (2000). Visually realistic mapping of a planar environment with stereo. *In Proceedings of the International Symposium on Experimental Robotics (ISER)*, pages 521–532. 2.5.2

Ishikawa, H. (2000). *Global optimization using embedded graphs.* Thèse de doctorat, New York University. Adviser : Davi Geiger. 3.5.3

Ishikawa, H. et Geiger, D. (1998). Occlusions, discontinuities, and epipolar lines in stereo. *In Proceedings of the 5th European Conference on Computer Vision (ECCV)*, pages 232–248. Springer-Verlag. 3.3.2, 3.3.2

Jo, S., Kwon, Y.-M. et Ko, H. (2006a). *Advances in Artificial Reality and Tele-Existence*, chapitre Indoor Environment Modeling for Interactive VR - Based Robot Security Service. Lecture Notes in Computer Science. Springer Berlin / Heidelberg. 2.5.2

Jo, S., Shahab, Q., Kwon, Y.-M. et Ahn, S. C. (2006b). Indoor modeling for interactive robot service. *In International Joint Conference SICE-ICASE*, pages 3531–3536, Bexco, Busan, Korea. 2.5.2

Julesz, B. (1962). Towards the automation of binocular depth perception. *In IFIP Congress*, pages 439–444. 3.3

Jung, I. K. (2004). *Simultaneous localization and mapping in 3D environments with stereovision.* Thèse de doctorat, Institut National Polytechnique de Toulouse, France. 2.3, 2.5.2

Jung, I. K. et Lacroix, S. (2001). A robust interest point matching algorithm. *In Proceedings of International Conference on Computer Vision (ICCV).* 5.5.2

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, 82:35–45. 4.6.3

Karzanov, A. V. (1974). Determining the maximal flow in a network by the method of preflows. *Soviet Mathematics Doklady*, 15:434–437. 3.4.8

Kohlhepp, P., Pozzo, P., Walther, M. et Dillmann, R. (2004). Sequential 3d-slam for mobile action planning. *In Proceedings of the IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, pages 722–729. 4.6.1

KOLMOGOROV, V. et ZABIH, R. (2001). Computing visual correspondence with occlusions using graph cuts. *In Proceedings of the 8th International Conference on Computer Vision (ICCV)*, pages 508–515. 3.3.2

KOLMOGOROV, V. et ZABIH, R. (2002a). Multi-camera scene reconstruction via graph cuts. *In Proceedings of the 7th European Conference on Computer Vision (ECCV)*, pages 82–96. Springer-Verlag. 3.3.2

KOLMOGOROV, V. et ZABIH, R. (2002b). What energy functions can be minimized via graph cuts ? *In Proceedings of the 7th European Conference on Computer Vision (ECCV)*, pages 65–81. Springer-Verlag. 3.3.2

LANGE, R. et SEITZ, P. (2001). Solid-state time-of-flight range camera. *IEEE Journal of Quantum Electronics*, 37(3):390–397. 2.6.3

LEMAIRE, T., LACROIX, S. et SOLA, J. (2005). A practical 3D Bearings Only SLAM algorithm. *In Proceedings of the IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*. 5.4.2, 5.7

LEONARD, J. J. et DURRANT-WHYTE, H. F. (1991). Simultaneous map building and localization for an autonomous mobile robot. *In IEEE/RSJ International Workshop on Intelligent Robots and Systems*, volume 3, pages 1442–1447. 2.3, 5.2.2

LIU, Y., EMERY, R., CHAKRABARTI, D., BURGARD, W. et THRUN, S. (2001). Using em to learn 3d models of indoor environments with mobile robots. *In International Conference on Machine Learning (ICML)*. 2.5.2, 4.6.1

MARR, D. et POGGIO, T. (1977). A theory of human stereo vision. Rapport technique AIM-451, Massachusetts Institute of Technology, Cambridge, MA, USA. 3.3.1

MATAS, J., GALAMBOS, C. et KITTLER, J. (1998). Progressive probabilistic hough transform. *In Proceedings of British Machine Vision Conference BMVC98*. 4.6.4

