NNT : 2016SACLE054

# Thèse de doctorat
# de
# l'Université Mohammed V de Rabat
# et de
# l'Université Paris-Saclay
# préparée a
# L'Université d'Evry Val d'Essonne

Ecole doctorale n°580
Sciences et Technologies de l'Information et de la Communication
Spécialité de doctorat : Informatique

par

**M. Hakim ELCHAOUI ELGHOR**

## Modélisation Planaire pour un RGB-D SLAM: Localisation éparse et Cartographie réduite

Thèse présentée et soutenue à "Rabat", le 06 Décembre 2016.

Composition du Jury :

| | |
|---|---|
| M.Mourad EL BELKACEMI, Professeur, Faculté des Sciences Rabat. Maroc | Président |
| M.EL HASSANE IBN EL HAJ, Professeur, INPT Rabat. Maroc | Rapporteur |
| M.Michel DEVY, Directeur de Recherche, LAAS-CNRS. France | Rapporteur |
| M.David ROUSSEL, MCF, ENSIIE d'Evry. France | Examinateur |
| M.El Houssine BOUYAKHF, Professeur, Faculté des Sciences Rabat. Maroc | co-Directeur |
| M.Fakhreddine ABABSA, MCF-HDR, Université d'Evry Val d'Essonne. France | co-Directeur |

# Dédicaces

A ma mère, A ma mère, A ma mère

A mon père

A mon frère

A ma soeur

A la mémoire de mon grand-père

# Préface

Ce mémoire présente le travail de thèse effectué dans le cadre d'une cotutelle internationale entre l'Université Paris-Saclay, France et l'Université Mohammed V de Rabat, Maroc.

Les travaux de recherche présentés ici ont été réalisés en collaboration, sous la direction de Messieurs Fakhreddine Ababsa et El-Houssine Bouyakhf, dans deux laboratoires de recherche : au sein de l'équipe IRA$^2$ (Interaction, Réalité Augmentée et Robotique Ambiante) du laboratoire IBISC (Informatique, Biologie Intégrative et Systèmes Complexes) attaché à l'université d'Evry Val d'Essonne, et au laboratoire LIMIARF (Informatique, Mathématiques appliquées, Intelligence Artificielle et Reconnaissance des Formes) attaché à la Faculté des Sciences de Rabat.

# Remerciement

Le bon déroulement des travaux de recherche réalisés dans cette thèse et la rédaction de ce mémoire n'auraient pu avoir lieu sans le concours de nombreuses personnes que je tiens à remercier :

En tout premier lieu j'adresse ma vive gratitude à mes directeurs de thèse, monsieur El-Houssine BOUYAKHF et monsieur Fakhreddine ABABSA, pour tout l'engagement, la gentillesse, l'orientation et la compréhension qu'ils m'ont prodigué tout au long de ces années. Je leur suis reconnaissant pour les conseils et remarques judicieuses qui ont grandement aidé à améliorer la qualité de cette thèse. C'était un réel plaisir de travailler avec des personnes très motivées, intelligentes, enthousiastes et passionnées par leur travail.

Je présente mes remerciements particuliers à monsieur David ROUSSEL, pour son excellent encadrement pendant toute la durée de cette thèse. Je ne peux que lui être reconnaissant et je le remercie pour le temps et l'aide qu'il m'a accordé, que cela soit pour la réalisation de mes travaux ou pour la rédaction, et pour les riches discussions qui m'ont poussé à donner le meilleur de moi-même.

Je tiens aussi à remercier les rapporteurs de cette thèse : monsieur Michel DEVY et monsieur El Hassane IBN EL HAJ. C'est un honneur qu'ils m'ont fait en acceptant d'évaluer ce travail et d'être membre de jury de cette thèse. Leurs commentaires détaillés ont considérablement amélioré la qualité de cette thèse.

Mes vifs remerciements vont à monsieur Mourad EL BELKACEMI, Doyen de la Faculté des Sciences de Rabat, de m'avoir fait l'honneur de présider le jury.

Je remercie tous les collègues, passés et présents, des deux laboratoires (IBISC et LIMIARF) pour les moments passés ensemble durant toutes ces années. Je ne peux pas les citer toutes et tous faute de place, mais je leur souhaite le meilleur pour la suite.

Merci également aux collègues de l'UFRST, personnels administratif et enseignants : Sabine, Cécile, Gislin, Fred, Samia, Jean-Yves, Elho, Amine, Malik, Lotfi et Naoufel.

Il y a beaucoup de gens à associer à cette thèse et qui méritent tous des remerciements, chacun pour des raisons particulières. Ce sont des personnes qui ont partagé le meilleur comme le pire et qui m'ont soutenu inconditionnellement dans les moments de faiblesse. Je m'adresse en particulier à : Kamal, Widad, Brahim, Abdelhak, H.A Hicham et Les Salapins.

Enfin, je voudrais conclure cette page en m'adressant à ma famille qui m'a soutenu et encouragé tout au long de ces années d'études : Je ne saurai jamais vous remercier autant que je le devrais.

# Table des matières

**Introduction Générale** **13**
0.1 Contexte général . . . . . 13
0.2 Contributions de la thèse . . . . . 14
0.3 Structure de la thèse . . . . . 15

**1 Éléments de base** **17**
1.1 Conventions de notation . . . . . 18
1.2 Capteurs RGB-D . . . . . 18
1.2.1 Mesure de la profondeur . . . . . 19
1.2.2 Construction d'un nuage de points 3D . . . . . 20
1.2.3 Limitations . . . . . 21
1.3 Notions de vision par ordinateur . . . . . 23
1.3.1 Point d'intérêts, Détection et Correspondances . . . . . 23
1.3.2 Estimation robuste de la pose . . . . . 26
1.3.2.1 Transformation rigide . . . . . 27
1.3.2.2 Estimation aux moindres carrés du déplacement 28
1.3.2.3 Random Sample Consensus RANSAC . . . . . 28
1.3.2.4 Iterative Closest Point ICP . . . . . 30
1.4 Bibliothèques Open Source . . . . . 31
1.4.1 OpenCV . . . . . 31
1.4.2 PCL . . . . . 32
1.4.3 ROS . . . . . 32
1.5 Conclusion . . . . . 33

**2 Localisation et Cartographie 3D par vision** **35**
2.1 Introduction . . . . . 36
2.1.1 Évolution du SLAM visuel . . . . . 36
2.1.2 Modélisation GraphSLAM . . . . . 38

2.1.3 Représentations de l'environnement . . . . . 41
2.1.3.1 Cartes Métriques . . . . . 41
2.1.3.2 Cartes Topologiques . . . . . 42
2.1.3.3 Cartes Sémantiques . . . . . 44
2.2 Systèmes RGB-D SLAM . . . . . 45
2.2.1 RGB-D SLAM sparse . . . . . 46
2.2.2 RGB-D SLAM dense . . . . . 49
2.2.3 Évaluations et Comparaison . . . . . 51
2.2.4 Synthèse . . . . . 56
2.3 Conclusion . . . . . 59

**3 RGB-D SLAM basé-plans** **61**
3.1 Introduction . . . . . 62
3.2 Travaux connexes . . . . . 62
3.3 RGB-D SLAM basé-plans . . . . . 65
3.3.1 Détection de plans . . . . . 70
3.3.2 Points d'intérêt Planaires . . . . . 72
3.3.3 Cartes 3D basées-Plans . . . . . 74
3.4 Conclusion . . . . . 80

**4 Évaluations et Résultats** **83**
4.1 Introduction . . . . . 84
4.2 Détection de plans . . . . . 85
4.2.1 Objectifs . . . . . 85
4.2.2 Protocoles . . . . . 85
4.2.3 Résultats . . . . . 87
4.3 Benchmarks TUM . . . . . 94
4.3.1 Objectifs . . . . . 94
4.3.2 Protocoles . . . . . 95
4.3.3 Résultats . . . . . 95
4.4 Séquence Bureau . . . . . 98
4.4.1 Représentation Cartographique . . . . . 98
4.4.1.1 Carte 3D basée-points . . . . . 98
4.4.1.2 Carte 3D basée-Plans . . . . . 101
4.4.1.3 Comparaison des représentations de la carte . . . 101
4.4.2 Estimation de la pose . . . . . 103
4.4.2.1 Objectifs . . . . . 103

4.4.2.2 Protocoles . . . . . 103
4.4.2.3 Résultats . . . . . 106
4.4.3 Évaluation de la Carte . . . . . 108
4.4.3.1 Objectifs . . . . . 108
4.4.3.2 Protocoles . . . . . 109
4.4.3.3 Résultats . . . . . 109
4.5 Conclusion . . . . . 110

**Conclusion Générale** **113**

**Bibliographie** **117**

**Résumé** **129**

**Abstract** **130**

# Table des figures


1.1 Principe du calcul de la profondeur en utilisant la lumière structurée. . . . . 20
1.2 Génération d'un nuage de points 3D texturé à partir de la correspondance Couleur-Profondeur. . . . . 21

2.1 Représentation du GraphSLAM. . . . . 39
2.2 Exemple de cartes métriques. Source[27] . . . . . 43
2.3 Exemple d'une carte topologique. (*a*) Projection dans le plan. (*b*) Structure topologique correspondante. (Extrait de [25]). . . . . 44
2.4 Exemples de cartes RGB-D SLAM sparse. . . . . 52
2.5 Exemples de cartes RGB-D SLAM dense. . . . . 53

3.1 Présentation de notre système RGB-D SLAM. . . . . 66
3.2 Système RGB-D SLAM de Endres *et al.* [31, 32]. . . . . 66
3.3 Représentation paramétrique d'un plan 3D dans son propre repère (vue face). . . . . 73
3.4 Transformation entre le repère local $\mathscr{F}_c$ et le repère global $\mathscr{F}_w$. . . 76
3.5 Algorithme de construction de la carte planaire. . . . . 77
3.6 Chevauchement de plans. . . . . 78

4.1 Erreur de l'estimation des inliers au plan . . . . . 90
4.2 Erreur de détection des régions planaires . . . . . 90
4.3 Notre scène expérimentale avec les plans détectés. . . . . 92
4.4 Variations du nombre de points inliers au plan détecté en fonction du facteur $\lambda$ et de la distance du plan. . . . . 93
4.5 Variations du nombre de points d'intérêt 3D planaires (du plan détecté) en fonction du facteur $\lambda$ et de la distance du plan. . . . . 94
4.6 Trajectoire estimée et Vérité terrain. Séquence fr3/structure texture far. . . . . 97

4.7 Notre scène expérimentale Bureau. . . . . 99
4.8 Comparaison des régions planaires dans la carte brute du RGB-D SLAM et notre carte. . . . . 100
4.9 Exemple d'une carte 3D planaire générée par notre système. **(à gauche)** Modèles de plans et leurs nuages de points correspondants. **(à droite)** Notre carte planaire avec les intersections de plans. . . . . 101
4.10 Comparaison des représentations basée-points et basée-plans. (Vue du dessus) . . . . . 102
4.11 Vue approchée d'un plan de la carte 3D : basée-points **(en haut)** et basée-plans **(en bas)** . . . . . 103
4.12 Dispositif expérimental pour l'évaluation des rotations estimées par notre système. . . . . 104
4.13 Exemples des emplacements d'évaluation dans notre scène bureau. 105
4.14 Vue du dessus des reconstructions de notre scène bureau. . . . . 107

# Introduction Générale

## 0.1 Contexte général

Avec l'évolution de la technologie, il est devenu possible d'utiliser des robots intelligents dans de nombreux domaines, même ceux où il y a un risque élevé, comme alternatives aux êtres humains dans l'accomplissement de ces tâches. Lors de la réalisation de ces tâches, ces robots doivent naviguer en toute sécurité dans des milieux inconnus, découvrir leurs environnements et faire face à des situations inattendues. Ainsi, pour pouvoir interagir d'une manière cohérente avec son environnement, un robot a besoin de construire la carte de cet environnement et de connaître son emplacement dans cette carte en permanence. Cela cause un paradoxe, car la construction de la carte nécessite la connaissance de la pose et l'estimation de pose à son tour exige la carte de l'environnement. Par conséquent, les deux processus, l'estimation de poses et la création de la carte, doivent être réalisés simultanément. Ce problème est l'un des plus importants dans le domaine de la vision par ordinateur et la robotique ces dernières décennies, et est connu sous le nom de Simultaneous Localisation and Mapping (SLAM). C'est un processus d'auto-localisation d'un système de perception autonome, un robot par exemple, par rapport à une carte de l'environnement créée et mise à jour par le système lui même. En particulier, pour de nombreuses applications SLAM, il est fondamental que ces tâches soient menées en temps réel. L'hypothèse que l'environnement est rigide et statique, plus facile à réaliser dans un milieu d'intérieur, peut simplifier la résolution du problème.

La cartographie est une tâche importante dans un système de SLAM visuel, notamment si elle est destinée à des applications robotiques telles que la navigation, la reconnaissance d'objets ou la planification de tâches. Pour se déplacer, un robot a besoin d'une bonne description de l'environnement autour de lui, et,

en fonction de la tâche à accomplir cette description varie de simples signaux unidimensionnels jusqu'à des cartes 3D. Le problème du SLAM 2D, utilisant des capteurs ultrasons ou des scanners lasers, a été considéré comme résolu depuis un certain temps et il y a eu de multiples travaux publiés avec leurs implémentations disponibles en open source [77]. Cependant, le SLAM 3D reste toujours un problème en plein développement.

Récemment, les caméras RGB-D (Kinect Microsoft ou Asus Xtion PRO) sont devenues populaires dans les communautés de vision par ordinateur et de la robotique, en raison de leur faible coût et leur capacité de fournir à la fois des image de couleur et de profondeur. Ce type de capteur RGB-D a été introduit dans la résolution du SLAM intérieur et a provoqué une accélération dans le développement des systèmes SLAM 3D. Grâce aux données RGB-D, ces systèmes, appelés RGB-D SLAM, arrivent à estimer les poses 3D de la caméra et à reconstruire l'environnement en trois dimensions. Cependant, ces approches souffrent de nombreux problèmes tels que la gestion de la masse de données, notamment dans les scènes de grandes dimensions, ou la limite de la représentation de l'environnement à des nuages de points lourds et redondants.

Pour un robot mobile commercial, robot accompagnant par exemple, embarquer un algorithme RGB-D SLAM peut être important dans la réalisation des tâches. En particulier, de tels robots ont besoin d'une carte exploitable de leur espace de travail pour la planification du mouvement et la navigation sans collision. Il est alors important d'envisager des solutions rapides et efficaces pour qu'ils puissent générer et consulter la carte en permanence.

## 0.2 Contributions de la thèse

L'objectif de cette thèse est de développer une approche RGB-D SLAM permettant de construire des cartes 3D légères de l'environnement. Cette approche est destinée principalement à des applications embarquant une capacité de traitement limitée. En effet, les cartes 3D reconstruites par les systèmes RGB-D SLAM existants provoquent une consommation importante de la mémoire et demandent des capacités de calcul qui ne sont pas communément disponibles sur des robots commerciaux. La sélection de l'information utile lors de la reconstruction d'une scène visitée est une solution efficace pour économiser les ressources du robot. Dans une scène d'intérieur, l'intégration des structures 3D

dominantes, particulièrement les plans 3D, peut être bénéfique pour la génération d'une carte 3D réduite. Ainsi, le travail réalisé dans cette thèse propose une approche RGB-D SLAM basée sur la modélisation planaire de l'environnement. Notre approche intègre les régions planaires visitées par la caméra dans la résolution du problème du SLAM. L'intérêt d'une telle approche est, d'une part, d'assurer une bonne estimation du mouvement de la caméra en utilisant des points 3D coplanaires moins bruités que les points bruts, et d'autre part de concevoir une représentation 3D légère et compacte de l'environnement. Contrairement aux représentations classiques, la représentation "basée-plans" proposée permet de réduire la taille de la carte 3D et offre une information enrichie de la scène vers une approche plus sémantique. Une telle carte peut être exploitée par d'autres applications telles que la navigation autonome ou la réalité augmentée.

## 0.3 Structure de la thèse

Ce manuscrit est divisé en quatre chapitres principaux. Le premier chapitre présente un rappel des outils de base de vision par ordinateur, nécessaires pour la réalisation de cette thèse ainsi que pour la compréhension des méthodes citées dans ce mémoire. Il offre une description du capteur RGB-D, des principaux algorithmes de vision et des bibliothèques open source utilisés. Le deuxième chapitre détaille les approches proposés dans la littérature pour la résolution du SLAM visuel. Cet état de l'art permet de souligner l'évolution de la manière dont le problème a été traité en distinguant notamment les modélisations du problème et les différentes représentations de l'environnement dans un SLAM. Ensuite, les systèmes RGB-D SLAM(s) récents sont détaillés, analysés et classifiés. Cela permet de justifier les choix adoptés dans cette thèse. Dans le troisième chapitre, nous présentons la nouvelle approche RGB-D SLAM développée dans nos travaux de thèse. Ce chapitre expose les concepts de notre approche basée sur les plans 3D détectés dans des scènes d'intérieur, et les différentes techniques élaborées pour la localisation et la cartographie. Le quatrième chapitre regroupe les études expérimentales réalisées pour évaluer notre approche ainsi que les résultats obtenus. Il présente les protocoles expérimentaux conçus, les contraintes et les performances de notre système RGB-D SLAM. Enfin, la conclusion générale résume les contributions apportées dans cette thèse et met en perspective le travail réalisé.

# Chapitre 1

# Éléments de base

## Sommaire


**1.1 Conventions de notation . . . . . 18**
**1.2 Capteurs RGB-D . . . . . 18**
1.2.1 Mesure de la profondeur . . . . . 19
1.2.2 Construction d'un nuage de points 3D . . . . . 20
1.2.3 Limitations . . . . . 21
**1.3 Notions de vision par ordinateur . . . . . 23**
1.3.1 Point d'intérêts, Détection et Correspondances . . . . . 23
1.3.2 Estimation robuste de la pose . . . . . 26
1.3.2.1 Transformation rigide . . . . . 27
1.3.2.2 Estimation aux moindres carrés du déplacement 28
1.3.2.3 Random Sample Consensus RANSAC . . . . . 28
1.3.2.4 Iterative Closest Point ICP . . . . . 30
**1.4 Bibliothèques Open Source . . . . . 31**
1.4.1 OpenCV . . . . . 31
1.4.2 PCL . . . . . 32
1.4.3 ROS . . . . . 32
**1.5 Conclusion . . . . . 33**


Dans ce chapitre, nous introduirons les éléments de base de la vision par ordinateur couramment employés dans la résolution du problème de Localisation et Cartographie Simultanées (SLAM) par vision ainsi que les notations nécessaires pour la compréhension de ce mémoire. Nous commencerons par la présentation des capteurs RGB-D récemment introduits dans les applications

de vision et plus particulièrement dans le SLAM, et nous discuterons leurs avantages et inconvénients. Puis, nous exposerons les techniques d'estimation de pose basées sur les points d'intérêt en détaillant les différentes étapes du processus depuis la détection de ces points d'intérêt jusqu'au raffinement de la transformation calculée. Enfin, nous présenterons les outils informatiques et modules intégrés dans les développements réalisés durant cette thèse.