MAYBECK, P. S. (1982). *Stochastic Models, Estimation, and Control*, volume 1. Academic Press. 5.2.7

MILLER, K. et LESKIW, D. (1987). *An Introduction to Kalman Filtering with Applications*. Krieger Publishing Company, 1 édition. 5.2.7

MONTEMERLO, M., THRUN, S., KOLLER, D. et WEGBREIT, B. (2002). FastSLAM : A factored solution to the simultaneous localization and mapping problem. *In Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada. 2.3

MONTEMERLO, M., THRUN, S., KOLLER, D. et WEGBREIT, B. (2003). FastSLAM 2.0 : An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico. 2.3

NASHASHIBI, F. et DEVY, M. (1993). 3d incremental modeling and robot localization in a structured environment using a laser range finder. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2.5.2, 5.5.2

Neira, J. et Tardós, J. (2001). Data association in stochastic mapping using the joint compatibility test. *In IEEE Transactions on Robotics and Automation*. 5.4, 5.4.1

Newman, P. (1999). *On The Structure and Solution of the Simultaneous Localisation and Map Building Problem*. Thèse de doctorat, University of Sydney, Australian Center For Field Robotics. 2.3

Nieto, J. I., Guivant, J. E. et Nebot, E. M. (2004). The hybrid metric maps (hymms) : A novel map representation for denseslam. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 391–396. 2.4.5

Nüchter, A., Surmann, H. et Hertzberg, J. (2003). Automatic model refinement for 3d reconstruction with mobile robots. *In Proceedings of the International Conference on 3-D Digital Imaging and Modeling (3DIM)*. 2.5.2

Ohta, Y. et Kanade, T. (1985). Stereo by intra- and inter-scanline search using dynamic programming. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 139–154. 3.3.2

Posner, I., Schroeter, D. et Newman, P. (2007). Describing composite urban workspaces. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Rome. 5.4.2

Rabaud, C. (2005). *Une nouvelle approche de mise en correspondance stereoscopique dense par methodes possibilistes*. Thèse de doctorat, Université Montpellier II. 3.3

Rencken, W. (1993). Concurrent localisation and map building for mobile robots using ultrasonic sensors. *In IEEE/RSJ International Workshop on Intelligent Robots and Systems*, volume 3, pages 2192–2197. 2.3

Roy, S. (1999). Stereo without epipolar lines : A maximum-flow formulation. *International Journal of Computer Vision*, 34(2-3):147–161. 3.3.2, 3.6

Roy, S. et Cox, I. (1998). A maximum-flow formulation of the n-camera stereo correspondence problem. *In Proceedings of the International Conference on Computer Vision (ICCV)*, pages 492–499. (document), 2.7, 3.3.2, 3.5.2, 3.5.3, 3.20, 6

Sára, R. (2002). Finding the largest unambiguous component of stereo matching. *In Proceedings of the 7th European Conference on Computer Vision (ECCV)*, pages 900–914. Springer-Verlag. 3.3.1

Scharstein, D. et Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47. 3.3, 3.6

Sequeira, V., Ng, K., Wolfart, E., Goncalves, J. et Hogg, D. (1999). Automated reconstruction of 3d models from real environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(1):1–22. 4.6.1

Siek, J. G., Lee, L.-Q. et Lumsdaine, A. (2001). *Boost Graph Library, The : User Guide and Reference Manual*. Addison-Wesley Professional. 3.6

Silveira, G., Malis, E. et Rives, P. (2006). Real-time robust detection of planar regions in a pair of images. *In Proc. IEEE/RSJ International Conference on Intelligent Robots Systems, Beijing, China*. 2.5.2

Simhon, S. et Dudek, G. (1998). A global topological map formed by local metric maps. *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1708–1714, Victoria, BC, Canada. 2.4.5

Smith, R., Self, M. et Cheeseman, P. (1988). A stochastic map for uncertain spatial relationships. *In The fourth international symposium*, pages 467–474. MIT Press. 2.3

Smith, R., Self, M. et Cheeseman, P. (1990). Estimating uncertain spatial relationships in robotics. *Autonomous robot vehicles*, pages 167–193. 2.3, 5.2.6

Sola, J., Monin, A., Devy, M. et Lemaire, T. (2005). Undelayed initialization in bearing only slam. *In Proceedings of the IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, pages 2751–2756. 2.3, 2.5.2, 5.4.2, 5.7

Takezawa, A., Herath, D. C. et Dissanayake, G. (2004). Slam in indoor environments with stereo vision. *In Proceedings of the IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*. 2.3, 2.5.2

Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71. 2.4.5

Thrun, S. (2002). Robotic mapping : A survey. *In* Lakemeyer, G. et Nebel, B., éditeurs : *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann. 2.4.1, 2.4.2, 2.5.2

Thrun, S., Burgard, W. et Fox, D. (1998). A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31(1-3):29–53. 2.3

Thrun, S., Fox, D. et Burgard, W. (2000). A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA. 2.5.2

Thrun, S., Martin, C., Liu, Y., Hähnel, D., Emery Montemerlo, R., Deepayan, C. et Burgard, W. (2003). A real-time expectation maximization algorithm for acquiring multiplanar maps of indoor environments with mobile robots. *IEEE Transactions on Robotics and Automation*, 20(3):433–442. 2.5.2, 2.6.2

Tomatis, N. (2001). *Hybrid, Metric-Topological, Mobile Robot Navigation*. Thèse de doctorat, Ecole Polytechnique Fédérale de Lausanne, Switzerland. 2.4.5

Unnikrishnan, R. et Hebert, M. (2005). Fast extrinsic calibration of a laser rangefinder to a camera. Rapport technique CMU-RI-TR-05-09, Robotics Institute, Carnegie Mellon University. 4.4, 4.4

Unser, M. (1995). Texture classification and segmentation using wavelet frames. *IEEE Transactions on Image Processing*, 4(11):1549–1560. 5.4.3

Veksler, O. (1999). *Efficient graph-based energy minimization methods in computer vision*. Thèse de doctorat, Cornell University. Adviser : Ramin Zabih. 3.3.2, 3.5.2

Weingarten, J. (2006). *Feature-based 3D SLAM*. Thèse de doctorat, École Polytechnique Fédérale de Lausanne. 2.5.2, 2.6.2, 4.6.1

Williams, S. (2001). *Efficient Solutions to Autonomous Mapping and Navigation Problems*. Thèse de doctorat, The University of Sydney. 5.2.4

Zhang, Z. (1999). Flexible camera calibration by viewing a plane from unknown orientations. *In Proceedings of the International Conference on Computer Vision (ICCV)*, pages 666–673. 4.4

Zureiki, A., Devy, M. et Chatila, R. (2007a). Mise en correspondance en stéréovision, fondée sur la coupure de graphe. *In Onzième congrès francophone des jeunes chercheurs en vision par ordinateur (ORASIS)*, Obernai, France. 3.5.4

Zureiki, A., Devy, M. et Chatila, R. (2007b). Stereo matching using reduced-graph cuts. *In IEEE International Conference on Image Processing (ICIP)*, San Antonio, Texas (USA). 3.5.4

Liste de mes publications

Chapitre de Bouquin :

1. Ayman Zureiki, Michel Devy et Raja Chatila. *Stereo Matching and Graph Cuts*. chapter of the book *Stereo Vision*, I-Tech Education and Publishing, Vienna, Austria, to appear in november 2008.

Conférences Internationales :

1. Ayman Zureiki, Michel Devy et Raja Chatila. *Stereo matching using reduced-graph cuts*. In IEEE International Conference on Image Processing (ICIP), San Antonio, Texas (USA), septembre 2007.

2. Ayman Zureiki, Michel Devy et Raja Chatila. *SLAM and Multi-feature Map by Fusing 3D Laser and Camera Data*. In Proceedings of the 5th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Funchal, Madeira (Portugal), mai 2008.

3. Ayman Zureiki et Michel Devy. *SLAM and data fusion from visual landmarks and 3D planes*. In Proceedings of the 17th IFAC World Congress, Seoul, Korea, juillet 2008.