## 1.1 Conventions de notation

Nous utiliserons les notations suivantes tout au long de ce mémoire :

| | |
|---|---|
| $\mathcal{D}$ | un ensemble de données |
| $a, b, \ldots$ | scalaires |
| $\mathbf{a}, \mathbf{b}, \ldots$ | vecteurs |
| $\mathbf{A}, \mathbf{B}, \ldots$ | matrices |
| $\mathbf{a}^\top, \mathbf{A}^\top, \ldots$ | vecteur, matrice transposés |
| $x_i$ | $i$ème pose de la caméra |
| $\mathbf{X}$ | vecteur de poses $x = (x_1, \ldots, x_i)$ (trajecroire) |
| $\widetilde{\mathbf{X}}$ | estimation initiale de $\mathbf{X}$ |
| $\hat{\mathbf{X}}$ | estimation de $\mathbf{X}$ |
| $\mathbf{X}^*$ | valeur optimale de $\mathbf{X}$ |
| $\mathbf{T}_{ij}$ | transformation entre deux poses $x_i$ et $x_j$ |
| $\mathscr{F}_c$ | repère local d'une caméra |
| $\mathscr{F}_w$ | repère global |
| $\boldsymbol{p}_c, \boldsymbol{p}_w$ | point 3D dans le repère local, global |
| *Feature point* | point d'intérêt |
| *Features* | primitives visuelles, géométriques, ... |
| *Planar features* | primitives planaires |
| *3D Planar features* | point d'intérêt planaires |

## 1.2 Capteurs RGB-D

Dans un SLAM par vision un intérêt particulier est porté au capteur utilisé. C'est la première interface en contact avec l'environnement et sur laquelle seront basés tout les développements ultérieurs. Récemment des capteurs RGB-D ont été introduit dans les applications de vision et robotique permettant l'expan-

sion des travaux de recherche dans ces domaines. Ces capteurs se composent d'un laser infrarouge projetant des motifs pseudo-aléatoires, une caméra infrarouge et une caméra couleur qui se trouve entre les deux. C'est une alternative intéressante aux scanners laser, traditionnellement utilisés en robotique, car ils fournissent en temps réel une carte de profondeur dense en plus de l'image RGB traditionnelle. Les premières caméras de type RGB-D commercialisées à petit prix sont la Kinect de Microsoft et l'Asus Xtion Pro (toutes les deux basées sur le projecteur de lumière structurée de la société PrimeSense [85, 106]). Elles étaient initialement destinées aux jeux vidéo permettant à l'utilisateur de contrôler le jeu sans utiliser de manette mais uniquement grâce aux mouvements du corps. La lumière structurée permet de mesurer la forme d'un objet en trois dimensions à l'aide des motifs lumineux et d'un système d'acquisition d'images. Ainsi, le laser infrarouge projette un motif sur la scène et la caméra infrarouge observe la déformation du motif projeté sur la scène (Figure 1.1). Les images de couleur et de profondeur mesurent $640 \times 480$ pixels et sont acquises avec une fréquence de 30Hz. La plage des mesures de profondeur varie entre 0.5m et 5.0m en fonction des conditions d'éclairage et de la réflectivité des surfaces. Dans l'objectif de réaliser un SLAM visuel destiné à des applications intérieures, telles que la robotique mobile et la réalité augmentée, le choix d'un tel capteur s'avère bénéfique vu les fonctionnalités qu'il offre et son rapport qualité/prix. Dans la réalisation de cette thèse nous avons utilisé une caméra RGB-D de type Kinect version 1.

### 1.2.1 Mesure de la profondeur

L'image de profondeur d'un capteur RGB-D mesure, pour chaque pixel, la distance dans l'espace 3D entre l'objet représenté par ce pixel et la caméra. Pour mesurer la profondeur, le capteur projette la lumière structurée du spectre infrarouge qui est perçue par la caméra infrarouge. Cette projection fournit alors la disparité $d$ entre le motif projeté et le motif observé par la caméra infrarouge. Connaissant la distance $b$ entre le laser et la caméra, la disparité est utilisée pour estimer la distance d'un point de l'espace par triangulation (Figure 1.1). Ainsi, pour chaque point $\mathbf{k}$, la distance $z_k$ est calculée par la formule suivante [53] :

$$z_k = \frac{z_0}{1 + \frac{z_0}{f*b} d} \tag{1.1}$$

Où $z_0$ est la distance du plan de référence, et $f$ est la distance focale de la caméra. Les valeurs de $z_0$, $b$ et $f$ peuvent être déterminées par la calibration du dispositif projecteur-caméra infrarouge.

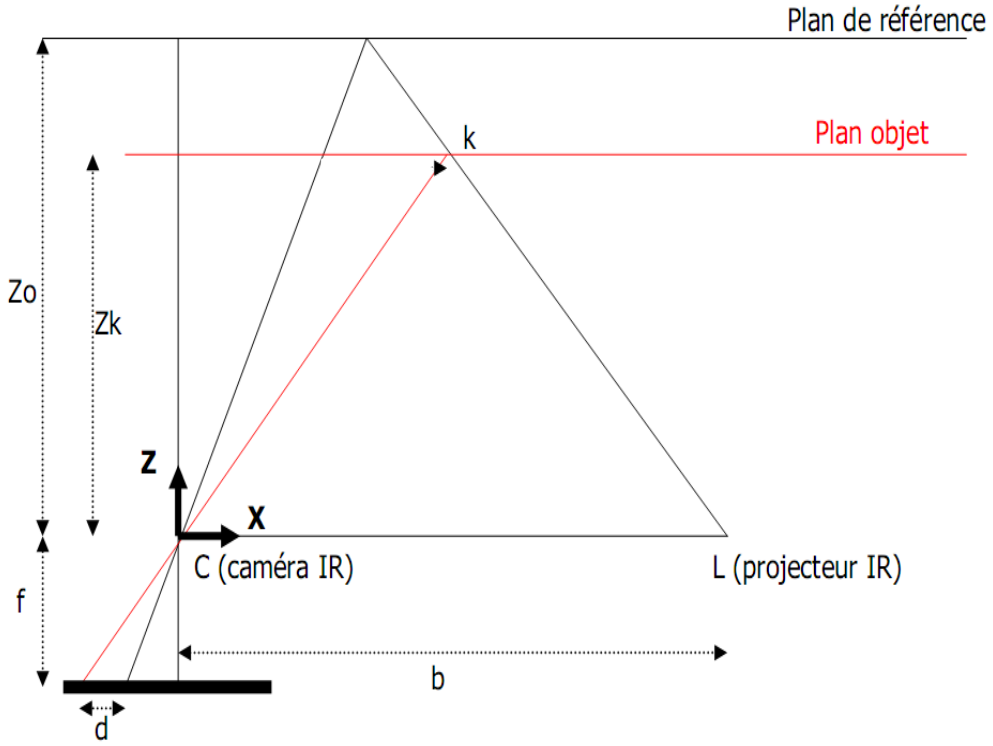



**FIGURE 1.1** – *Principe du calcul de la profondeur en utilisant la lumière structurée.*

### 1.2.2 Construction d'un nuage de points 3D

Les caméras RGB-D fournissent deux types d'informations sur l'environnement. La première information est la même qu'enregistre une caméra classique, c'est-à-dire une image couleur de son champs visuel. La seconde est une image de profondeur, soit la distance à laquelle se situe chaque point enregistré par rapport au point de vue. L'association de l'image couleur (RGB) et de l'image de profondeur (Depth) permet la reconstruction de l'environnement en générant des nuages de points 3D colorés. Afin d'obtenir une précision maximale, l'image de profondeur doit être alignée avec l'image de la couleur. Cependant, pour de nombreuses applications, l'utilisation de la correspondance pixel à pixel entre la profondeur et la couleur est suffisante. Ainsi, les données de profondeur sont utilisés pour créer des nuages de points où chaque pixel

(u,v) d'une image 2D se transforme en un point 3D en utilisant les équations suivantes :

$$z = Depth(u, v) \tag{1.2}$$
$$x = (u - c_x) * z / f_x \tag{1.3}$$
$$y = (v - c_y) * z / f_y \tag{1.4}$$

Où $c_x, c_y, f_x et f_y$ sont les paramètres intrinsèques de la caméra de profondeur, qui sont estimés par calibration. La figure 1.2 montre l'exemple d'un nuage de point construit à partir d'une image de profondeur et d'une image couleur obtenues par une Kinect.

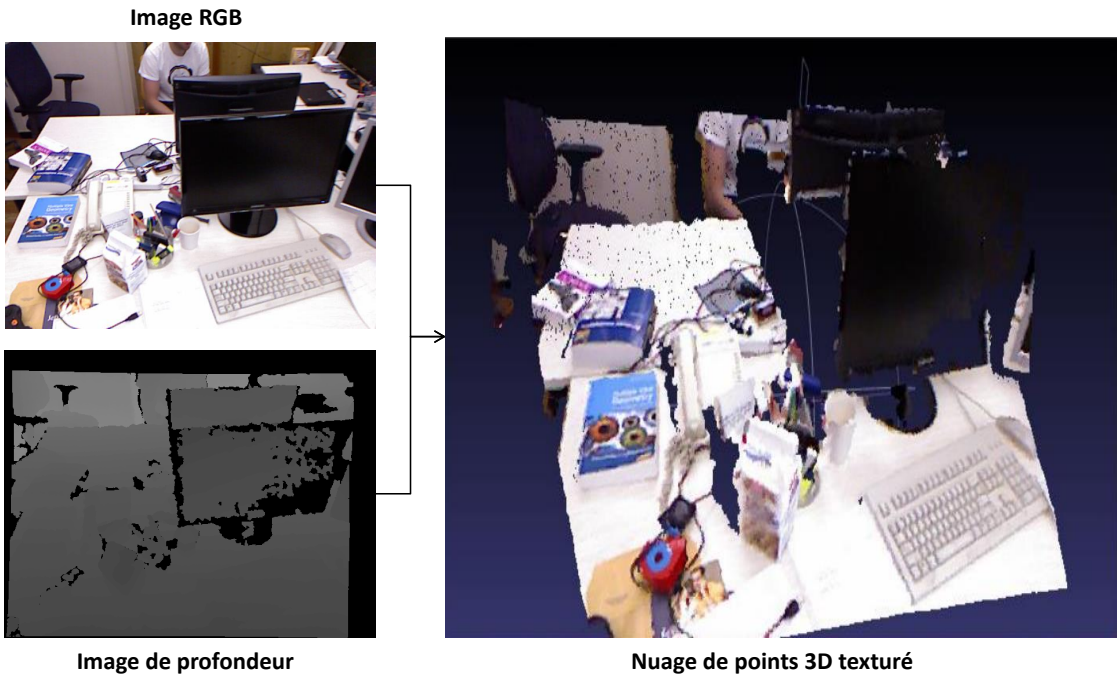



**FIGURE 1.2** – *Génération d'un nuage de points 3D texturé à partir de la correspondance Couleur-Profondeur.*

### 1.2.3 Limitations

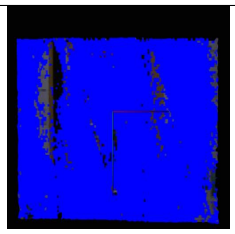
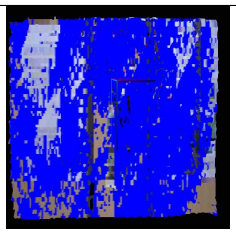
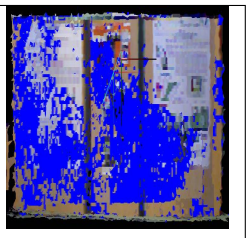
Le capteur RGB-D est un bon choix pour les applications intérieures, telles que le SLAM intérieur, vu son coût faible et les données 3D reçues avec une grande fréquence. Cependant, comme tout autre capteur, il présente quelques inconvénients. Explicitement le capteur est très sensible à l'éclairage en raison

du projecteur de lumière structurée qu'il utilise. Il n'est donc généralement pas applicable en plein soleil. Un problème se pose aussi lorsque la lumière infrarouge est directement réfléchie (surfaces réfléchissantes), absorbée (surface sombres) ou transmise (surface transparentes). Les nuages de points ainsi produits contiennent des lacunes [108] comme on peut le voir dans la figure 1.2 : accoudoir, pied écran.

D'autre part, l'une des caractéristiques des caméras RGB-D est de quantifier la distance selon un pas qui augmente à mesure que nous nous éloignons du point de vue. Ainsi, les éléments proches présentent une erreur de mesure plus faible que les éléments lointains. Dans des applications de cartographie, différents travaux [53, 108, 3] recommandent l'utilisation de la Kinect sur des distances comprises entre $0.8m$ et $3.8m$ pour garantir une bonne précision.

Nous avons effectué une série d'expériences pour évaluer la fiabilité des données du capteur RGB-D. Étant donné que notre objectif est l'utilisation des plans 3D détectés dans les nuages de points en provenance d'une Kinect afin de construire des cartes 3D, la précision des plans détectés est par conséquent un critère primordial pour la robustesse de notre approche. La description détaillée du protocole de la détection des plans sera donnée dans le troisième chapitre lors de la présentation de notre approche (page 70). Cette détection est réalisée sur les cartes de profondeur par l'algorithme RANSAC (pseudo-code page 30). Ici nous examinons les distances estimées de plans, détectés avec une Kinect fixe, par rapport aux distances réelles dans la scène (entre les plans et la caméra). Pour le même plan, nous avons effectué plusieurs détections et mesuré la précision de la distance estimée. Différentes distances Kinect-Plan ont été utilisées pour définir les limites de détection. Le tableau 1.1 reporte les résultats de cette expérience sur une moyenne de 20 détections par plan. Les images dans le tableau montrent les points coplanaires détectés qui sont colorés en bleu.
Il est évident que la précision de l'estimation diminue quand les plans se trouvent loin de la caméra. Ceci est dû au faible nombre d'inliers pour les grandes distances, ce qui est du à son tour au bruit caractéristique du système de triangulation projecteur-caméra infrarouge. À partir de distances supérieures à $3.5m$ l'erreur devient très importante et l'estimation de points coplanaires échoue. Ces résultats rejoignent les conclusions des travaux antérieurs [53, 108, 3] quant aux limites de l'utilisation robuste des capteurs RGB-D.

**TABLE 1.1** – *Précision de la Kinect en fonction de la distance d'un plan.*

| |  |  |  |  |
|---|---|---|---|---|
| Distance réelle | 1.50m | 2.50m | 3.00m | 3.50m |
| Distance estimée | 1.45m | 2.57m | 3.09m | 3.31m |
| Nombre d'inliers | 11039 | 7199 | 1552 | 1014 |

## 1.3 Notions de vision par ordinateur

### 1.3.1 Point d'intérêts, Détection et Correspondances

Dans les systèmes SLAM visuel classiques, pour estimer le mouvement d'une caméra entre deux poses, l'approche standard suit le chemin habituel de la détection, la description et la mise en correspondance (matching) de points d'intérêt (feature points). Dans cette thèse, nous avons opté pour un SLAM visuel qui utilise une caméra RGB-D pour l'estimation de la trajectoire et la reconstruction de l'environnement. Le fait que les images couleurs soient aussi disponibles offre la possibilité d'utiliser les pipelines visuels courants en parallèle avec les données de la profondeur. Dans nos travaux, nous adoptons une approche éparse "*sparse*" (section 2.2.1, page 46) qui repose sur l'utilisation des points d'intérêt pour estimer le déplacement (la transformation) $\mathbf{T}_{ij}$ entre deux poses $\mathbf{x}_i$ et $\mathbf{x}_j$ de la caméra. En effet, les points d'intérêt facilitent l'association des données et la rendent plus rapide. Dans la littérature nous trouvons plusieurs méthodes pour la détection et la description des points d'intérêt. Elles sont basées le plus souvent sur les zones saillantes ou les coins (corners) des images. Nous pouvons citer à titre d'exemple les SIFT[59], les SURF[8] et les ORB[79] connus pour leur robustesse et leur efficacité. Le voisinage d'un point d'intérêt est caractérisé par un descripteur $d_i$. C'est une signature qui permet d'identifier ce point de manière unique pour pouvoir le retrouver dans d'autres images. Les mesures de similarité des descripteurs peuvent être définies. Ensuite, le problème principal est l'obtention des correspondances 2D-2D ou image à image sans connaissance des poses de la caméra. Étant donné une nouvelle image avec des descripteurs associés aux points d'intérêts, la correspondance est obtenue en cherchant l'appariement des descripteurs de cette image parmi ceux d'une

autre. Pour une paire de descripteurs ($d_i$, $d_j$), le matching est effectué généralement en calculant leur distance dans l'espace du descripteur. Pour les descripteurs populaires SIFT et SURF la mesure de la distance utilisée est la distance Euclidienne, à savoir, $\mid d_i - d_j \mid$. Les descripteurs binaires comme ORB, où un descripteur est une chaîne de bits, emploient la distance de Hamming (nombre de positions binaires de valeurs différentes entre deux descripteurs). Cependant, la distance n'est pas un critère efficace pour l'association des points d'intérêt étant donné que la distance entre les descripteurs correspondants peut varier considérablement. Par ailleurs, plusieurs points d'intérêt d'une image courante peuvent n'avoir aucun correspondant correct dans une autre image si ils proviennent du bruit de fond ou dans le cas où ils ne sont pas détectés dans les deux images. Lowe [59] a proposé d'utiliser le rapport entre la distance des premier et deuxième plus proches voisins (*respectivement* $d_{n1}$) et $d_{n2}$) dans l'espace des descripteurs des points d'intérêt comme un seuil dans la mise en correspondance. Pour SIFT et SURF, si $d_i$ est le descripteur que nous cherchons à apparier, le ratio test est :

$$r = \frac{\mid d_i - d_{n1} \mid}{\mid d_i - d_{n2} \mid} \tag{1.5}$$

Dans l'hypothèse où un point d'intérêt correspond seulement à un seul autre point d'intérêt dans deux images différentes, la distance au second plus proche voisin devrait être beaucoup plus grande. Ainsi, un seuil sur le rapport $r$ peut être utilisé efficacement pour contrôler les mauvaises correspondances. Expérimentalement, le rejet de tous les appariements dans lesquels ce rapport est supérieure à 0.8 élimine 90% des fausses correspondances tout en perdant moins de 5% de celles qui sont correctes. Pour accélérer la recherche des appariements lorsque le nombre candidat est important, Muja et Lowe [68] ont proposé l'algorithme FLANN (Fast Library for Approximate Nearest Neighbors) pour la recherche des plus proches voisins. Cet algorithme utilise les Arbres Hiérarchiques K-means (K-means Tree) ce qui rend la recherche des voisins proches plus rapide dans les grands ensembles de données. La libraire OpenCV (section 1.4.1) propose son propre algorithme de matching BruteForce (BF). Cet algorithme offre plusieurs sortes de matching utilisant la distance Euclidienne ou la distance de Hamming (pour les descripteurs binaires). La différence avec le FLANN est que le BruteForce effectue une recherche exhaustive en parcourant tous les points d'intérêt de l'image pour trouver les candidats possibles pour le matching. Ainsi, pour un grand ensemble de points d'intérêt le meilleur choix sera d'utiliser le FLANN car il cherche le meilleur matching à partir d'un sous

ensemble de potentiels candidats contrairement au BruteForce. Notons que lors de l'utilisation des descripteurs binaires BruteForce doit être utilisé puisque FLANN n'utilise que la distance euclidienne.

Ainsi, les ensembles de points appariés obtenus vont servir pour l'estimation de la pose de la caméra après un déplacement. Néanmoins, cette estimation peut être affectée par les mauvais appariements ce qui produira une mauvaise trajectoire et par conséquent une reconstruction peu fidèle de l'environnement réel. Afin d'éviter ce genre de situation nous faisons appel à des méthodes Robustes après l'estimation initiale de pose (Section suivante).

Il faut mentionner aussi que les performances d'un système SLAM basé sur les points d'intérêt dépendent fortement des algorithmes et des valeurs des paramètres utilisés pour la détection, la description et l'appariement. Plusieurs études ont été réalisées pour comparer les différentes composantes de ce pipeline et trouver les meilleures combinaisons [76], [31], [38].

Dans l'objectif de choisir entre les algorithmes cités précédemment lors de l'application du pipeline relatif aux points d'intérêt dans notre thèse, nous avons procédé à des expériences comparatives. Dans ces expériences nous avons utilisé des images couleurs dans lesquelles des points d'intérêt ont été détectés, décrits et matchés.
Le tableau 4.5 présente les résultats de la détection et description des points d'intérêt 2D dans la même image par les différents algorithmes SIFT, SURF et ORB. La comparaison des temps d'exécution de ces algorithmes montre que les ORB sont plus rapides que les SIFT et SURF. Cependant, l'étude des performances de ces algorithmes réalisée dans [32] montre que les erreurs de l'estimation de poses résultantes de l'utilisation des ORB sont beaucoup plus importantes par rapport aux deux autres algorithmes. En effet, la grande faiblesse des ORB est la redondance dans la détection des points d'intérêt. Plusieurs points d'intérêt peuvent être enregistrés au même pixel de l'image ce qui engendre des erreurs lors du matching entre deux images.

Ensuite, nous avons comparé le temps d'exécution des algorithmes FLANN et BruteForce pour le matching de points d'intérêt détectés dans deux images successives. Le tableau 1.3 présente cette comparaison en fonction du nombre maximal de points d'intérêt détectés dans chacune des deux images. Les résultats du tableau confirment les remarques introduites précédemment quant à la rapidité du BruteForce lorsque le nombre de points d'intérêt est relativement

**TABLE 1.2** – *Temps de processus de détection et description des points d'intérêt en fonction de l'algorithme utilisé.*

| | Nombre de points d'intérêt détectés | Temps d'exécution Détection + Description |
|---|---|---|
| SIFT | 999 | 0.29s |
| SURF | 871 | 0.18s |
| ORB | 998 | 0.02s |

faible (<500) et inversement pour le FLANN qui devient plus performant quand ce nombre augmente.