4. Ayman Zureiki et Michel Devy. *Appearance-based Data Association for 3D and Multisensory SLAM in Structured Environment*. In The International Conference on Information & Communication Technologies : from Theory to Applications (ICTTA), avril 2008.

Conférences Nationales :

1. Ayman Zureiki et Raja Chatila. *Modélisation tridimensionnelle de l'environnement pour un robot mobile*. In 7ème Congrès Ecole Doctoral Systèmes (EDSYS), Tarbes, France, 11 mai 2006.

2. Ayman Zureiki, Michel Devy et Raja Chatila. *Mise en correspondance en stéréovision, fondée sur la coupure de graphe*. In Onzième congrès francophone des jeunes chercheurs en vision par ordinateur (ORASIS), Obernai, France, 4-8 juin 2007.

Abstract

Title :

Multi-Sensor Data Fusion for Incremental Construction of a Textured Tridimensional Model of Indoor Environment by a Mobile Robot

This thesis examines the problem of 3D Modelling of indoor environment (supposed unknown) by a mobile robot. Our main contribution consists in constructing a dense geometrical model represented as a heterogeneous map containing textured planar landmarks, 3D lines and interest points. In order to achieve this mission, we must fuse geometrical and photometrical data. Hence, we began by improving the stereo vision algorithm. We proposed a new approach that transform the stereo matching into a problem of minimization of a global energy function. The minimum of this function is found by finding a minimum cut in a graph. The most significant contribution on stereovision is to propose, for the first time, the construction of a reduced graph that allows to accelerate the stereo correspondence using graph cuts and to provide better results than the local methods.

Nevertheless, this method remains non applicable in mobile robotics. Also, to perceive the environment, the robot is equipped by a laser scanner rotating around a horizontal axis and by a camera. We proposed an algorithmic chain allowing to construct incrementally the heterogeneous map, using the algorithm of Simultaneous Localization and Mapping based on the extended Kalman filter (EKF-SLAM). Mapping the texture on the planar landmarks makes the model more realistic for a user, and makes more robust the phase of data association, which is essential to guarantee the map consistency. The different developed mechanisms were illustrated by experimental results.

Keywords : Data Fusion, Simultaneous Localization and Mapping (SLAM), 3D Modelling, Mobile Robot, Stereovision, Graph Cuts.

Résumé

Ce travail traite la problématique de la Modélisation 3D d'un environnement intérieur supposé inconnu par un robot mobile. Notre principale contribution concerne la construction d'un modèle géométrique dense représenté par une carte hétérogène qui combine des amers plans texturés, des lignes 3D et des points d'intérêt. Afin de réaliser cette tâche, nous devons fusionner des données géométriques et photométriques. Pour cela, nous avons d'abord amélioré la stéréovision dense, en proposant une approche qui transforme le problème de la mise en correspondance stéréoscopique en un problème de minimisation d'une fonction d'énergie globale. Le minimum de cette fonction est trouvé par une coupure minimale dans un graphe. Notre contribution réside dans la construction d'un graphe réduit qui a permis d'accélérer considérablement cette méthode globale de l'appariement stéréoscopique et d'obtenir de meilleurs résultats que les méthodes locales.

Néanmoins, cette méthode reste non applicable en robotique mobile. Aussi, pour percevoir l'environnement, le robot est équipé d'un télémètre laser pivotant autour d'un axe horizontal et d'une caméra. Nous proposons une chaîne algorithmique permettant de construire de manière incrémentale une carte hétérogène, par l'algorithme de Cartographie et Localisation Simultanées basé sur le filtre de Kalman étendu (EKF-SLAM). Le placage de la texture sur les facettes planes rend le modèle plus réaliste pour un opérateur ; il a permis aussi de solidifier l'étape d'association de données, une étape essentielle pour garantir la cohérence de la carte. Les différents mécanismes développés sont illustrés et validés par des résultats expérimentaux.

Mots-clés : Fusion de Données, Cartographie et Localisation Simultanées (SLAM), Modélisation 3D, Robot Mobile, Stéréovision, Coupure de Graphe.