**TABLE 1.3** – *Comparaison entre le temps d'exécution du matching des algorithmes FLANN et BruteForce.*

| | Nombre de features 2D | 200 | 500 | 900 |
|---|---|---|---|---|
| Temps d'exécution | FLANN | 0.004s | 0.012s | 0.024s |
| | BruteForce | 0.002s | 0.017s | 0.039s |
| Nombre d'appariements | FLANN | 101 | 271 | 488 |
| | BruteForce | 98 | 272 | 498 |

Nous pouvons conclure que les choix dépendent du contexte de l'application, de l'environnement ainsi que des techniques utilisés. Par exemple, pour une application de reconnaissance de visage, fonctionnant en mode hors ligne, et exigeant une grande robustesse contre les erreurs de détection et de description des points d'intérêt, le meilleur choix serait les SIFT. Cependant, pour un robot mobile de capacités calculatoire limitées et qui réalise des traitements en temps réel, les SURF peuvent être utilisés. D'autre part, le matching dépend de la nature des descripteurs et du nombre de points d'intérêt détectés. Une approche adaptative peut donc être adoptée pour choisir l'algorithme du matching en fonction des données traitées. Notons aussi que la combinaison entre différents algorithmes pour la détection et la description des points d'intérêt est toujours possible.

### 1.3.2 Estimation robuste de la pose

L'objectif du pipeline détection, description et appariement des points d'intérêt (section précedente) est de maximiser efficacement la probabilité d'avoir des correspondances valides de points 2D de la scène dans les images observées tout en minimisant les occurrences des fausses correspondances. À partir

de ces appariements plusieurs techniques permettent d'estimer le mouvement 3D d'une caméra. Néanmoins, cette estimation doit être raffinée afin de garantir un certain niveau de robustesse dans l'estimation des poses. Dans cette section nous détaillons les algorithmes utilisés pour l'estimation de la pose à partir de deux ensembles de points 3D.

#### 1.3.2.1 Transformation rigide

Avant d'aller plus loin dans les détails de l'estimation de la pose, nous rappelons quelques notions sur la transformation rigide entre deux repères dans l'espace 3D. Cette transformation $\mathbf{T}$ se compose d'une Rotation $\mathbf{R}$ représentée par une matrice (3×3), et d'une Translation $\mathbf{t}$ qui correspond à un vecteur (3×1). Un point $\mathbf{p}$ dans l'espace peut être transformé rigidement (rotation et translation) de son propre repère vers un autre repère par l'intermédiaire de la formule suivante :

$$\mathbf{p}_b = \mathbf{R}_{ab}\mathbf{p}_a + \mathbf{t}_{ab} = \begin{pmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{pmatrix} \mathbf{p}_b + \begin{pmatrix} t_0 \\ t_1 \\ t_2 \end{pmatrix} \tag{1.6}$$

où les indices des points indiquent les repères dans lesquels ils sont présentés.

Cette expression peut être écrite sous une forme plus compacte en utilisant les coordonnées homogènes du point et la matrice de transformation $\mathbf{T}_{ab}$ (4×4) combinant la rotation et la translation :

$$\mathbf{p}_b = \mathbf{T}_{ab}\mathbf{p}_a = \begin{pmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{12} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{p}_b = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0^\top & 1 \end{pmatrix} \mathbf{p}_a \tag{1.7}$$

Après $N$ transformations, pour trouver les coordonnées du point $\mathbf{p}$ dans un repère il suffit de multiplier les matrices des transformations enregistrées à chaque déplacement de la caméra à partir de ce repère :

$$\mathbf{p}_n = \mathbf{T}_{mn} * \ldots * \mathbf{T}_{bc} * \mathbf{T}_{ab}\mathbf{p}_a \tag{1.8}$$

#### 1.3.2.2 Estimation aux moindres carrés du déplacement

Il existe plusieurs approches pour estimer une transformation entre deux poses lors d'un déplacement d'une caméra. Dans les systèmes de vision, la technique des moindres carrés est souvent utilisée. Dans cette thèse, elle a été directement appliquée pour résoudre le problème de l'estimation initiale de la pose et elle correspond aussi à la composante de base d'un optimiseur de graphe de poses.

Nous nous limitons ici à la description d'une méthode basée sur la technique des moindres carrés pour estimer la transformation entre deux ensembles de points 3D. La méthode est détaillée dans le travail de S. Umeyama [98]. L'auteur a démontré que la transformation entre deux ensembles de points 3D peut être calculée, à partir des correspondances établies entre ses deux ensembles, en utilisant la Décomposition en Valeurs Singulières (SVD) d'une matrice.

Soit la matrice $\mathbf{A}$ qui représente un ensemble de points 3D (nuage de points) et $\mathbf{B}$ est la matrice représentant leurs points appariés dans l'ordre. Les deux matrices ayant alors la même dimension.
Si $\mathbf{UDV}^\top$ est le résultat de la décomposition en valeurs singulières de la matrice $\mathbf{M} = \mathbf{AB}^\top$, alors la matrice de rotation entre les deux nuages de points est :

$$\mathbf{R} = \mathbf{UV}^\top \tag{1.9}$$

Notons que le déterminant de la matrice $\mathbf{R}$ doit être égal à 1 pour que celle-ci soit bien une matrice de rotation.
En utilisant cette rotation, la translation est définie par :

$$\mathbf{t} = \delta_B - \delta_A * \mathbf{R} \tag{1.10}$$

Où $\delta_A$ et $\delta_B$ représentent respectivement les centroïdes des nuages de points A et B. Dans cette thèse nous utilisons une implémentation de cette méthode développée par la librairie "Point Cloud Library" (PCL) (section 1.4.2).

#### 1.3.2.3 Random Sample Consensus RANSAC

Random Sample Consensus (RANSAC) [35] est une méthode itérative permettant l'estimation robuste des paramètres d'un modèle contre les outliers [1]

1. points de données ne se conformant pas au modèle à estimer

dans les données d'entrée. Comme indiqué auparavant, les méthodes robustes peuvent être utilisées pour diminuer l'effet des données aberrantes à partir d'une estimation initiale. En utilisant de telles méthodes, même si l'ensemble de données contient de nombreuses valeurs aberrantes, il y a une grande probabilité que l'estimation soit calculée uniquement à partir des inliers (données correspondants au modèle recherché). Dans le contexte de l'estimation de la pose, nous employons l'algorithme RANSAC pour raffiner l'estimation initiale de la transformation entre deux nuages de points 3D. En effet, les mauvais appariements, lorsqu'ils sont incorporés naïvement dans une estimation aux moindres carrés (Sous-section précédente), engendrent généralement une erreur grave dans l'estimation finale. RANSAC résout le problème en utilisant itérativement des échantillons aléatoires des points 3D appariés pour mettre à jour l'estimation. A chaque itération, la qualité de la solution est évaluée en fonction du nombre d'échantillons de l'ensemble de points 3D d'origine satisfaisants l'estimation. Typiquement, il calcule un seuil de l'erreur résiduelle par rapport à cette estimation. Cette erreur correspond aux erreurs entre les points d'intérêt appariés après application de la transformation estimée. Les tests de validation s'appuient sur le calcul de la distance Euclidien ou la distance de Mahalanobis entre ces points. La distance de Mahalanobis est une mesure de distance basée sur la corrélation entre les données (ici points d'intérêt appariés). Contrairement à la distance Euclidienne où toutes les composantes des données sont traitées de la même façon, cette distance accorde un poids moins important aux composantes avec peu de confiance. Généralement ce poids est modélisé par une matrice de covariance $\Sigma$ qui représente les incertitudes de mesures. La distance de Mahalanobis est calculée en utilisant la formule suivante :

$$D(x) = \sqrt{\Delta_p^\top \Sigma^{-1} \Delta_p}$$

Où $\Delta_p$ correspond à l'écart entre la projection des points d'intérêt 3D correspondants dans les deux sens (de la pose actuelle à la pose précédente et inversement) en utilisant la transformation estimée. Ainsi, après le calcul de l'erreur résiduel, la solution peut être (éventuellement) améliorée en incorporant à chaque itération les inliers actuels. L'ensemble du processus est répété avec des nouveaux échantillons pour un nombre d'itérations donné. Généralement la probabilité que l'estimation générée à partir des valeurs aberrantes soit soutenue par plus d'échantillons que l'estimation calculée à partir des inliers est marginale, notamment dans les données de grande dimension. L'estimation avec le

nombre d'inliers le plus élevé est donc retenue comme la solution finale. Le nombre d'itérations nécessaires pour trouver une solution optimale avec une certaine probabilité peut être définie par l'utilisateur ou dynamiquement en fonction du rapport des inliers et des outliers (valeurs aberrantes). Le pseudo-code du RANSAC est donné dans l'algorithme 1.

**Algorithme 1** Pseudo-code de l'algorithme RANSAC

**Entrée:** Données $\mathcal{D}$, Seuil $\tau$, Nombre d'itération $i$
**Sortie:** Paramètres du modèle $\mathcal{P}$
  $n = 0$ //Taille de l'ensemble initial des inliers
  **Pour** 1 **À** $i$ **faire**
    $\mathcal{S}$ = Tirage aléatoire d'un ensemble de Données($\mathcal{D}$)
    $\tilde{\mathcal{P}}$ = Estimation des Paramètres du modèle de ($\mathcal{S}$)
    $\mathcal{E}$ = Calcul des l'Erreur ($\mathcal{D}$, $\tilde{\mathcal{P}}$)
    $\mathcal{I}$ = Calcul de l'ensemble des Inliers ($\mathcal{D}$, $\mathcal{E}$, $\tau$) //Mesure de la qualité
    **Si** taille de $\mathcal{I} > n$ **Alors**
      $\mathcal{P}$ = Estimation des paramètres du modèle de ($\mathcal{I}$)
      $n$ = taille de $\mathcal{I}$
    **Fin Si**
  **Fin Pour**
  **Retourner** $\mathcal{P}$

RANSAC permet alors de ne sélectionner que les observations correctes et élimine les mesures aberrantes du problème. Il existe plusieurs variantes de cette méthode (PROSAC, MLESAC, etc.) dont des évaluations ont été proposées dans [78] et [14].

#### 1.3.2.4 Iterative Closest Point ICP

Parmi les méthodes populaires pour l'estimation de pose que nous trouverons dans plusieurs systèmes par la suite, nous tenons à citer brièvement l'algorithme ICP (Iterative Closest Point) introduit la première fois dans [107]. En effet, cet algorithme est largement utilisé dans les systèmes RGB-D SLAM pour l'estimation du mouvement de la caméra en alignant les nuages de points 3D consécutifs lorsqu'une estimation initiale de la pose est connue. La technique utilisée est la minimisation d'une métrique d'erreur qui correspond généralement à la distance entre les deux ensembles de points. Plus concrètement, l'algorithme cherche itérativement à minimiser l'erreur entre les plus proches voisins. Partant de l'estimation initiale, à chaque itération il met en correspondance les points 3D dans les deux nuages de points en se basant sur le critère

des plus proches voisins et ré-estime une nouvelle transformation jusqu'à la convergence. Le critère de convergence est définit par un seuil au dessous duquel l'erreur résiduelle des distances entre les points appariés est acceptable. Cette erreur est calculée après le recalage des deux nuages de points en utilisant la nouvelle transformation. Sinon, l'algorithme peut s'arrêter quand le nombre d'itérations maximal est atteint. L'inconvénient majeur de cet algorithme est sa complexité [5] qui rend le temps de calcul assez long (de l'ordre de 3 minutes pour trouver la correspondance entre deux nuages de points). Pour remédier à ce problème, les chercheurs ont tendance ces dernières années à utiliser des processeurs graphiques de haute performance de type GPU ce qui peut être aussi coûteux en terme d'implémentation.

## 1.4 Bibliothèques Open Source

Une grande partie du travail de cette thèse a été réalisée en utilisant des logiciels et frameworks open source développé en `C++`. Notons que la communauté *OpenSLAM* [88] collecte différentes implémentions du SLAM et offre aux chercheurs du domaine l'accès et la réutilisation des travaux déjà validés. Ici nous citons brièvement les bibliothèques de base pertinentes.

### 1.4.1 OpenCV

Open source Computer Vision (OpenCV) [50] est une bibliothèque éminente de vision par ordinateur écrite en `C/C++` et `Python` qui fonctionne sous Linux, Windows et MacOS et plus récemment Android et iOS. Elle est conçue pour l'efficacité de calcul, en particulier pour les applications exigeant une exécution en temps réel. Grace au grand nombre d'algorithmes qu'elle propose, elle construit une simple infrastructure de vision informatique qui facilite le développement d'applications de vision relativement complexes. Dans le contexte d'un SLAM visuel, nous avons pu profiter de cette bibliothèque notamment dans l'utilisation des algorithmes les plus courants de vision par ordinateur, notamment pour les traitements relatifs aux points d'intérêt (section 1.3.1), les transformations et la détection des frontières des régions planaires, ainsi que pour la gestion des flux d'images et des interfaces graphiques.

### 1.4.2 PCL

Point Cloud Library (PCL) [81] figure parmi les bibliothèques de vision 3D les plus utilisées ces dernières années. C'est une référence pour les traitements des nuages de points qui est très appréciée par la communauté de vision 3D. En outre, elle offre plusieurs fonctionnalités 2D basiques telle que la détection des points d'intérêt. PCL fournit de nombreuses méthodes dédiées à la géométrie, la segmentation, l'estimation des transformations et également à la visualisation 3D. Dans notre travail nous avons été amené au traitement des données provenant d'un capteur RGB-D, cette bibliothèque nous a principalement servie pour la construction des nuages de points 3D à partir des données de profondeur fournies par la Kinect et la détection les points coplanaires dans ces nuages de points.

### 1.4.3 ROS

Robot Operating System [1] (ROS) est un ensemble de bibliothèques logicielles et d'outils (open source) dédiés au développement des applications robotiques. C'est un méta système d'exploitation, en pleine évolution, intégré sous Linux et est géré par la communauté robotique ayant la volonté de mettre à disposition des utilisateurs une base commune d'outils, frameworks et travaux de recherches du domaine afin de standardiser les futurs développements. Le système est conçu sous la forme d'un ensemble de processus appelés "noeuds" qui communiquent entre eux de manière directe ou en utilisant des sujets "topics" typés par les données "messages" qu'ils reçoivent ou diffusent. Cette architecture permet la répartition des tâches entre plusieurs noeuds ce qui contribue efficacement à l'optimisation de l'exécution des programmes et est en cohérence avec les bonnes pratiques du développement d'applications informatiques. Comme tout système d'exploitation, ROS permet l'installation et l'utilisation de bibliothèques extérieures. Le travail présenté dans cette thèse est basé sur un framework développé sous ROS et qui sera traité plus tard dans ce manuscrit. Les bibliothèques OpenCV et PCL décrites précédemment sont installées sous ROS et ont été utilisées à travers ce système.

1. `http://www.ros.org/`

## 1.5 Conclusion

Nous avons présenté dans ce chapitre les outils fondamentaux intervenants dans la réalisation d'un SLAM visuel et plus particulièrement lors de l'utilisation du capteur RGB-D choisi pour la réalisation de cette thèse. Nous avons tout d'abord détaillé la modélisation du capteur, à savoir le calcul des valeurs de la profondeur des objets et la construction des nuages de points texturés à partir de la correspondance entre les images couleurs et les cartes de la profondeur. Les conditions d'utilisation du capteur ont été ensuite évaluées et discutées. Nous avons vu que l'erreur des valeurs de la profondeur augmente avec la distance du capteur et par conséquent pour garantir des résultats robustes lors de l'utilisation de la caméra RGB-D dans notre travail les limites ont été définies. Ce point permettra, comme cela sera détaillé par la suite, de proposer un algorithme visant à diminuer l'effet des données aberrantes sur les deux tâches principales du SLAM, à savoir la localisation et la cartographie. Ensuite, nous avons rappelé quelques algorithmes standards de vision par ordinateur omniprésents dans les systèmes de SLAM visuel, et qui sont également applicables pour les images couleurs du capteur RGB-D, tel que l'estimation de la pose basée sur les points d'intérêt. Cela facilitera les choix techniques que nous ferons par la suite. Nous avons aussi précisé l'intérêt du raffinement de la transformation issue d'un tel pipeline afin d'éviter des erreurs grossières dans l'estimation de la trajectoire. Finalement, nous avons tenu à présenter les librairies open source utilisées pendant la réalisation de cette thèse et qui sont mis à la disposition des utilisateurs par la communauté scientifique. Dans le chapitre suivant, nous allons présenter l'état de l'art du SLAM visuel et plus particulièrement les approches utilisant une caméra RGB-D "Les RGB-D SLAM(s)". Ces approches récentes ont suscitées notre intérêt étant donné qu'elles utilisent le même capteur que nous avons prévu pour cette thèse. Ainsi, l'étude et l'analyse des travaux proposés nous permettrons de définir les avantages et les inconvénients de chacun de ces travaux. La comparaison détaillée des approches permettra alors de faire les choix convenables en vue de produire une représentation légère de l'environnement basée-primitives.

# Chapitre 2

# Localisation et Cartographie 3D par vision

## Sommaire


**2.1 Introduction . . . . . 36**
2.1.1 Évolution du SLAM visuel . . . . . 36
2.1.2 Modélisation GraphSLAM . . . . . 38
2.1.3 Représentations de l'environnement . . . . . 41
2.1.3.1 Cartes Métriques . . . . . 41
2.1.3.2 Cartes Topologiques . . . . . 42
2.1.3.3 Cartes Sémantiques . . . . . 44
**2.2 Systèmes RGB-D SLAM . . . . . 45**
2.2.1 RGB-D SLAM sparse . . . . . 46
2.2.2 RGB-D SLAM dense . . . . . 49
2.2.3 Évaluations et Comparaison . . . . . 51
2.2.4 Synthèse . . . . . 56
**2.3 Conclusion . . . . . 59**


Dans ce chapitre nous commencerons par un rappel de l'état de l'art des algorithmes et modélisations proposés pour l'implémentation d'un SLAM par vision (SLAM visuel). Nous détaillerons particulièrement l'approche basée sur les graphes de poses qui a été adoptée dans cette thèse pour la résolution du problème. Cette approche propose une résolution aisée à la fois du problème SLAM et des sous problèmes relatifs notamment la fermeture de boucle, la propagation de l'erreur et la bonne gestion des ressources. Aussi, nous mettrons l'accent sur

les méthodes de reconstructions de l'environnement existantes en développant les atouts et faiblesses de chaque type de cartographie exposé. L'objectif sera ici de choisir la représentation la plus adaptée aux objectifs de la thèse, notamment la génération de cartes 3D de tailles réduites permettant l'implémentation de l'algorithme sur des dispositifs de faible coût.Ensuite, nous nous concentrerons sur les approches SLAM récentes qui utilisent les capteurs RGB-D. De nombreux travaux ont été proposés que nous proposerons de les classifier et de faire une analyse comparative de leurs performances. Enfin, nous présenterons les choix effectués dans le cadre de cette thèse.

## 2.1 Introduction

La communauté du SLAM a proposé une grande variété d'approches qui couvrent différents types de capteurs, différentes techniques d'optimisation de l'erreur et différentes représentations cartographiques. Cependant, le SLAM visuel (par vision) représente une grande part des travaux réalisés et suscite un intérêt particulier. En effet, les caméras permettent une implémentation légère sur les robots mobiles et facilitent la création d'une carte de l'environnement à partir des correspondances visuelles entre les images acquises. Dans l'espace de travail d'un robot, la disponibilité de cette carte est indispensable pour accomplir plusieurs tâches autonomes telles que la localisation, la planification de mouvement et l'évitement d'obstacles. Cette section présente l'historique du développement des techniques de SLAM visuel suivie de la typologie des représentations possibles de l'environnement.

### 2.1.1 Évolution du SLAM visuel

Le SLAM est un axe de recherche particulièrement actif depuis les années 80. Au début, la localisation d'un robot était fondée sur des cartes connues a priori des environnements dans lesquels il a été déployé. Par exemple, Leonard *et al.* [58] effectuent la localisation du robot en utilisant une carte prédéfinie détectée par des réseaux sonar. Effectuer à la fois la localisation et la cartographie dans la même boucle simultanée restait difficile en raison de la nature corrélée des deux problèmes : Les imprécisions dans l'un conduisent à des résultats erronés dans l'autre. Les éléments théoriques fondamentaux présents dans les sys-

tèmes de SLAM modernes ont été définis dans les travaux pionniers de Smith et Cheeseman en 1986 [87] et de Moutarlier et Chatila [67] en 1989. Ils ont proposé de conserver des estimations probabilistes conjointes à la fois des endroits marquants de la carte "*amers visuels (landmarks)*" et des poses du capteur, puis les raffiner en utilisant un filtre de Kalman étendu (EKF) à chaque réception de nouvelles mesures de pose et de la carte. L'EKF est basé sur l'utilisation de l'état du système précédemment estimé (pose de la caméra et amers de la carte) et les commandes de contrôle et/ou le modèle de mouvement pour prédire la pose actuelle. Ensuite, cette pose est mise à jour en utilisant les observations des amers dans la caméra actuelle. Depuis, le SLAM est devenu un domaine de recherche important pour la robotique mobile et plusieurs travaux on été développés autour de l'approche probabiliste EKF [9, 71, 11]. L'un des premiers systèmes à mettre en œuvre une approche SLAM utilisant la vision (SLAM visuel) était celui de Davison *et al.* en 1998 [19]. Ils ont ainsi réalisé un système de vision active en focalisant la caméra sur des amers visuels (endroits particuliers) de la scène pour la construction d'une carte 3D. Cependant, la localisation de la caméra a été effectuée par une plate-forme (robot à roues) se déplaçant dans l'espace 2D. Par la suite, Davison a développé "*MonoSLAM*" en utilisant une caméra monoculaire portée à la main entre 2003 et 2007 [17, 18]. Ce système, basé sur la correspondance des points d'intérêt détectés dans les images, effectue un suivi incrémental en temps réel de la caméra avec une solution de cartographie en suivant l'approche probabiliste EKF. Un ouvrage rassemblant les bases théoriques du SLAM probabiliste a été publié par Thrun, Burgard, et Fox "Probabilistic Robotics" [95].

Une alternative à l'EKF a été proposée par Montermerlo *et al.* [66, 65] en 2002. Dans cette approche connue sous le nom de "*FastSLAM*", le filtre de Kalman est remplacé par un filtre particulaire *Rao-Blackwellisé* qui simule plusieurs hypothèses de l'état du système et calcule les amers visuels indépendamment. En effet, l'utilisation d'un filtre particulaire pour estimer de la trajectoire du robot et les amers de la carte est très coûteux. Ainsi, connaissant la trajectoire, l'estimation des amers peut être faite de manière indépendante. Le système commence par générer $N$ hypothèses (particules) contenant, chacune, les poses du robot et d'associer un ensemble d'amers visuels à chaque particule. Ensuite les poses sont estimées par un filtre particulaire, alors que les amers de la carte sont estimés par un filtre de Kalman permettant de réduire la complexité du calcul.

D'autres travaux ont étudié l'obtention des structures 3D à partir des mou-

vements 2D des images dans un processus hors ligne. Ce sont les méthodes dites de *Structure from Motion* (SfM) basées notamment sur l'ajustement de faisceaux [36, 2]. L'objectif de ces méthodes est de trouver les positions des points 3D et les poses optimales de la caméra qui minimisent les erreurs de reprojection dans l'ensemble complet des images. Contrairement au SLAM visuel, dans les approches SfM un intérêt particulier est porté sur la précision des estimations au détriment du temps de calcul. En principe, les données à traiter sont acquises à priori et par conséquent le besoin de faire face à l'incertitude dans l'estimation en temps réel n'est pas présent. Nister *et al.* [72] ont développé une approche utilisant des "fenêtres glissantes" pour appliquer un ajustement de faisceaux local. Ils parviennent ainsi à réaliser une bonne odométrie visuelle (estimation de pose) en temps réel sans générer la carte globale (qui dérive à long terme). Le PTAM (*Parallel Tracking and Mapping*) proposé par Klein et Murray [54] représente la carte par des amers visuels épars détectés, suivis et reconstruits à l'aide d'une seule caméra. Ce système permet d'accorder plus de temps pour la construction de la carte en exécutant l'ajustement de faisceaux dans un thread en arrière-plan, tandis qu'un autre thread gère le suivi de la caméra à la fréquence des images. La carte globale est construite en utilisant des images clés qui sont échantillonnées spatialement plutôt que temporellement et donc le système devient fiable et rapide dans les longues scènes. Les auteurs mentionnent que la densité des amers peut être augmentée sensiblement à condition que la cartographie soit réalisée à un rythme différent (ou plus lent) que celui de la localisation.

### 2.1.2 Modélisation GraphSLAM

Si l'utilisation du filtrage probabiliste EKF a amené à plusieurs succès dans la résolution du problème du SLAM, de nombreuses techniques modernes de SLAM évitent cette formulation en faveur d'une modélisation du problème à base de graphe pour gagner en efficacité de calcul du processus. En effet, l'estimation conjointe de l'état, à partir des amers et des poses de la caméra, limite le nombre de ces mesures dans un EKF pour un fonctionnement en temps réel à cause du coût calculatoire. En outre, la méthode est peu robuste à l'erreur des mesures et peut dériver rapidement à cause des associations incorrectes de données.

Les techniques à base de graphe résolvent le problème SLAM en construi-

sant un graphe de l'ensemble de la trajectoire du robot et en maintenant des images clés "*Keyframes*" de la scène exploitée qui servent à la détection de la fermeture de boucle lors de passage par des lieux qui ont été déjà visités. La modélisation graphique offre l'avantage d'une représentation plus éparse qui peut être résolue de manière efficace. Dans cette configuration (Figure 2.1, page 39), un noeud du graphe représente la pose de la caméra $x_i$ et enregistre la mesure acquise dans cette pose (correspondant à l'image dans un SLAM visuel), alors que les arêtes $z_{ij}$ (aussi appelées contraintes) correspondent aux informations sur le déplacement estimé entre deux noeuds $i$ et $j$ ou sur la fermeture de boucle. La fonction d'erreur $e_{ij} := e(x_i, x_j)$ mesure la façon dont les poses $x_i$ et

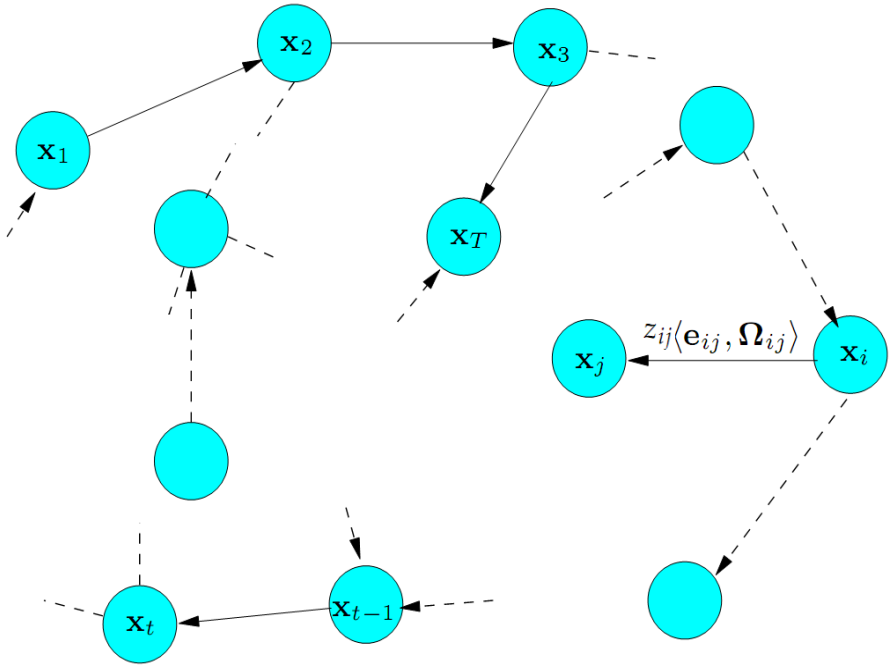



**FIGURE 2.1** – *Représentation du GraphSLAM.*

$x_j$ satisfont la contrainte $z_{ij}$. Elle est égale à 0 lorsque $x_i$ et $x_j$ remplissent parfaitement la contrainte. $\Omega_{ij}$ représente la matrice d'information de la contrainte (l'incertitude de mesure).

Typiquement, les techniques de SLAM basées sur un graphe se divisent en deux blocs : le *Frontend* où le graphe est construit par l'enregistrement des noeuds et des arêtes et le *Backend* qui correspond à l'étape d'optimisation du graphe en minimisant l'erreur globale des transformations estimées entre les noeuds afin de générer une trajectoire correcte.

Lu et Milios ont été les premiers en 1997 à proposer le concept de SLAM

basé graphe dont l'optimisation consiste en un système d'équations pour réduire l'erreur introduite par les contraintes [60]. Cependant, il a fallu attendre plusieurs années pour que cette formulation devienne célèbre en raison de la complexité relativement élevée pour résoudre le problème de minimisation des erreurs du graphe en utilisant des techniques standards.

Le *GraphSLAM* introduit dans le travail de Thrun *et al.* [93] parvient à résoudre les contraintes du graphe par une estimation globale cohérente à travers un processus de linéarisation de ces contraintes généralement non linéaires. Cette linéarisation réduit le système à un problème de minimisation aux moindres carrés couramment résolu en utilisant des techniques classiques d'optimisation telles que la méthode du gradient conjugué.

Dellaert et Kaess [21] ont proposé de résoudre le problème de l'estimation de pose par factorisation d'une matrice d'information éparse, contenant les contraintes entre les poses, dans un SLAM off-line. Après chaque déplacement, cette matrice est factorisée complètement ce qui entraîne une grande charge de calcul.

Strasdat *et al.* ont proposé dans leurs travaux [89, 90] une technique d'optimisation d'un graphe de poses par ajustement de faisceau utilisant les Keyframes. Ces travaux expliquent que les techniques d'optimisation globale sont meilleures que les filtres, car elles permettent une correction efficace de la dérive aux fermetures de boucle. Les auteurs montrent que l'utilisation d'un grand de points d'intérêt par image joue un rôle principal dans l'augmentation de la précision dans l'estimation de mouvement. En outre, l'utilisation des points d'intérêt provenant des images étroitement espacés "Keyframes" permet des traitements plus rapides dans l'optimisation du graphe et par la suite la minimisation des erreurs de l'estimation de poses.

Un tutoriel détaillant les techniques et l'utilisation du GraphSALM a été proposé par Grisetti *et al.* [40].

L'avantage de la modélisation graphique du SLAM est la faculté de trouver une configuration des noeuds qui minimise l'erreur introduite dans les arêtes. Aussi, l'utilisation des Keyframes enregistrées dans différents endroits facilite la détection d'une fermeture de boucle. La minimisation de l'erreur est réalisée à travers le processus de l'optimisation du graphe qui peut être défini comme un problème aux moindres carrés.
Si $\mathbf{X} = (x_1 \dots x_n)$ est le vecteur représentant les poses du graphe, l'objectif de l'optimisation est de trouver la configuration $\mathbf{x}^*$ des noeuds qui minimise l'er-

reur de toutes les transformations estimées. Ce problème peut être résolu par la recherche du minimum d'une fonction de la forme :

$$F(\mathbf{X}) = \sum_{\langle i,j \rangle \in \mathcal{C}} \underbrace{e_{ij}^{\top} \Omega_{ij} e_{ij}}_{F_{ij}} \tag{2.1}$$

$$\mathbf{X}^* = \underset{x}{argmin} F(\mathbf{X}) \tag{2.2}$$

Où $\mathcal{C}$ est l'ensemble des paires d'indices $i, j$ pour lesquels une contrainte $z_{ij}$ existe.
Si une estimation initiale $\widetilde{X}$ de poses est connue, le problème peut être résolu par la linéarisation du système (Section 3.3, Chapitre 3).

### 2.1.3 Représentations de l'environnement

Dans un SLAM, la cartographie est la représentation de l'environnement de la navigation. Le choix de cette représentation est une étape importante puisque la carte permettra à l'utilisateur de planifier ses actions en vue d'atteindre un objectif comme la recherche d'un objet ou la navigation en toute sécurité. Cependant, plusieurs types de représentation peuvent être construites par le robot dans une phase d'initialisation : un modèle dédié "localisation", un modèle dédié "planification de trajectoires", un modèle dédié "détection et reconnaissance d'objets". Une revue détaillée sur les types de cartes existantes peut être trouvée dans [99] (Section 2.2.1 Map Representation) ou notamment dans [34, 64]. Dans cette section, nous revoyons les types des cartes obtenues avec des systèmes SLAM visuel compte tenu de l'intérêt porté à ce point dans le cadre de notre travail de thèse.

#### 2.1.3.1 Cartes Métriques

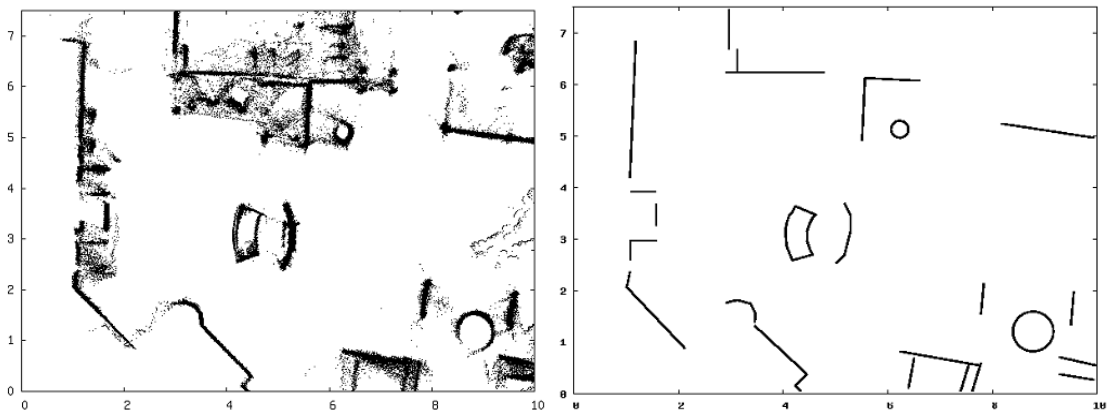
Les cartes métriques décrivent d'une manière précise l'environnement de navigation avec des coordonnées dans un référentiel global de l'espace cartésien. Nous distinguons les représentations métriques les plus populaires dans trois catégories de cartes : directes, basées-primitives et les grilles d'occupation. La méthode directe consiste à reproduire l'environnement à partir des mesures des capteurs par un ensemble de points bruts. Dans ce cas, la carte est construite par la superposition des points bruts sous forme de nuages de points assemblés

[18, 94]. Une telle représentation soufre d'un grand nombre de points redondants qui amène à une consommation inutile de la mémoire. En outre, ce genre de carte n'offre aucune information sémantique sur la scène.
Les grilles d'occupation ont été initialement proposées par Elfes [29] qui subdivise la carte en une grille de cellules. Cette méthode résulte en une représentation discrète de l'environnement. Chaque cellule est caractérisée par une probabilité d'occupation qui marque si elle est occupée, vide ou inexplorée. La taille fixe des cellules définit la résolution de cette représentation. Certains inconvénients peuvent être notés comme l'absence d'un moyen pour vérifier l'incertitude des mesures ainsi que la taille importante de la carte qui contraint l'utilisation à grande échelle.
Finalement, les cartes basées-primitives représentent l'environnement avec la position et l'orientation de primitives visuelles. Ces primitives peuvent être des coins, des bords, des plans ou toute sorte d'objets comme des portes ou des arbres. Ce genre de carte permet une interprétation de haut niveau de la scène, ce qui peut être utile pour la navigation d'un robot si on exploite ces primitives et les relations entre elles comme dans les cartes topologiques (sous-section suivante). Un cas particulier des cartes basées-primitives sont les cartes basées-primitives géométriques [12]. Leur grand avantage est la reconstruction de la scène sous forme de surfaces géométriques continues et compactes (plans, lines, ...). Cependant, la modélisation géométrique des primitives n'est pas toujours possible ce qui limite l'utilisation de ce genre de représentation aux milieux intérieurs.

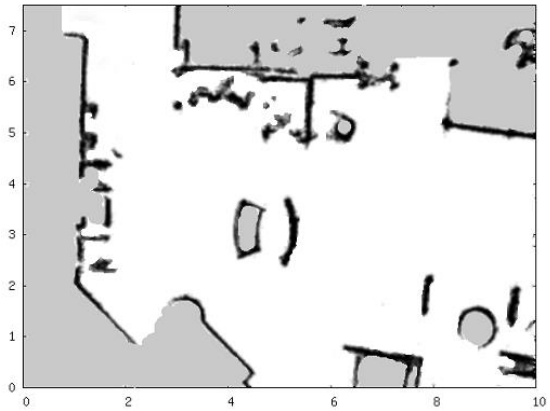
La figure 2.2 présente une carte métrique obtenue par les trois méthodes décrites ci dessus.

#### 2.1.3.2 Cartes Topologiques

Les cartes topologiques illustrent un changement conceptuel majeur dans la représentation de l'environnement. Dans les cartes métriques l'environnement est défini comme un ensemble de coordonnées dans l'espace cartésien. Cependant, les cartes topologiques ne reposent pas sur des mesures métriques et représentent l'environnement plutôt en termes de lieux et de relations entre ces lieux. L'objectif de ces cartes est de caractériser l'apparence d'un lieu d'une manière générale (symbolique) [55, 56]. Elles sont modélisées par une structure de graphe où les noeuds du graphe définissent des endroits précis dans l'en-



***(a)*** *Carte directe* ***(b)*** *Carte basée-primitives*



***(c)*** *Carte de grilles d'occupation*

**FIGURE 2.2 –** *Exemple de cartes métriques. Source[27]*

vironnement (appelés endroits distinctifs) et les arêtes qui représentent les informations sur les déplacements entre les noeuds. Cette modélisation est légère et fournit une représentation compacte de la carte. Il n'existe pas une manière unique pour définir les noeuds et les arêtes lors de la construction d'une carte topologique [33]. Souvent les images acquises lors des déplacements sont comparées entre elles en utilisant des mesures de similarité pour caractériser les lieux. Dans certains systèmes, un noeud est ajouté à chaque acquisition d'une nouvelle image et une arête correspond à la mesure de la similarité entre les images qu'elle relie [10]. Dans d'autres systèmes, la mesure de similarité est réalisée entre les images reçues en temps réel et une base de connaissances construite dans une phase préalable hors-ligne [37].

La figure 2.3 représente un exemple d'une carte topologique avec une représentation métrique de l'environnement associée à des lieux topologiques reliés entre eux par des relations d'adjacence et la représentation purement topolo-

gique correspondante.

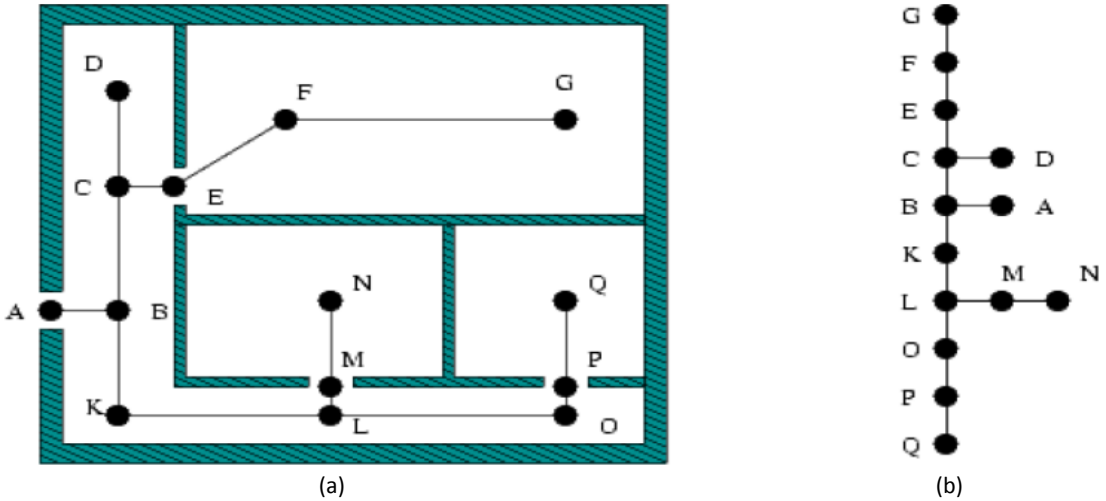



**FIGURE 2.3** – *Exemple d'une carte topologique.* (a) *Projection dans le plan.* (b) *Structure topologique correspondante. (Extrait de [25]).*

La représentation topologique des cartes est généralement appropriée aux grandes surfaces telles que les routes d'une ville. Les avantages d'une telle représentation de la carte sont la production simple, l'organisation logique pour la navigation planifiée, ainsi que la possibilité d'utiliser des algorithmes de graphes standards pour les opérations de planification de haut niveau tels que la recherche du plus court chemin entre deux noeuds non adjacents [22]. Néanmoins, ces cartes présentent quelques inconvénients :

- Les chemins décrits par les arêtes ne sont pas toujours les plus optimaux en termes de déplacements entre deux lieux de la scène.
- Les cartes topologiques ne contiennent aucune description géométrique précise de l'environnement.
- Les algorithmes de la navigation planifiée ne sont pas appropriés pour les cartes topologiques dans les milieux fortement structurés.

#### 2.1.3.3 Cartes Sémantiques

La cartographie sémantique vise à construire des cartes plus riches et plus utiles incluant des informations sémantiques sur l'environnement. Nous distinguons deux types de sémantiques : la reconnaissance d'objet et la reconnaissance des lieux. Un intérêt particulier a été porté récemment sur la détection

des objets dans les environnements intérieurs ainsi qu'à la modélisation et l'étiquetage d'objets en ligne. Il existe ainsi beaucoup de travaux pertinents dans le domaine de la segmentation des environnements en régions et catégories, mais une étude détaillée est au-delà du cadre de cette thèse. À titre d'exemple, dans le travail de Ekvall *et al.* [26] une carte issue du SLAM a été augmentée avec des informations basées sur la reconnaissance d'objets, fournissant ainsi une représentation plus riche de l'environnement dans un scénario de robot de service. Pronobis *et al.* [75] ont étudié quant à eux la catégorisation des lieux pour générer des cartes sémantiques. Ils utilisent à la fois des primitives visuelles extraites dans des images et géométriques obtenues par un laser dans un système de classification des espaces de l'environnement intérieur qui permet l'identification et la reconnaissance des endroits. Bao *et al.* [7] ont représenté une scène comme un ensemble de points, d'objets et de régions dans une approche SfM, en résolvant conjointement, le problème de l'étiquetage et la reconstruction de l'environnement prenant en compte les interactions entre toutes les entités de la scène.
La faiblesse principale de ce genre de carte est le besoin permanent d'une base de connaissance ce qui demande toujours une phase d'apprentissage.

## 2.2 Systèmes RGB-D SLAM

Au début, la cartographie 3D reposait essentiellement sur les scanners laser 3D ou les caméras stéréo. Bien que les scanners laser fournissent des mesures très précises, ils sont coûteux et lourds ce qui empêche leur utilisation pour des produits commerciaux. Les caméras stéréo, nécessitent une forte texture dans la scène pour pouvoir calculer les mesures de distance par triangulation. Le lancement des caméras RGB-D a offert une alternative puissante pour la perception 3D à faible coût. Dans le cadre de cette thèse nous nous sommes intéressés aux systèmes SLAM visuels qui utilisent les capteurs RGB-D pour produire une trajectoire de la caméra et des cartes de la scène sans avoir accès à des objets modélisés à priori ou aux amers visuels. Cette section fournit une étude plus approfondie de l'état actuel des systèmes dits *RGB-D SLAM*. En effet, le capteur RGB-D a suscité un intérêt particulier dans la communauté SLAM et son introduction dans la résolution du problème a donné lieu à plusieurs travaux que nous pouvons classifier en deux grandes familles [42] : Les RGB-D SLAM épars "*sparse*" et *dense*.

### 2.2.1 RGB-D SLAM sparse

Les systèmes SLAM visuels *sparse* basent leur reconstruction de la scène sur l'utilisation de points d'intérêt et leurs correspondances précises. L'idée est de calculer la transformation d'une fraction de points distincts qui sont extraits dans des images successives. Grâce à l'information de la profondeur, la relation géométrique entre ces points peut être trouvée. Typiquement, les modèles produits par ces systèmes résultent en un nuage de points composé de millions de points 3D.

Le premier système RGB-D SLAM, *scientifiquement publié*, est celui de Henry *et al.* [43, 44]. Les auteurs ont développé un pipeline complet de SLAM visuel sparse pour les environnements intérieurs. Ce système est un GraphSLAM basé sur la correspondance des points d'intérêt SIFT. À l'aide d'un RANSAC, ces points d'intérêt servent pour l'alignement entre les images. L'algorithme ICP est utilisé sur les cartes de profondeur disponibles, en premier lieu pour la construction de la carte en alignant les nuages de points, puis pour augmenter la robustesse de l'estimation de pose lorsque le nombre des points d'intérêt disponibles est faible. Pour construire un graphe épars, et donc léger, les noeuds ont été représentés par des *Keyframes* avec leurs images clés correspondantes. Un *Keyframe* contient un sous-ensemble de nuages de points successifs alignés. Une image clé, correspondant à un nouveau *Keyframe*, est ajoutée quand le nombre de correspondances avec l'image clé précédente est inférieur à un seuil. Ainsi, quand une nouvelle image reçue est matchée avec la dernière image clé, le nouveau nuage de points correspondant est aligné au dernier *Keyframe*, sinon l'image est ajoutée comme une nouvelle image clé et son nuage de points correspondant initialise la construction d'un nouveau *Keyframe*. La carte est construite par l'alignement de tous les Keyframes. Les images clés enregistrées dans le graphe sont utilisées pour la recherche de la fermeture de boucle en les comparant avec toute nouvelle image. Pour l'optimisation du graphe, les auteurs ont utilisé l'optimiseur TORO [41] permettant d'assurer la cohérence globale du graphe à l'aide des correspondances des points d'intérêt RGB. Pour construire des surfaces de grande qualité, ils ont utilisé la méthode Surfels (éléments de surface) [74] qui représente chaque point de la carte sous forme d'une surface, généralement un disque, autour de ce point comprenant l'emplacement, l'échelle et l'orientation 3D. Ce dernier traitement est effectué hors ligne sur les nuages de points alignés précédemment. Bien qu'il ait été souhaitable que la représentation de l'environnement soit calculable en ligne, les

auteurs notent que la mise à jour incrémentale de la représentation Surfels est prohibitif puisque les points utilisés dans le calcul d'un Surfel donné dépendent des poses dans lesquelles sont enregistrés et qui continuent de changer au cours de l'optimisation globale. Le système montre de bonnes performances sur des ensembles de données réelles construisant des scènes de bureaux, cependant les mouvements doivent être relativement lents pour assurer la correspondance visuelle et le suivi de la caméra.

Endres *et al.* [31, 32] ont proposé un RGB-D SLAM similaire à [43, 44]. Leur système est devenu très populaire pour les utilisateurs du système d'exploitation robotique (ROS) [1] en raison de sa disponibilité. Malheureusement, l'implémentation ne profite pas des avantages du système ROS (la communication inter-noeuds) car il a été développé en un seul noeud. Comme toutes les approches sparse, le système extrait les points d'intérêt sur les images couleur et utilise les images de profondeur pour trouver leurs coordonnées en 3D. RANSAC est utilisé ensuite pour estimer les transformations entre les points d'intérêt correspondants. L'implémentation et l'optimisation du pose-graphe est effectuée par le framework $G^2o$ [57]. Pour représenter l'environnement, les auteurs ont utilisé des cartes 3D basées grilles d'occupation générées par l'approche OctoMapping [46] basée Octree [62] qui subdivise hiérarchiquement l'espace 3D contenu dans un volume "voxel" en huit sous-volumes jusqu'à la taille minimale de voxel qui détermine la résolution de la carte. Contrairement à plusieurs systèmes RGB-D SLAM, ils ont effectué une évaluation expérimentale détaillée sur un ensemble de données de référence *Benchmark* [91] et ont discuté de nombreux paramètres, tels que le choix des détecteurs et descripteurs des points d'intérêt. Le système est robuste pour plusieurs scénarios tels que les mouvements rapides et offre un bon compromis entre la qualité des poses et le coût de calcul de leurs estimations. La carte basée sur les grilles d'occupation réduit la taille énorme des nuages de points, cependant, comme la représentation Surfels, elle est générée à la fin du processus (hors ligne) car une fois construite elle ne peut pas être mise à jour après l'optimisation du graphe.

Hu *et al.* [47] ont proposé un système du SLAM qui bascule heuristiquement entre l'ajustement de faisceaux RGB-D et RGB en fonction de la disponibilité des données de profondeur. Les cartes sont créées en appliquant l'ajustement de faisceaux sparse sur deux étapes de reprojection par RANSAC suivi de ICP. Théoriquement, l'approche semble robuste au manque d'information

1. wiki.ros.org/rgbdslam

de profondeur dû par exemple aux limitations de distance. D'après leurs expérience, l'algorithme semble bien adapté à l'utilisation dans des environnements à grande échelle. Malheureusement, l'approche n'a pas été évaluée en utilisant des séquences de références et les expériences ne détaillent pas les performances du système notamment la précision des poses estimées et la qualité de la carte.

Maier *et al.* [61] ont étudié les performances de l'ajustement du faisceau incluant des amers visuels dans un graphe avec un système semblable à [31, 32]. Cette approche consiste à optimiser des sous-cartes locales pour ensuite faire l'alignement global de celles-ci. Les auteurs montrent que cette méthode est plus rapide comparée à l'optimisation globale une fois toutes les images disponibles. Dans l'ensemble, ils atteignent une précision comparable au système [31, 32].

Les méthodes RGB-D SLAM décrites ici peuvent être utilisées pour des traitements en ligne. En effet, les approches SLAM sparse sont généralement rapides en raison de l'estimation du mouvement du capteur qui repose sur des points épars. L'implémentation légère peut être intégrée facilement sur des robots mobiles et des petits appareils car les traitements ne demandent pas de grandes capacités calculatoire. Cependant, la qualité des reconstructions est limitée à un ensemble de points 3D dispersés conduisant à de nombreux points redondants dans la carte brute et à l'absence de description sémantique de l'environnement. En outre, une représentation basée points consomme une grande quantité de mémoire (307200 points par nuage de points) et est difficilement exploitable par une tâche de navigation par exemple. Dans le cas des grilles d'occupation ou des Surfels, les cartes sont beaucoup plus légères mais le calcul consomme beaucoup de ressources. Aussi, elles ne peuvent pas être mises à jour de manière efficace en cas de corrections de la trajectoire ce qui impose de les créer hors ligne. Il faut noter un problème commun aux approches *sparse* qui apparaît lorsqu'une nouvelle image ne peut pas être appariée aux images précédentes et qui provoque une remise à zéro ou un arrêt complet du processus.

La figure 2.4 (page 52) montre des exemples de cartes issues des systèmes RGB-D SLAM sparse. La figure ( 2.4a) présente une carte directement générée par l'alignement de plusieurs nuages de points, elle contient plus de 61 millions points 3D. L'Octomap présentée dans la figure ( 2.4b) a été produite dans [32] avec des voxels de taille égale à $1cm^3$. Les voxels occupés sont représentés avec les couleurs de l'image issue de la camera RGB du capteur. Dans la figure ( 2.4c), une région de la carte globale des Surfels générée par [44] est présentée.

### 2.2.2 RGB-D SLAM dense

Contrairement aux approches sparse, les approches RGB-D SLAM dense utilisent exclusivement les données 3D de la caméra RGB-D. Pour estimer le mouvement de la caméra, l'alignement des nuages de points 3D est réalisé sur les cartes de profondeur. Par conséquent, la plupart de ces développements utilisent des processeurs GPGPU afin d'accélérer les calculs coûteux dûs à la grande taille des nuages de points, mais qui peuvent ainsi exploiter pleinement le parallélisme massif des cartes GPU. Le RGB-D SLAM dense a été introduit dans les travaux pionniers réalisés pour Microsoft par Newcombe, Izadi *et al.* appelé KinectFusion [70, 49]. Ils ont proposé un système parallélisé qui peut effectuer un suivi rapide d'une caméra Kinect avec une reconstruction dense de l'environnement. Pour l'estimation de la pose de la caméra en temps réel, ils utilisent un ICP dense entre les nouvelles cartes de profondeur reçues et la carte construite. Cette carte est une représentation volumétrique (voxels) utilisant une fonction de distance signée tronquée (TSDF) [16] où chaque élément stocke une distance signée à la surface la plus proche. Chaque nouvelle carte de profondeur est directement fusionnée dans la représentation de voxels en tenant compte des valeurs précédemment stockées. L'implémentation parallélisée est réalisée sur une carte graphique de haut de gamme permettant ainsi une reconstruction 3D en temps réel. Ce système propose une représentation 3D de l'environnement de très bonne qualité tout en estimant la pose de la caméra en ligne. Cependant, la taille de la grille de voxels a une influence cubique sur l'utilisation de la mémoire, ce qui contraint l'application à des petits espaces. Aussi, le suivi progressif de la caméra rend le système incapable de corriger les erreurs précédentes de l'alignement. Cela impose que le processus soit remis à zéro si la pose de la caméra est perdue. En outre, la dérive est accumulée et aucune technique de fermeture de boucle n'est employée. Un autre problème lié à l'utilisation de la TSDF est qu'elle a besoin d'une mémoire considérable et est stockée dans la GPU ce qui n'est pas donné pour les petits dispositifs (robots, smartphones, ...).

Whelan *et al.* ont réussi à surmonter certaines limitations de KinectFusion dans leurs algorithme appelé Kintinuous [104, 101]. La première amélioration mise en oeuvre est le déplacement de la grille de voxels avec la pose actuelle de la caméra, permettant ainsi la cartographie dans des scènes plus larges. Une fois la limite du volume est atteinte, la TSDF est déplacée, ce qui libère l'espace mémoire pour reconstruire de nouvelles zones de la scène. Les parties qui sont

déplacées en dehors du volume de la reconstruction courant sont triangulées pour former un maillage. Pour faire face à la fermeture de la boucle ils utilisent une technique de déformation des maillages [102]. Un maillage enregistré et désactivé (déplacé hors du volume courant) peut être réactivé et aligné à un nouveau maillage actif si une fermeture de boucle est détectée.

Meilland et Comport [63] proposent une approche combinant les représentations denses basées-voxels avec les avantages de l'utilisation des *Keyframes* dans un RGB-D SLAM. Ils proposent de fusionner plusieurs nuages de points dans un *Keyframe* étendu pour éviter la perte d'informations en raison de la correspondance incomplète des champs de vue des images couleurs et de profondeur, qui se manifeste lors de l'utilisation de l'alignement des nuages de points successifs. Cet algorithme permet une cartographie dense de haute qualité à grande échelle, cependant aucune détection de fermetures de boucles ou correction de la dérive n'est proposée.

Keller et al. (2013) ont présenté un système de fusion basée points pour la cartographie dense en utilisant les Surfels. La cartographie utilisant les Surfels (éléments de surface) facilite la gestion de l'alignement des nuages de points 3D, l'insertion, et la suppression des régions dans la carte, comparés aux maillages structurés comme des triangles avec des sommets (vertex) sur les faces qui sont plus difficiles à manipuler. En outre, les Surfels basés-points économisent la consommation de la mémoire comparés aux méthodes basées-voxels comme KinectFusion. Néanmoins, comme la majorité des approches denses, la correction de la dérive ou la détection de fermeture de boucle ne figurent pas dans cette approche.

En général, les RGB-D SLAM denses permettent une bonne estimation de la pose et une représentation bien détaillée de la scène. Les cartes résultantes, de haute qualité, élargissent le champ d'applications du SLAM à d'autres domaines, tels que la reconnaissance d'objet et la sémantique. Cependant, ces systèmes ont tendance à dériver au fil du temps et rencontrent des difficultés dans les scènes pauvres en matière de structures géométriques. Aussi, les représentations denses de l'environnement sont coûteuses en termes de mémoire et de calcul. Pour surmonter les coûts élevés de calcul, ces approches utilisent des équipements spécialisés tels que les processeurs graphiques de haute performance GPU ce qui peut contraindre la plate-forme utilisée pour réaliser le SLAM.

Des exemples de cartes denses générées par de tels systèmes sont repré-

sentées dans la figure 2.5 (page 53). La figure ( 2.5a) présente la reconstruction d'une scène de bureau réalisée dans [69] avec une résolution de $480 \times 384 \times 384$ voxels dans un volume de délimitation d'environ $1.5 \times 1 \times 1m^3$. Dans la figure ( 2.5b) le bureau a été reconstruit en temps réel par l'approche de fusion "multi-keyframe" proposée dans [63].

### 2.2.3 Évaluations et Comparaison

Après avoir détaillé les deux grands types du RGB-D SLAM (dense et sparse), nous procédons à une évaluation de quelques approches parmi les deux familles en fonction de la disponibilité des implémentations et des résultats publiés. La comparaison est faite sur les ensembles de données (Benchmarks) de l'Université Technique de Munich (TUM) réalisés dans [91] et disponibles sur leur site internet [2]. Ces données, capturées avec une Kinect Microsoft, fournissent des séquences visant différents scénarios d'application tels que des scènes statiques ou dynamiques. La vérité terrain synchronisée de chaque trajectoire du capteur a été obtenue par un système de suivi externe de haute précision de type *Vicon* [1] avec huit caméras de suivi haut débit (100 Hz).

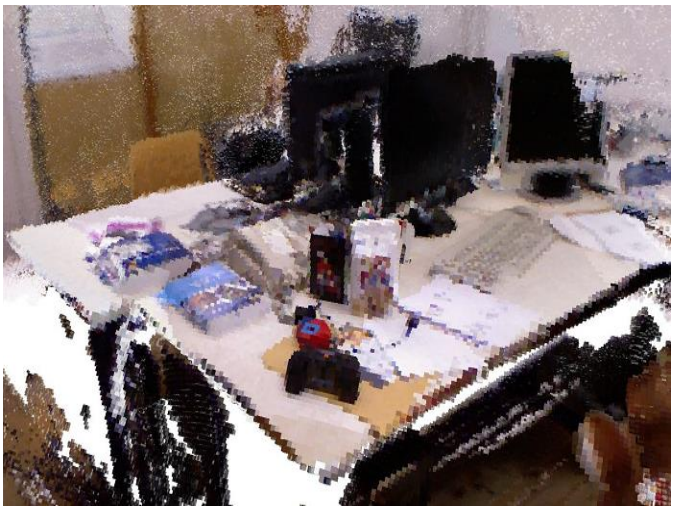
En outre, des outils d'évaluations sont fournis pour mesurer l'erreur de la trajectoire estimée par un système SLAM par rapport à la vérité terrain. Les métriques d'erreurs choisies dans ce travail n'évaluent pas directement la qualité de la carte mais se concentre plutôt sur la reconstruction de la trajectoire. Les auteurs estiment qu'il est raisonnable de supposer que l'erreur de la carte est en général très corrélée avec l'erreur de la trajectoire.

La première métrique est l'ATE (*Absolute Trajectory Error*) qui présente la différence absolue entre la vérité terrain et les poses estimées après alignement. C'est la moyenne des carrés des distances euclidiennes entre les poses mesurées par le SLAM et celles mesurées par le système de tracking extérieur. La seconde, la RPE (Relative Pose Error), mesure la précision locale de la trajectoire sur un intervalle de temps fixe. L'erreur utilisée est la racine carrée moyenne des erreurs (RMSE *Root-Mean-Square Error*).

Pour une trajectoire estimée $\hat{X} = \hat{x}_1 \dots \hat{x}_n$ et la vérité terrain correspondante

2. vision.in.tum.de/data/datasets/rgbd-dataset

***(a)*** *Carte directe (nuage de points)*

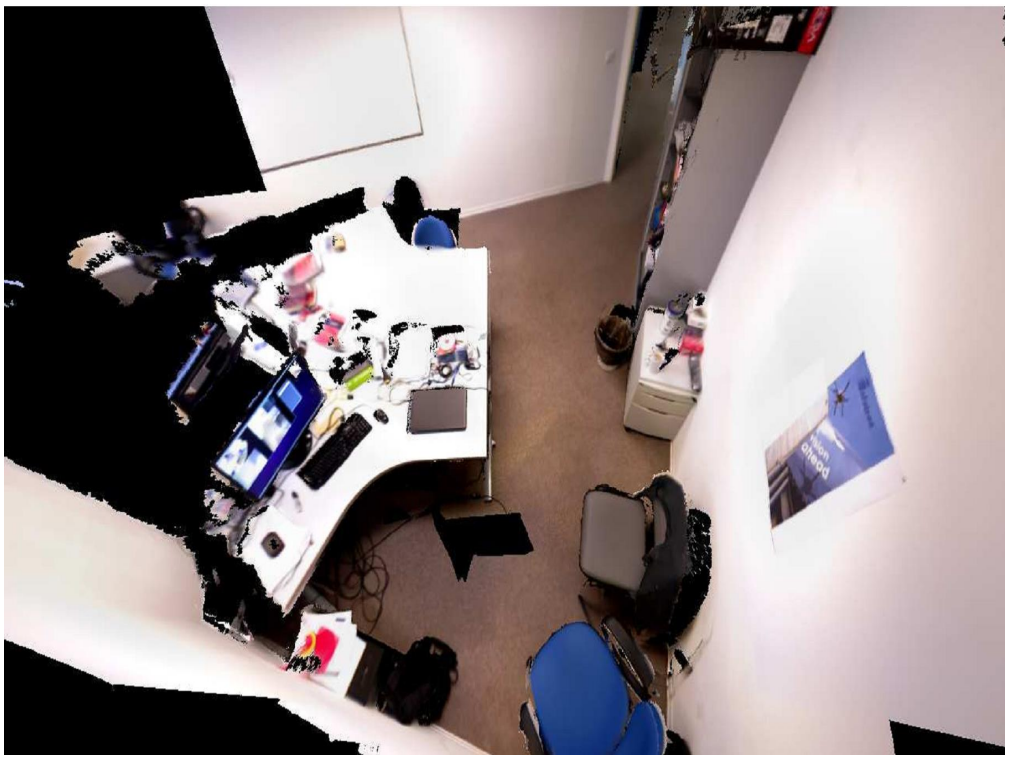


***(b)*** *Octomap [32]*

***(c)*** *Surfels [44]*

**FIGURE 2.4** – *Exemples de cartes RGB-D SLAM sparse.*

***(a)*** *Newcombe [69]*



***(b)*** *Meilland [63]*

**FIGURE 2.5 –** *Exemples de cartes RGB-D SLAM dense.*

$X$, ces erreurs sont données par les formules suivantes sont :

$$e_{ATE-RMSE}(\hat{X}, X) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \parallel trans(\hat{x}_i) - trans(x_i) \parallel^2} \tag{2.3}$$

$$e_{RPE-RMSE}(\hat{X}, X) = \sqrt{\frac{1}{N}\sum_{ij\in\mathcal{T}} trans(\hat{\delta}_{ij} \ominus \delta_{ij})^2 + rot(\hat{\delta}_{ij} \ominus \delta_{ij})^2} \tag{2.4}$$

Où $\hat{\delta}_{ij}$ et $\delta_{ij}$ représentent la transformation relative entre deux poses sur un intervalle $ij$ pour l'estimation et la vérité terrain et $\ominus$ désigne l'opérateur de la différence de mouvement. Un intervalle peut correspondre au nombre de poses (images), temps (secondes), degré de rotation (radians) ou à la distance parcourue (mètres) entre les deux poses. $N$ est le nombre de transformations relatives évaluées et $\mathcal{T}$ est l'ensemble des intervalles sur lesquels les évaluations sont effectuées. Les fonctions "*trans*" et "*rot*" calculent respectivement l'écart de translation et de rotation entre deux trajectoires.

Nous avons comparé les résultats d'estimation de trajectoires de trois approches RGB-D SLAM citées dans l'état de l'art. Pour élargir la comparaison, nous prendrons aussi en compte une approche d'odométrie visuelle *sparse* utilisant une caméra RGB-D [48] dont les résultats sont apparus dans [101]. À la différence du SLAM, l'odométrie visuelle consiste juste à estimer le déplacement d'une caméra à partir d'une séquence d'images. Elle permet ainsi d'estimer la pose de la caméra d'une manière incrémentale en mettant en correspondance les images. La fermeture de boucle peut être recherché à partir des Keyframes enregistrés. Puisqu'il ne s'agit pas d'un SLAM, l'approche d'odométrie visuelle [48] ne sera pas détaillée ici. Toutefois, son implémentation a été publiée récemment sous le nom de *Fovis* [3]. Afin de pouvoir comparer ces systèmes, nous n'affichons que des résultats publiés sur les mêmes séquences parmi l'ensemble des évaluations. Aussi, nous ne comparerons que les valeurs des erreurs médiane et maximale, des trajectoires et poses estimées, car ce sont les données communes dans toutes ces publications respectives.

Une présentation des différentes séquences utilisées dans cette évaluation est donnée dans le tableau 2.1. Dans toutes les séquences la caméra a été portée et déplacée à la main. Les images dans le tableau montrent la nature de l'environnement dans lequel chaque séquence a été réalisée. Ces images ont été capturées, de gauche à droite, dans le sens du déplacement de la caméra dans

3. wiki.ros.org/fovis

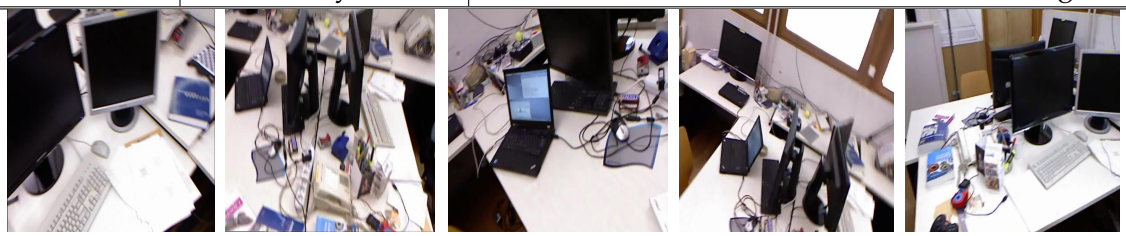
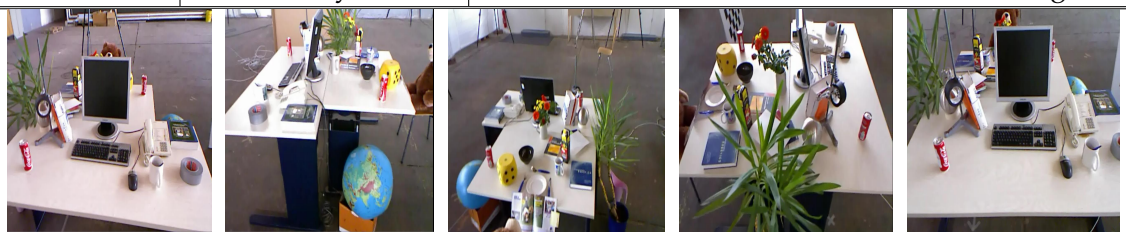
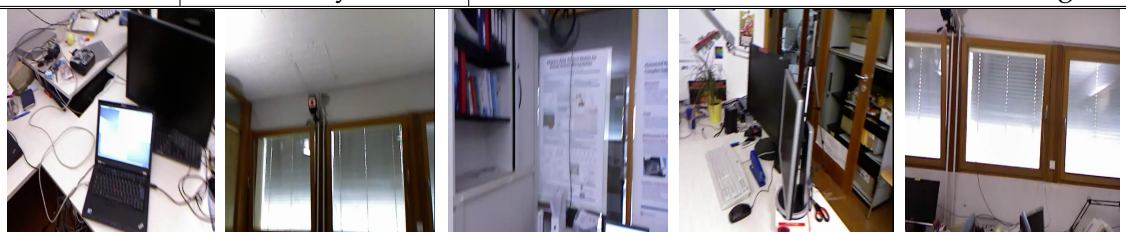
la scène. Les séquences "fr1/desk" et "fr1/room" ont été réalisées dans la même pièce (environnement de bureau). Cependant, dans la première séquence la caméra a été focalisée sur les bureaux présents dans la scène, alors que la seconde séquence présente plusieurs balayages à travers l'ensemble de la pièce jusqu'à fermeture boucle. Cette dernière séquence est bien adaptée pour évaluer dans quelle mesure un système de SLAM peut faire face aux fermetures de boucle. La séquence "fr2/desk" présente également une autre scène de bureau avec deux tables, un écran, clavier,... etc. La Kinect a été déplacée autour des deux tables jusqu'à la fermeture boucle. Dans la séquence "fr2/large no loop" une longue trajectoire a été parcourue à travers le hall industriel sans fermeture de boucle. Le système de suivi externe ne couvre que le début et la fin de la trajectoire, il n'y a pas de vérité terrain pour la partie centrale. Par conséquent, cette séquence peut être utilisée pour vérifier la dérive cumulative d'un SLAM.

Le tableau 2.2 présente une comparaison de l'erreur absolue de la trajectoire (ATE) en mètre. Nous mettons en gras les meilleurs résultats qui représentent le minimum de l'erreur parmi toutes les approches. Les systèmes comparés sont exécutés en temps réel et traitent les données à une fréquence de 30Hz en moyenne. À partir de ces résultats, les algorithmes évalués semblent avoir des performances comparables à l'exception d'un petit ensemble de résultats aberrants tels que les erreurs maximales sur la séquence "fr2/large no loop".

Dans les tableaux 2.3 et 2.4 nous examinons l'erreur relative de la pose pour la translation et la rotation respectivement. Le choix de l'évaluation des poses adjacentes met l'accent sur la dérive locale. Là aussi les résultats sont du même ordre de grandeur sauf pour le RGB-D SLAM de Endres qui montre quelques faiblesses notamment au niveau de la rotation. Ces erreurs sont dues au processus d'optimisation globale du graphe de poses utilisé dans cette approche qui, en minimisant l'erreur globale, augmente l'erreur des estimations locales de poses (Section 3.3, Chapitre 3).

Notons cependant que les approches de Endres *et al.* [32] et *Fovis* utilisent uniquement le CPU, tandis que les deux autres approches nécessitent une GPU puissante. Avec un petit écart en faveur du RGB-D SLAM [32] (par rapport à la médiane dans le tableau 2.2), ce système s'avère avantageux en terme de rapport qualité/coût de mise en œuvre.

**TABLE 2.1** – *Séquences utilisées dans l'évaluation des systèmes RGB-D SLAM. [91]*

| | | |
|---|---|---|
| Séquence fr1/desk | Caractéristiques | Durée : 23.40s, Dimension : 2.42m x 1.34m x 0.66m |
| | Vitesse moyenne de | translation : 0.413m/s, rotation : 23.327deg/s |
|  | | |
| Séquence fr2/desk | Caractéristiques | Durée : 99.36ss, Dimension : 3.90m x 4.13m x 0.57m |
| | Vitesse moyenne de | translation : 0.193m/s, rotation : 6.338deg/s |
|  | | |
| Séquence fr1/room | Caractéristiques | Durée : 48.90s, Dimension : 2.54m x 2.21m x 0.51m |
| | Vitesse moyenne de | translation : 0.334m/s, rotation : 29.882deg/s |
|  | | |
| Séquence fr2/ large no loop | Caractéristiques | Durée : 112.37s, Dimension : 3.63m x 5.07m x 0.20m |
| | Vitesse moyenne de | translation : 0.243m/s, rotation : 15.090deg/s |
|  | | |

### 2.2.4 Synthèse

La synthèse générale des systèmes RGB-D SLAM, tenant en compte des approches précédentes, de leurs implémentations et de leurs résultats, est regroupée dans le tableau 2.5 page 58.

Ce tableau met en évidence plusieurs avantages des RGB-D SLAM sparse comparés à des approches denses qui possèdent plusieurs inconvénients. De

**TABLE 2.2** – *L'erreur absolue de la trajectoire (ATE) des trajectoires estimées (médiane et maximale).*

| Séquences | | fr1/desk | fr2/desk | fr1/room | fr2/large no loop |
|---|---|---|---|---|---|
| Endres [32] | Median | **0.016m** | **0.074m** | **0.076m** | 0.826m |
| | Max | 0.080m | **0.111m** | **0.124m** | 1.574m |
| Whelan [101] | Median | 0.069m | 0.119m | 0.158m | 0.256m |
| | Max | 0.234m | 0.362m | 0.421m | 0.878m |
| Meilland [63] | Median | 0.018m | 0.093m | 0.144m | **0.187m** |
| | Max | **0.066m** | 0.116m | 0.339m | **0.317m** |
| Fovis ([101]) | Median | 0.221m | 0.112m | 0.238m | 0.273m |
| | Max | 0.799m | 0.217m | 0.508m | 0.897m |

**TABLE 2.3** – *L'erreur relative de la pose (RPE) image à image de la translation (médiane et maximale).*

| Séquences | | fr1/desk | fr2/desk | fr1/room | fr2/large no loop |
|---|---|---|---|---|---|
| Endres [32] | Median | 0.0059m | 0.0021m | 0.0044m | 0.0201m |
| | Max | 0.0630m | 0.0685m | 0.1590m | 0.1114m |
| Whelan [101] | Median | 0.0056m | 0.0025m | 0.0045m | 0.0087m |
| | Max | 0.0655m | 0.0108m | 0.0892m | 0.1101m |
| Meilland [63] | Median | **0.0055m** | **0.0015m** | **0.0041m** | **0.0075m** |
| | Max | **0.0390m** | **0.0098m** | **0.0275m** | 0.1620m |
| Fovis ([101]) | Median | 0.0059m | 0.0024m | 0.0051m | 0.0112m |
| | Max | 0.0419m | 0.0122m | 0.0763m | **0.1056m** |

**TABLE 2.4** – *L'erreur relative de la pose (RPE) image à image de la rotation (médiane et maximale).*

| Séquences | | fr1/desk | fr2/desk | fr1/room | fr2/large no loop |
|---|---|---|---|---|---|
| Endres [32] | Median | **0.0060°** | 0.0033° | 0.0057° | 0.0064° |
| | Max | 6.8548° | 2.4685° | 5.1501° | 2.1947° |
| Whelan [101] | Median | 0.0070° | **0.0032°** | **0.0050°** | **0.0039°** |
| | Max | **2.8915°** | 1.3137° | 3.0983° | 2.4962° |
| Meilland [63] | Median | * | * | * | * |
| | Max | * | * | * | * |
| Fovis ([101]) | Median | 0.0071° | 0.0032° | 0.0053° | 0.0043° |
| | Max | 8.0873° | **1.2952°** | **1.9648°** | **1.4295°** |

notre point de vue, fondé sur notre intention de proposer un système qui peut être implémenté sur des appareils à faible coût, les systèmes dense présentent un grand obstacle qui est l'utilisation d'une GPU. Sur des robots mobiles desti-

**TABLE 2.5** – *Synthèse générale des systèmes RGB-D SLAM.*

| | RGB-D SLAM Dense | RGB-D SLAM Sparse |
|---|---|---|
| Temps réel | + | + |
| Robustesse | + | + |
| Qualité de la carte | ++ | – |
| Coût de l'implémentation | – | + |
| Fermeture de boucle | - | + |
| Application à grande échelle | - | + |
| Correction de l'erreur | - | ++ |
| Défaillance due au manque de structure | - | * |
| Défaillance due au manque de texture | * | - |
| Examen des données de profondeur | - | - |

nés à des tâches basiques, comme le Turtlebot [4] par exemple, cette solution n'est pas envisageable. En outre, dans ces systèmes le suivi progressif de la caméra empêche la correction des erreurs introduites dans les estimations précédentes. Cependant, la construction de la carte en temps réel présente un problème dans un RGB-D SLAM sparse : Comme nous l'avons vu précédemment, les cartes de Surfels ou de grilles d'occupation (Octomap) sont construites dans un processus hors ligne après l'optimisation globale du graphe de poses ce qui peut contraindre l'utilisation d'un robot mobile nécessitant une carte lors de ses déplacements. Par conséquent, pour pouvoir construire la carte en ligne, un alignement des nuages de points bruts est nécessaire. La carte globale résultante est un grand nuage de points contenant des millions de points 3D dispersés. Le chevauchement des champs de vue de la caméra, lors des déplacements, produit de nombreux points redondants dans la carte. Avec la grande taille des nuages de points (307200 points) cette carte devient lourde et consomme beaucoup de mémoire. De plus, en raison des erreurs locales de l'alignement, plusieurs parties de la scène reconstruite perdent leurs apparences géométriques réelles (tables, murs,...). Tout ces défauts rendent cette carte basée points difficile à exploiter ce qui peut gêner le bon fonctionnement des robots dans les milieux intérieurs. Par ailleurs, dans les deux approches, sparse et dense, aucune étape d'évaluation des valeurs de profondeur acquises par la caméra n'est effectuée. Toute erreur introduite par ces données, peut engendrer des erreurs graves dans l'estimation de poses et par conséquent dans la carte résultante des alignements de nuages de points.

4. www.turtlebot.com

Ainsi, l'objectif de cette thèse est de proposer des améliorations pertinentes aux problèmes détaillés ci dessus dans un RGB-D SLAM sparse. En effet, le choix de l'utilisation d'un système sparse est raisonnable car son implémentation, basée sur les points d'intérêt, n'exige pas l'utilisation d'une GPU, et est envisageable pour des appareils de petites capacités calculatoire (robot mobile, smartphone,...) Cependant, comme la carte résultante peut être lourde, notre objectif sera de proposer une représentation permettant de réduire la taille de cette carte en se basant sur les primitives existantes dans l'environnement. Étant donné que dans les milieux intérieurs, les régions planaires sont courantes (portes, bureaux, murs,...), nous avons choisi d'intégrer leurs modèles géométriques directement dans la carte au lieu d'utiliser les points 3D bruts. Une telle solution permettra de générer des cartes de taille réduite, contenant des modèles géométriques simples, et qui peuvent être exploitées facilement par des applications de navigation mobile ou de réalité augmentée. En outre, les modèles géométriques extraits dans une scène peuvent fournir un moyen de vérification des valeurs de profondeurs lors d'une correspondance 2D-3D.

## 2.3 Conclusion

Dans ce chapitre nous avons effectué une revue de l'état de l'art du SLAM visuel. L'objectif de cet état de l'art était de présenter les méthodes et techniques basées sur la vision pour la résolution du problème du SLAM. Nous avons mis en avant la modélisation graphique du problème dans laquelle le SLAM est représenté par un graphe de poses. Les noeuds du graphe correspondent aux poses de la caméra et les arêtes présentent les transformations entre ces poses. Cette modélisation légère du SLAM permet la minimisation des erreurs de l'estimation de la pose lors de l'optimisation du graphe, et la détection des éventuelles fermetures de boucle. Toutefois, un intérêt particulier a été porté sur deux éléments caractéristiques : les types de représentations de l'environnement (cartographie) dans un SLAM et les systèmes SLAM récents utilisant un capteur RGB-D (les RGB-D SLAM). Depuis les cartes métriques jusqu'à la représentation sémantique, le choix de la cartographie joue un rôle primordial dans un SLAM et définit le champ de l'utilisation. Nous avons vu que le RGB-D SLAM dense permet d'enrichir la connaissance de la scène qu'il représente comme des surfaces denses et continues. Dans les systèmes où la représentation de l'environnement est un nuage de points unique, résultant de l'alignement de

plusieurs nuages de points, la carte résultante devient difficilement utilisable. Cependant, les solutions dense sont coûteuses et présentent plus d'inconvénients que d'avantages, notamment le problème de propagation de l'erreur des estimations de poses et la contrainte de l'utilisation dans des petits espaces. Le travail de synthèse réalisé nous a permis de définir les choix pertinents dans le cadre de cette thèse. Ainsi, une technique RGB-D SLAM sparse sera adoptée puisqu'elle présente un atout essentiel pour nous, à savoir, son implémentation légère et généralement robuste. Pour améliorer les cartes construites par une telle technique, nous proposons de développer une solution utilisant des primitives planaires fréquentes dans les milieux intérieurs. Cette solution permettra de donner plus de sémantique à la carte vu que les représentations sparse actuelles échouent à ce niveau. Cette approche sera détaillée dans le chapitre suivant.

# Chapitre 3

# RGB-D SLAM basé-plans

## Sommaire


**3.1 Introduction . . . . . 62**

**3.2 Travaux connexes . . . . . 62**

**3.3 RGB-D SLAM basé-plans . . . . . 65**

3.3.1 Détection de plans . . . . . 70

3.3.2 Points d'intérêt Planaires . . . . . 72

3.3.3 Cartes 3D basées-Plans . . . . . 74

**3.4 Conclusion . . . . . 80**


Dans ce chapitre nous allons présenter la contribution majeure de la thèse. Une méthode intégrant les plans 3D détectés dans l'environnement lors des traitements réalisés par un système RGB-D SLAM . Tout d'abord nous introduirons l'intérêt et les motivations de l'utilisation des structures planaires dans la résolution du problème du SLAM en intérieur. Ensuite, nous exposerons les systèmes RGB-D SLAM récents qui intègrent également les primitives planaires dans la résolution du problème. Nous proposerons ensuite notre méthode RGB-D SLAM basée sur l'utilisation des plans 3D détectés dans une scène pour estimer la trajectoire de la caméra et la reconstruction de l'environnement. Nous mettrons en avant les techniques que nous avons développées pour créer une représentation planaire de l'environnement dédiée à une utilisation dans des scènes d'intérieur structurées, et pour générer des points d'intérêt 3D planaires utilisés pour l'estimation de poses de la caméra.

## 3.1 Introduction

Le SLAM est un problème complexe où le robot est obligé de connaître en permanence sa localisation par rapport à la partie cartographiée de l'environnement. Indépendamment de la complexité du problème, un algorithme de SLAM est éventuellement nécessaire dans des robots mobiles simples embarquant une puissance de traitement limitée. De tels robots sont généralement destinés à une utilisation en intérieur comme par exemple pour les tâches d'entretien ou bien pour accompagner des personnes âgées. Cependement, peu d'efforts de recherche ont été fait pour le développement d'algorithmes SLAM 3D en intérieur qui respectent les contraintes de ce genre de robots, notamment dans les RGB-D SLAM détaillés dans le deuxième chapitre (section 2.2). Si les approches "sparse", basées-points d'intérêt, arrivent à proposer des algorithmes relativement légers pour l'estimation de la pose, ce n'est pas le cas pour la cartographie qui reste un problème loin d'être résolu vu la taille et la complexité des cartes produites. Par ailleurs, la navigation autonome, qui représente une fonction fondamentale pour un robot mobile, nécessite la disponibilité en temps réel, d'une carte compacte de l'environnement. Par conséquent, proposer des solutions économiques est indispensable pour de nombreuses applications actuelles mais aussi pour de potentielles nouvelles applications. Une solution judicieuse serait alors d'utiliser des primitives dans la carte à la place des nuages de points complets. En effet, à l'opposé de l'utilisation des points bruts, l'évolution naturelle du SLAM visuel basé sur les points d'intérêt consiste à introduire des primitives qui peuvent être définies par leurs modélisations géométriques ce qui rend la carte plus légère. Comme ils sont très fréquents dans les environnements intérieurs et facilement déduits des nuages de points, les plans 3D représentent un choix pertinent. Ainsi la proposition d'une nouvelle méthode permettant de réaliser un SLAM léger avec une carte compacte basée sur des primitives planaires constitue l'apport principal de cette thèse.

## 3.2 Travaux connexes

L'utilisation des plans dans la résolution du problème de SLAM n'est pas une nouvelle idée et a déjà fait l'objet de plusieurs recherches. Parmi les premiers travaux intégrants les plans dans le SLAM, on peut citer le système de Weingarten et Siegwart [100] : L'algorithme extrait les plans directement à par-

tir des nuages de points générés par un scanner laser 2D, et les utilise comme des primitives planaires nommées ici "*Planar features*" dans le cadre d'un EKF SLAM. La carte est cependant représentée par un grand nuage de points 3D rassemblant tous les points planaires détectés. Chaque plan est délimité par une enveloppe convexe qui peut être étendue progressivement à mesure que de nouvelles portions de plans sont observées. Chekhlov *et al.* [13] et Gee *et al.* [39] ont proposé un système similaire dans leur SLAM monoculaire pour une application de Réalité Augmentée.

Avec l'apparition des caméras RGB-D, l'utilisation des modèles géométriques et la reconnaissance d'objets, notamment les plans, a connu une croissance importante. En effet, les cartes de profondeur permettent la détection des structures planaires sans faire appel à la texture utilisée habituellement. Un autre facteur qui a contribué à cette explosion est la diversité des approches de segmentation des nuages de points 3D [80, 45, 97, 83]. La majorité de ces approches montrent que les surfaces planes peuvent être extraites très efficacement à partir des nuages de points et indirectement les objets se trouvant sur ces surfaces planes.

Trevor *et al.* [96], par exemple, combinent une Kinect avec un scanner laser 2D pour générer à la fois des lignes et des plans, utilisés comme primitives, dans une représentation sous forme de graphe. La carte est construite à partir des primitives géométriques, segments et plans, détectées.

Dou *et al.* [23] ont amélioré la qualité de la reconstruction 3D en intérieur par rapport à l'existant via un SLAM avec ajustement de faisceaux qui combine les erreurs d'alignement des surfaces planes avec les erreurs de reprojection des points d'intérêt pour estimer la pose. Dans la carte, les plans sont fusionnés grâce à un critère de distance entre les points limites des régions planaires détectées. Le système réalise l'extraction des plans en 3 secondes environs alors que l'optimisation globale prend quelques minutes, ce qui le rend inapproprié pour une application en ligne.

Taguchi *et al.* [92] ont également combiné les points d'intérêt et les plans 3D dans un système d'ajustement de faisceau avec une caméra RGB-D. Ils utilisent les combinaisons de correspondances plan-à-plan, point-à-point ou point-à-plan pour estimer la pose. Leur système montre une représentation compacte de l'environnement mais un suivi lent de la caméra (2 à 3 images par seconde). Ce travail a été étendu par Ataer-Cansizoglu *et al.* [6] pour trouver les corres-

pondances des points et des plans à l'aide de la prédiction du mouvement de la caméra en utilisant le flot optique pour prédire les emplacements de pixels suivis dans les images. Toutefois, l'hypothèse d'un mouvement à vitesse constante utilisée pour prédire la pose reste difficile à satisfaire lors d'une utilisation en conditions réelles (mouvement naturel par exemple).

Salas-Moreno *et al.* [82] ont étendu le RGB-D SLAM dense de Keller *et al.* [52] pour respecter la planarité dans la reconstruction dense de l'environnement d'une application de réalité augmentée. Les surfels appartenant à la même structure planaire (plan) sont étiquetés et des contraintes de planarité sont appliqués dans l'estimation de la pose.

Dans un travail récent, G. Xiang et T. Zhang [38] ont proposé un RGB-D SLAM basé sur des primitives planaires. Ils commencent par détecter les plans dans les nuages de points, puis à partir de chaque plan 3D détecté ils génèrent une image 2D correspondante et en extraient des points d'intérêt SIFT "*Features*" 2D. Ces *features* sont ensuite reprojetés dans l'image de profondeur pour générer des points d'intérêt 3D utilisés pour estimer le mouvement de la caméra grace à l'algorithme ICP. La reconstruction de l'environnement reste cependant sous forme d'un nuage de points épars qui n'offre aucune information sur la planarité de la scène.

Plus récemment, Whelan *et al.* [103] ont développé une extension de leurs travaux antérieurs [104, 101], qui effectue une segmentation planaire incrémentale des nuages de points pour générer un maillage global dense, contenant les surfaces planes et non planes qui sont construites parallèlement. Cette reconstruction de l'environnement offre une information détaillée de la scène mais présente les même inconvénients que les cartes denses, à savoir une consommation importante de la mémoire et l'utilisation d'une GPU.

Dans le travail de Kaess [51] la représentation des plans infinis est réduite à une représentation de points dans une sphère unité $\mathbb{S}3$. Cela permet de paramétrer un plan sous la forme d'un quaternion unitaire représentant le modèle d'un plan et ainsi de formuler le problème de l'estimation de la pose, en utilisant les plans, comme une optimisation aux moindres carrés d'un graphe de plans infinis. L'avantage de cette représentation minimale est qu'elle permet une optimisation plus rapide du graphe. Cependant, la représentation de l'environnement en terme de modèles 3D est limitée à un nuage de points.

Dans nos travaux, nous avons également utilisé des primitives 3D planaires

pour développer un système RGB-D SLAM épars basé-plans (Section suivante). Contrairement aux travaux antérieurs, notre objectif est de générer une représentation légère et compacte de l'environnement avec une bonne estimation de poses. Cependant, le type de la carte résultante (grille d'occupation, topologique,...) n'a pas fait l'objet de nos travaux. Nous avons étudié le problème SLAM d'une manière générale : indépendamment de l'application robotique dans laquelle la carte résultante peut être exploitée (planification, reconnaissance d'objet,...).

## 3.3 RGB-D SLAM basé-plans

Le schéma général de notre approche est représenté dans la figure 3.1. Le coeur de cette approche est basé sur le RGB-D SLAM sparse de Endres *et al.* [31, 32] (figure 3.2). Nous avons utilisé leur implémentation open source disponible sous ROS [1]. Le corps de ce système a été conservé, cependant nous l'avons adapté afin de créer des cartes basées sur des primitives planaires (Section 3.3.3, page 74) au lieu de cartes de points. En outre, nous avons enchéri l'estimation de poses, basée sur les points d'intérêt 3D bruts, par une étape de régularisation de ces points afin de diminuer l'influence du bruit des images de profondeur. Nous avons utilisé les plans détectés pour générer des "Points d'intérêt 3D Planaires" (Section 3.3.2, page 72) qui vont remplacer les points 3D bruts dans l'estimation de poses. Notre contribution, par rapport à l'approche de base, est représentée par les blocs de couleur verte. La description de ces blocs sera détaillée dans les sous-sections suivantes.

Comme mentionné auparavant, l'approche d'origine est un GraphSLAM utilisant les données RGB-D. C'est un SLAM sparse basé sur les points d'intérêt.
Le rôle du FontEnd est l'acquisition des données de capteurs (images RGB et de profondeur) et l'association de ces données (mise en correspondance) pour estimer les relations géométriques entre les déplacements du capteur ainsi que pour détecter les éventuelles fermetures de boucle lors du passage par des endroits visités précédemment. Pour estimer une transformation entre deux poses de la caméra, le système commence par la détection et la description des points d'intérêt 2D dans les images acquises en utilisant un algorithme de dé-

1. http ://wiki.ros.org/rgbdslam

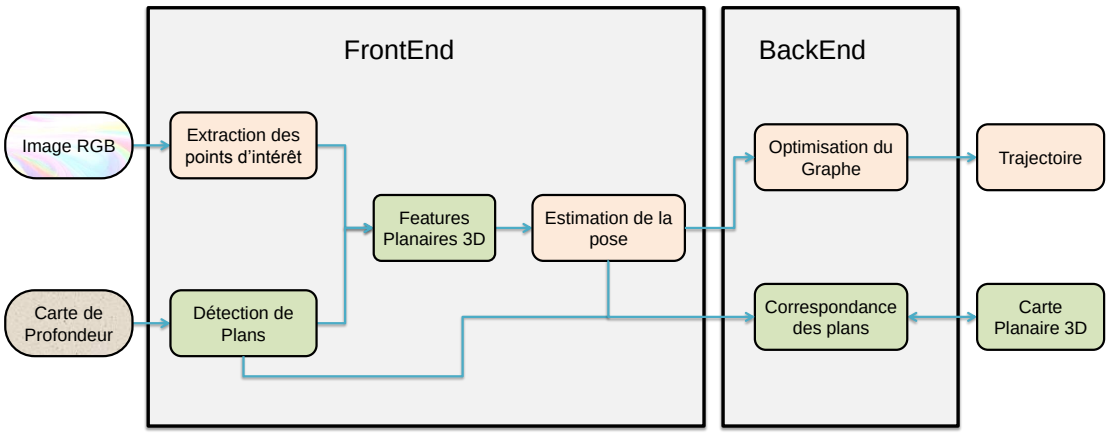



**FIGURE 3.1** – *Présentation de notre système RGB-D SLAM.*

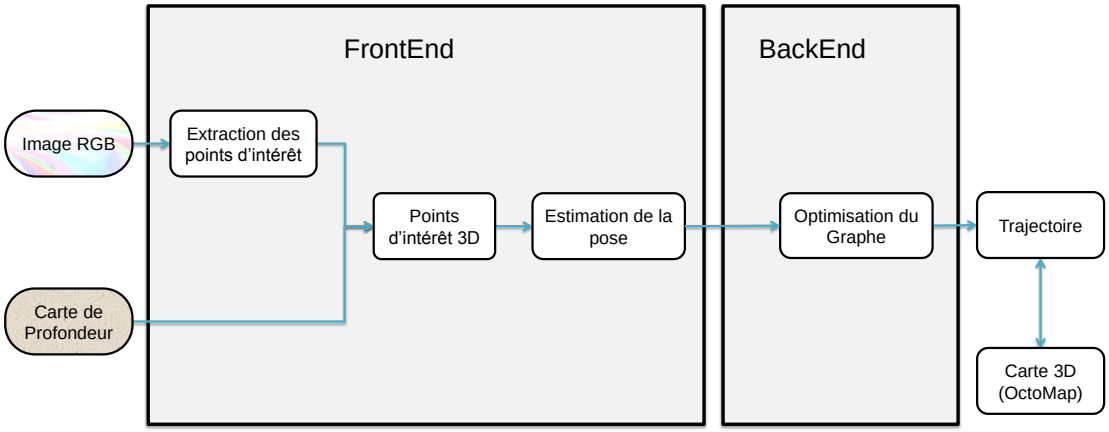



**FIGURE 3.2** – *Système RGB-D SLAM de Endres* et al. *[31, 32].*

tection/description de points d'intérêt. Le système d'origine propose d'utiliser l'une des implémentations des algorithmes SIFT, SURF ou ORB à partir de la bibliothèque OpenCV. Les points d'intérêt 2D extraits sont ensuite projetés en points 3D grâce aux données de la profondeur pour construire des points d'intérêt 3D. Ensuite, les points d'intérêt 2D de chaque nouvelle image sont mis en correspondance avec ceux de l'image précédente en utilisant un algorithme de matching (FLANN ou BruteForce). Un seuil de points correspondants est exigé, à savoir 3 appariements, qui représente le nombre minimal de points à partir duquel une transformation rigide peut être estimée. L'algorithme RANSAC est utilisé lors de l'estimation de la pose afin d'augmenter la robustesse face aux faux appariements. Pour estimer une transformation (Rotation $\mathbf{R}$ et translation $\mathbf{t}$), la technique d'estimation au sens des moindre carrés détaillée au

premier chapitre (section 1.3.2.2, page 28) est appliquée sur les points d'intérêt 3D correspondants aux points d'intérêt 2D appariés dans deux images. À chaque itération de l'algorithme RANSAC, la transformation est estimée et raffinée en utilisant les inliers. Un point d'intérêt 3D est considéré inlier si il est reprojeté, par la transformation estimée, à une distance inférieure à un seuil ($\theta = 3cm$) par rapport à son correspondant. Ce seuil est réduit à chaque itération de l'algorithme RANSAC comme il a été proposé par Chum *et al.* [15]. Ainsi, un nouveau noeud est ajouté au graphe lorsqu'une transformation existe entre deux nuages de points. Une arête est ajoutée entre le nouveau noeud et le précédent pour représenter cette transformation dans le graphe.

Le BackEnd, en fonction des résultats du FrontEnd, est responsable de la construction et de l'optimisation du graphe de pose afin de minimiser l'erreur globale de la trajectoire de la caméra. Nous avons vu dans le deuxième chapitre (section 2.1.2 page 38) que l'optimisation est une étape très importante qui sert à minimiser l'erreur globale des poses estimées en minimisant la fonction :

$$F(\mathbf{X}) = \sum_{\langle i,j \rangle \in \mathcal{C}} \underbrace{e_{ij}^{\top} \Omega_{ij} e_{ij}}_{F_{ij}} \tag{3.1}$$

Où $\Omega_{ij}$ représente la qualité de l'estimation mesurée entre deux poses $i$ et $j$. Elle correspond au pourcentage d'inliers satisfaisant la transformation estimée parmi tout l'ensemble des appariements. La fonction de l'erreur $e_{ij}$ représente la somme des distances de Mahalanobis entre les paires de points d'intérêt 3D appariés après l'application de cette transformation. L'optimisation globale du graphe est déclenchée à chaque ajout d'un nouveau noeud au graphe de façon à minimiser cette erreur globale en permanence. Le problème revient à trouver la disposition des noeuds (poses) du graphe $\mathbf{X}^*$ qui minimise l'erreur globale des distances de Mahalanobis entre les points d'intérêt 3D correspondants.

$$\mathbf{X}^* = \underset{x}{argmin} F(\mathbf{X}) \tag{3.2}$$

Le framework $G^2o$ [57] adopté par l'approche d'origine, utilise une technique des moindres carrés pour résoudre le problème, à partir de l'estimation initiale de poses $\widetilde{\mathbf{X}}$. La fonction d'erreur globale est approchée en utilisant le développement de Taylor du premier ordre autour de cette estimation initiale ($\widetilde{\mathbf{X}}$) des poses.
Localement, ce développement de Taylor de la fonction d'erreur ($e_{ij} := e_{ij}(x_i, x_j) :=$

$e_{ij}(x)$) est donné par :

$$\begin{aligned} e_{ij}(\widetilde{x}_i + \Delta x_i, \widetilde{x}_j + \Delta x_j) &= e_{ij}(\widetilde{x} + \Delta x) \\ &\simeq e_{ij}(\widetilde{x}) + \mathbf{J}_{ij}\Delta x \end{aligned} \tag{3.3}$$

Où $\mathbf{J}_{ij}$ représente la Jacobienne de $e_{ij}(x)$ calculée en $\widetilde{x}$.
En remplaçant dans l'équation de l'erreur globale (3.1), la linéarisation du système est réalisée par :

$$\begin{aligned} F(\widetilde{\mathbf{X}} + \Delta\mathbf{X}) &= \sum_{\langle i,j\rangle \in \mathcal{C}} e_{ij}(\widetilde{x} + \Delta x)^\top \Omega_{ij} e_{ij}(\widetilde{x} + \Delta x) \\ &\simeq \sum_{\langle i,j\rangle \in \mathcal{C}} (e_{ij}(\widetilde{x}) + \mathbf{J}_{ij}\Delta x)^\top \Omega_{ij} (e_{ij}(\widetilde{x}) + \mathbf{J}_{ij}\Delta x) \\ &= \sum_{\langle i,j\rangle \in \mathcal{C}} e_{ij}(\widetilde{x})^\top \Omega_{ij} e_{ij}(\widetilde{x}) + 2e_{ij}(\widetilde{x})^\top \Omega_{ij}\mathbf{J}_{ij}\Delta x + \Delta x^\top \mathbf{J}_{ij}^\top \Omega_{ij}\mathbf{J}_{ij}\Delta x \\ &= \sum_{\langle i,j\rangle \in \mathcal{C}} c_{ij} + 2b_{ij}\Delta x + \Delta x^\top H_{ij}\Delta x \\ &= c + 2b\Delta X + \Delta X^\top H \Delta X \end{aligned} \tag{3.4}$$

Avec $c = \sum c_{ij}$, $b = \sum b_{ij}$ et $H = \sum H_{ij}$. L'erreur globale est minimisée en résolvant le système linéaire :

$$\Delta X^* = -b/H \tag{3.5}$$

Finalement, la solution est obtenue en incrémentant l'estimation initiale $\widetilde{X}$ par $\Delta X^*$ :

$$X^* = \widetilde{X} + \Delta X^* \tag{3.6}$$

La recherche de la solution optimale peut être réalisée en utilisant des algorithmes populaires comme Gauss-Newton ou Levenberg-Marquardt.

Concrètement, l'optimisation du graphe consiste à minimiser globalement les erreurs entre les points d'intérêt 3D appariés, entre chaque paire de poses $x_i$ et $x_j$, en modifiant les transformations enregistrées entre ces poses par la solution estimée. Cela revient à trouver la configuration du graphe qui présente la meilleure qualité globale de l'estimation ($\Omega = \sum \Omega_{ij}$), qui correspond au plus grand pourcentage d'inliers satisfaisant les transformations. Ainsi, après chaque optimisation du graphe, les transformations estimées sont mises à jour par les nouvelles transformations qui minimisent l'erreur globale des poses.

**TABLE 3.1** – *Comparaison des erreurs métriques du RGB-D SLAM avec et sans optimisation du graphe.*

| Séquence | Type de l'erreur | Avec Optimisation | | Sans Optimisation | |
|---|---|---|---|---|---|
| | | Median | Max | Median | Max |
| fr1/desk | ATE | **0.016m** | **0.08m** | 0.064m | 0.133m |
| | RPE Translation | 0.006m | 0.063m | 0.007m | 0.099m |
| | RPE Rotation | 0.006° | 6.854° | **0.004°** | **2.584°** |
| fr1/room | ATE | **0.076m** | **0.124m** | 0.243m | 0.548m |
| | RPE Translation | 0.004m | 0.159m | 0.005m | 0.044m |
| | RPE Rotation | 0.006° | 5.150° | **0.005°** | **2.054°** |
| fr2/large no loop | ATE | **0.826m** | **1.574m** | 0.998m | 3.022m |
| | RPE Translation | 0.020m | 0.111m | 0.019m | 0.146m |
| | RPE Rotation | 0.006° | 2.194° | **0.005°** | **1.145°** |

Par la suite, la reconstruction de l'environnement dans le RGB-D SLAM natif (de Endres *et al.*) est réalisée soit en ligne, par l'alignement de chaque nouveau nuage de points 3D, enregistré dans une nouvelle pose, à la carte globale, soit dans une phase hors ligne en générant des cartes 3D basées grilles d'occupation par l'approche OctoMapping [46].

Dans notre approche, nous avons conservé la technique d'optimisation globale, du graphe de poses, pour l'estimation des trajectoires de la caméra, vu qu'elle a montré de bons résultats lors de l'évaluation du RGB-D SLAM d'origine dans le deuxième chapitre (Tableau 2.2 page 57). Toutefois, lors de la reconstruction de notre carte nous n'avons pas utilisé la trajectoire optimale estimée par cette technique. En effet, même si elle est très utile pour générer des trajectoires globalement précises, l'optimisation globale du graphe peut engendrer des erreurs dans les transformations locales entre les poses, notamment au niveau de la rotation. Le tableau 3.1 présente une comparaison entre les erreurs des estimations globales et locales de poses pour des trajectoires générées, avec et sans optimisation globale du graphe, par le RGB-D SLAM natif. Rappelons que l'erreur globale de la trajectoire, l'ATE (*Absolute Trajectory Error*), représente la différence absolue entre la vérité terrain mesurée par le système de tracking extérieur et les poses estimées par le système. Cependant, l'erreur locale, la RPE (Relative Pose Error), mesure la précision des estimations pose par pose. Les séquences sont les mêmes que celles utilisées dans l'évaluation du deuxième chapitre (Tableau 56). Elles présentent des scènes d'intérieur réalisées dans un environnement de bureau ou un hall industriel.

Les résultats montrent que, pour les trajectoires optimisées, les erreurs absolues sont généralement minimales. Cependant, dans l'estimation locale de pose, particulièrement de la rotation, les erreurs sont plus importantes contrairement aux trajectoires générées sans optimisation globale du graphe. Nous pouvons en conclure que le problème des erreurs locales élevées est dû à l'étape d'optimisation globale du graphe. En effet, comme les transformations estimées ne sont pas linéaires, la résolution du problème nécessite l'étape de linéarisation détaillée auparavant. Cette étape résulte en une simplification du système permettant la recherche de la solution qui minimise l'erreur globale, sans prendre en considération les erreurs locales entre les poses, ce qui peut conduire à une dégradation de la qualité de la rotation dans les résultats. Les auteurs du RGB-D SLAM ne mettent pas en avant ces résultats, comme ils l'annoncent, leur intérêt est de générer une trajectoire correcte en évaluant l'ATE. Cependant, dans notre approche la précision de l'orientation de la pose estimée est très importante. En effet, les plans détectés à chaque pose vont servir pour construire la carte globale (Section 3.3.3, page 74). Dans le repère global (de la carte), la comparaison des orientations par exemple de deux plans, détectés dans deux poses différentes et transformés dans la carte globale, permettra de savoir si ces deux plans correspondent dans le monde réel, d'où l'intérêt d'avoir une estimation précise de la rotation.

### 3.3.1 Détection de plans

Le pseudo-code de la technique de détection de plans 3D élaborée, est donné dans l'algorithme 2. Dans notre système, les régions planaires ont été détectées directement à partir des cartes de profondeur. La librairie PCL [81] offre un outil de détection utilisant l'algorithme RANSAC qui permet d'extraire l'ensemble principal de points coplanaires dans un nuage de points. Comme notre objectif est de construire une carte qui contient tout les plans existants dans la scène, nous effectuons un RANSAC itératif sur chaque carte de profondeur reçue afin de détecter les $k$ plans principaux. À chaque itération, nous enregistrons les paramètres du plan détecté et supprimons les points lui appartenant du nuage de points, puis nous continuons à extraire d'autres plans tant que la taille, nombre $N$ de points 3D, du nouveau plan est significative.

Chaque région planaire $\pi$, détectée dans un nuage de points, est représentée par son modèle du plan $(n_x, n_y, n_z, d)^\top$ où $n^\top(n_x, n_y, n_z)$ est son vecteur normal

**Algorithme 2** Technique de détection de régions planaires

**Entrée:** Carte de Profondeur *Dmap*
**Sortie:** Ensemble de Plans 3D détectés

Segmenter *Dmap* avec RANSAC
Estimer le plan principal $\pi$
**Si** Nombre de points coplanaires $N >$ seuil minimal **Alors**
    Région planaire $\pi$ détectée
    Enregistrer n, d et N
    Estimer $\delta_\pi$ barycentre de $\pi$
    Calculer la matrice $\mathbf{M}_{moments}$ des moments de $\pi$
    Décomposer $\mathbf{M}_{moments}$ en Valeurs Singulières (SVD)
    Estimer les axes principaux de $\pi$
    Enregistrer la boite englobante $\mathcal{B}$
    Extraire les points 3D $\in \pi$ de *Dmap*
    Refaire 1
**else**
    Points coplanaires insuffisants
    Arrêter
**Fin Si**

**Retourner** Paramètres de Plans 3D enregistrés

et $d$ sa distance de l'origine. Un point $\boldsymbol{p}$ de cordonnées $(x, y, z)$ appartenant à cette région satisfait l'équation du plan :

$$n_x x + n_y y + n_z z = -d \tag{3.7}$$

autrement écrite :

$$n^\top \boldsymbol{p} = -d \tag{3.8}$$

Pour pouvoir représenter un plan dans la carte nous avons aussi besoin de définir sa zone de délimitation. Nous utilisons alors les coordonnées des points extrêmaux de la région planaire selon ses axes principaux centrés sur son barycentre. Quand une région est détectée, dans le repère local $\mathscr{F}_c$, nous enregistrons le nuage de points 3D correspondant à cette région. Ensuite, nous construisons la matrice des moments centrés du second ordre de ce nuage de points, afin de trouver les vecteurs directeurs (axes principaux) du plan détecté.
Ces moments sont calculés comme suit :

$$I_{xx} = \frac{\sum\limits_{x\in\pi}(x - x_{\delta_\pi})^2}{N}, I_{yy} = \frac{\sum\limits_{y\in\pi}(y - y_{\delta_\pi})^2}{N}, I_{zz} = \frac{\sum\limits_{z\in\pi}(z - z_{\delta_\pi})^2}{N}$$

$$I_{xy} = \frac{\sum\limits_{x,y \in \pi} (x - x_{\delta_\pi})(y - y_{\delta_\pi})}{N}$$
$$I_{xz} = \frac{\sum\limits_{x,z \in \pi} (x - x_{\delta_\pi})(z - z_{\delta_\pi})}{N}$$
$$I_{yz} = \frac{\sum\limits_{y,z \in \pi} (y - y_{\delta_\pi})(z - z_{\delta_\pi})}{N}$$

Avec $\delta_\pi$ est le barycentre de la région planaire $\pi$. Ainsi la matrice des moments $\mathbf{M}_{moments}$ est :

$$\mathbf{M}_{moments} = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{pmatrix} \tag{3.9}$$

La Décomposition en Valeurs Singulières (SVD) de cette matrice permet d'obtenir les trois axes principaux de la région planaire. Ainsi nous cherchons les points extrêmaux suivant les deux premiers axes principaux étant donné que le troisième représente la normale au plan que nous avons déjà obtenue lors de la détection. Ces points sont projetés dans le repère global $\mathcal{F}_w$, une fois l'estimation de la pose courante réalisée, pour définir les limites du plan 3D dans ce repère. Nous pouvons ainsi définir la boite englobante du plan $\mathcal{B}$. La figure 3.3 montre l'exemple d'un plan 3D représenté par son modèle englobant les points 3D de la région détectée ainsi que ses vecteurs directeurs.

Par la suite, chaque région planaire détectée sera définie par $\pi[n, d, N, \mathcal{B}]$ rassemblant le modèle du plan $(n, d)$, son nombre d'inliers $N$ ainsi que sa boite englobante $\mathcal{B}$.

### 3.3.2 Points d'intérêt Planaires

Les points d'intérêt 3D, tels qu'ils sont utilisés dans l'approche de base, sont générés en associant à chaque point d'intérêt 2D la valeur correspondante dans la carte brute de la profondeur sans aucune étape de vérification des valeurs récupérées.
Cependant, quand nous avons étudié le capteur RGB-D, nous avons conclu que l'une de ses grandes faiblesses est l'erreur sur les données de profondeur qui augmente de manière proportionnelle à la distance.
Compte tenu du problème, un mécanisme de pré-traitement de données de profondeur a été mis en place utilisant les plans détectés pour améliorer la copla-

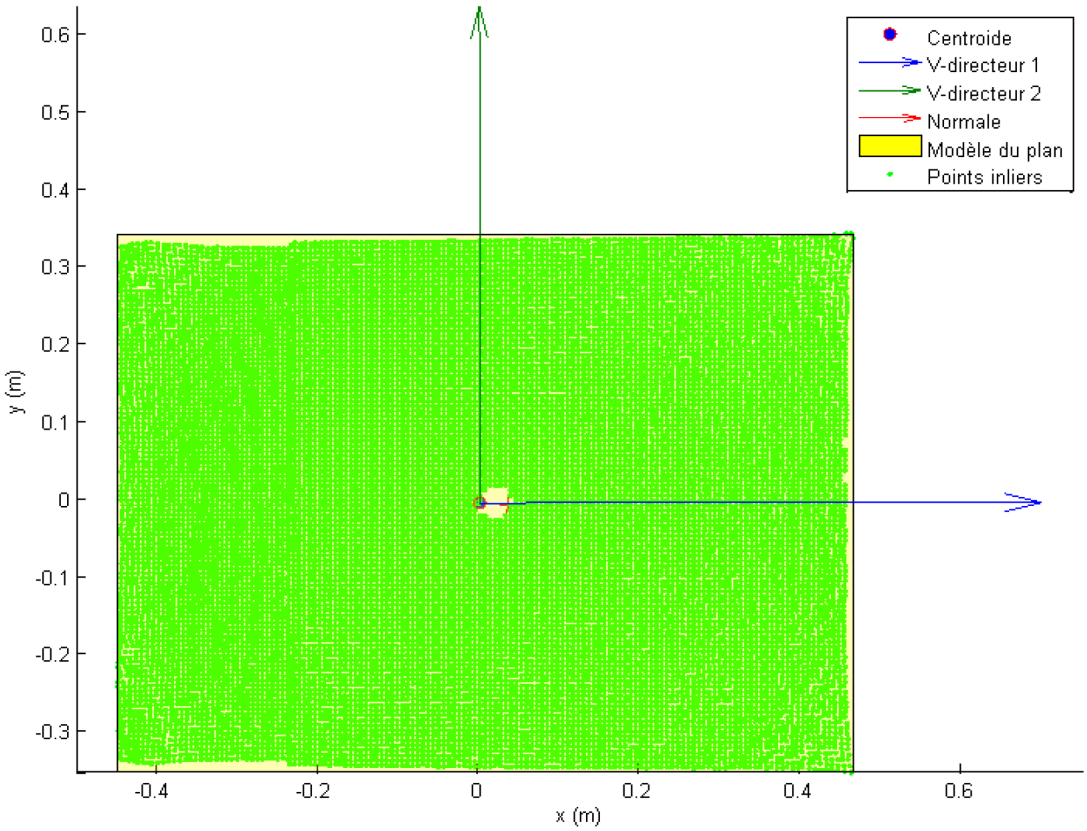



**FIGURE 3.3** – *Représentation paramétrique d'un plan 3D dans son propre repère (vue face).*

narité des points d'intérêt 3D et par conséquent l'estimation de la pose. Ainsi, la nouveauté dans notre système réside dans l'utilisation des points d'intérêt planaires pour la localisation au lieu des points d'intérêt 3D bruts.
Les études menées dans [38] et [92] ont montré que les primitives planaires sont plus sûres et robustes au bruit. Puisque les plans 3D sont estimés à partir d'un grand ensemble de points 3D (consensus), les points leur appartenant présentent une probabilité d'erreur moins élevée que les points bruts. Par conséquent, l'utilisation des primitives planaires conduit à plus de précision lors de l'estimation de pose qui converge rapidement.
Contrairement aux approches précédentes, nous définissons un point d'intérêt 3D planaire "*3D Planar Features*" comme un point d'intérêt satisfaisant le modèle d'un plan 3D détecté. Cela permet de profiter, à la fois de la légèreté des points d'intérêt et de l'efficacité de l'utilisation des plans 3D, dans le processus d'estimation de pose.

La génération de nos points d'intérêt planaires commence, à la réception

d'une image couleur et de la carte de profondeur correspondante, par la détection des régions planaires 3D dans le nuage de points et l'extraction des points d'intérêt visuels 2D à partir de l'image couleur séparément. Pour chaque point d'intérêt 2D, nous récupérons la valeur de profondeur correspondante pour générer un point 3D dont l'appartenance à une des régions planaires détectées dans la même pose est ensuite vérifiée. Les points 3D satisfaisants ainsi l'équation d'un modèle de plan sont conservés et les autres sont rejetés. Finalement, nous effectuons une étape de régularisation en projetant tous les points 3D conservés dans leur modèle de plans respectifs et ainsi nos points d'intérêt 3D planaires sont élaborés. Cependant, nous vérifions que le nombre de points d'intérêt planaires est suffisant pour pouvoir les utiliser dans l'estimation de mouvement, mais aussi pour éviter le cas où la scène manque des structures planaires, faute de quoi nous conservons les points d'intérêt 3D bruts générés par défaut. Théoriquement, la transformation entre deux poses peut être estimée à partir de trois points 3D appariés dans ces deux poses. Cependant, il est apparu lors de nos expériences que, pour pouvoir estimer une bonne transformation, au mois huit points appariés entre deux poses doivent être utilisés. Ainsi, nous avons fixé le nombre minimum de points d'intérêt 3D planaires, dans une pose, à huit points pour garantir le seuil de correspondances entre les points d'intérêt planaires lors de l'estimation d'une transformation.

Le pseudo-code de la méthode de génération de nos points d'intérêt 3D planaires est donné dans l'algorithme 3.

Dans cette thèse, nous avons choisi d'utiliser le détécteur/descripteur SURF pour le traitement des points d'intérêt visuels. Lors d'une l'application en temps réel, SURF offre le meilleur compromis précision des estimations et temps d'exécution par rapport aux SIFT et aux ORB (expérience ) Même si SIFT est plus robuste à l'erreur, son inconvénient est en effet la consommation importante de ressources et le temps de calcul élevé. En revanche, si une GPU est disponible, ce qui n'a pas été envisagé pour cette thèse, une version SIFT GPU a été proposée par Sinha[86] et reste le meilleur choix.

### 3.3.3 Cartes 3D basées-Plans

Comme mentionné précédemment, notre objectif est de produire des cartes 3D réduites basées sur les plans 3D, qui peuvent être utilisées pour la navigation d'un robot mobile ou dans des applications de la réalité augmentée dans un

**Algorithme 3** Génération des points d'intérêt planaires

**Entrée:** Image RGB, Carte de Profondeur
**Sortie:** Points d'intérêt 2D et 3D

```
Extraire les points d'intérêt visuel 2D || Extraire les Plans 3D π
Pour chaque Plan π_i faire
    Pour chaque point d'intérêt 2D faire
        Si Point 3D correspondant ∈ π_i Alors
            Conserver le point d'intérêt 2D
            Générer le point d'intérêt 3D planaire
        Fin Si
    Fin Pour
Fin Pour
Si Taille des points d'intérêt planaires < Seuil Alors
    Conserver les points d'intérêt 2D d'origine
    Construire les points d'intérêt 3D ordinaire
Fin Si
```

**Retourner** Points d'intérêt 2D et 3D

milieu d'intérieur.
Nous avons vu que chaque nuage de points RGB-D contient des milliers de points et nécessite environs 3,4 mégaoctets de mémoire. Pour les applications à faible coût tels que les petits robots, notre choix de créer des cartes basées-plans s'avère efficace et permet d'éviter les représentations basées-points qui sont très redondantes, en raison des chevauchements partiels de champs de vue de la caméra, et nécessitent beaucoup de ressources mémoire. Basé sur les régions planaires détectées, et la transformation estimée à chaque pose, notre système ajoute des structures planaires à la carte globale et les développe au cours du temps.

Pour construire la carte 3D globale, les plans doivent être représentés dans le même repère. Comme la détection des régions planaires est effectuée sur les nuages de points locaux, les plans enregistrés doivent être transformés depuis le repère locale de la caméra $\mathscr{F}_c$ vers le repère global $\mathscr{F}_w$ en utilisant la pose globale $T_w$ (Figure 3.4).

Nous définissons le repère globale comme étant le repère du premier noeud ajouté au graphe. À chaque ajout d'un nouveau noeud, nous enregistrons la transformation globale $T_w$ menant à ce repère (comme expliqué dans la section 1.3.2.1 page 27). Pour un point 3D local $\boldsymbol{p}_c$, son correspondant dans le repère global $\boldsymbol{p}_w$ peut être retrouvé facilement à travers cette transformation

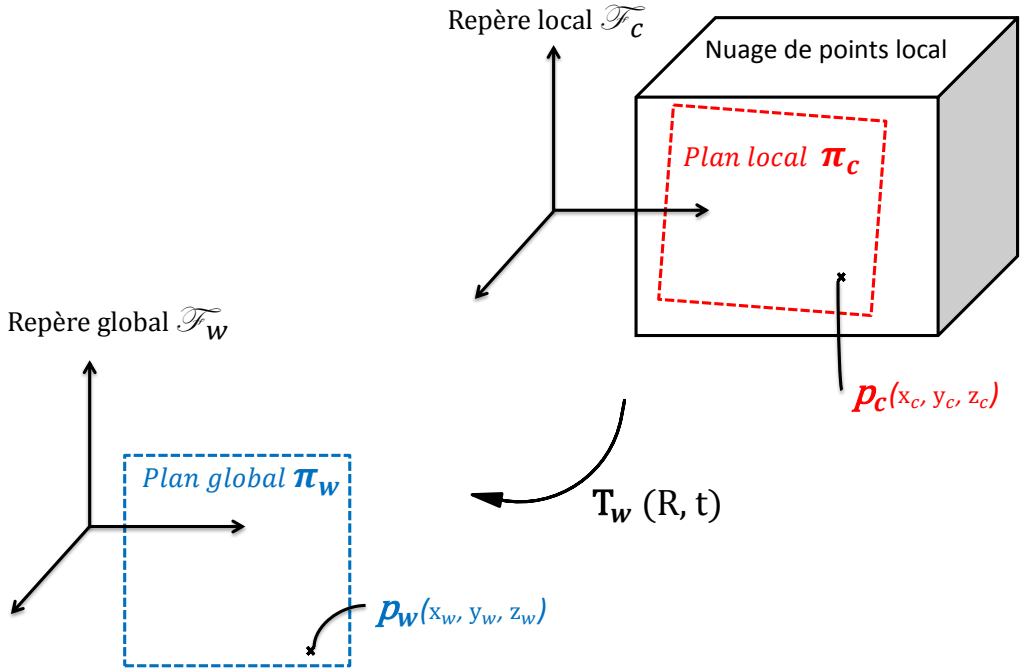



**FIGURE 3.4** – *Transformation entre le repère local $\mathcal{F}_c$ et le repère global $\mathcal{F}_w$.*

(Rotation **R** et Translation **t**) en utilisant l'équation habituelle :

$$\boldsymbol{p}_w = \mathbf{R}\boldsymbol{p}_c + \mathbf{t} \tag{3.10}$$

Et inversement

$$\boldsymbol{p}_c = \mathbf{R}^\top \boldsymbol{p}_w - \mathbf{R}^\top \mathbf{t} \tag{3.11}$$

Si ce point appartient à un plan local $\pi_c(n_c, d_c)$, il vérifie alors l'équation (3.8), ainsi

$$\begin{aligned}
n_c^\top(\mathbf{R}^\top \boldsymbol{p}_w - \mathbf{R}^\top \mathbf{t}) &= -d_c \\
n_c^\top \mathbf{R}^\top \boldsymbol{p}_w - n_c^\top \mathbf{R}^\top \mathbf{t} &= -d_c \\
(n_c^\top \mathbf{R}^\top)\boldsymbol{p}_w &= -d_c + n_c^\top \mathbf{R}^\top \mathbf{t} \\
(\mathbf{R}n_c)^\top \boldsymbol{p}_w &= -(d_c - n_c^\top \mathbf{R}^\top \mathbf{t}) \\
(n_w)^\top \boldsymbol{p}_w &= -d_w
\end{aligned}$$

Avec $n_w = \mathbf{R}n_c$ et $d_w = d_c - n_c^\top \mathbf{R}^\top \mathbf{t}$. Ainsi, le plan est défini dans le repère

global par sa normale et sa distance $\pi_w(n_w, d_w)$.

Une fois ces paramètres définis, nous procédons à une comparaison plan-à-plan afin de mettre à jour la carte globale. Si aucune correspondance ne peut être trouvée, le modèle est ajouté à la carte comme un nouveau plan. Sinon, à chaque fois qu'une nouvelle région planaire correspond à une autre enregistrée auparavant, nous mettons à jour la carte en fusionnant les modèles des deux plans correspondants en un seul plan résultant. La correspondance entre les régions planaires est examinée en deux étapes (figure 3.5, Page 77).

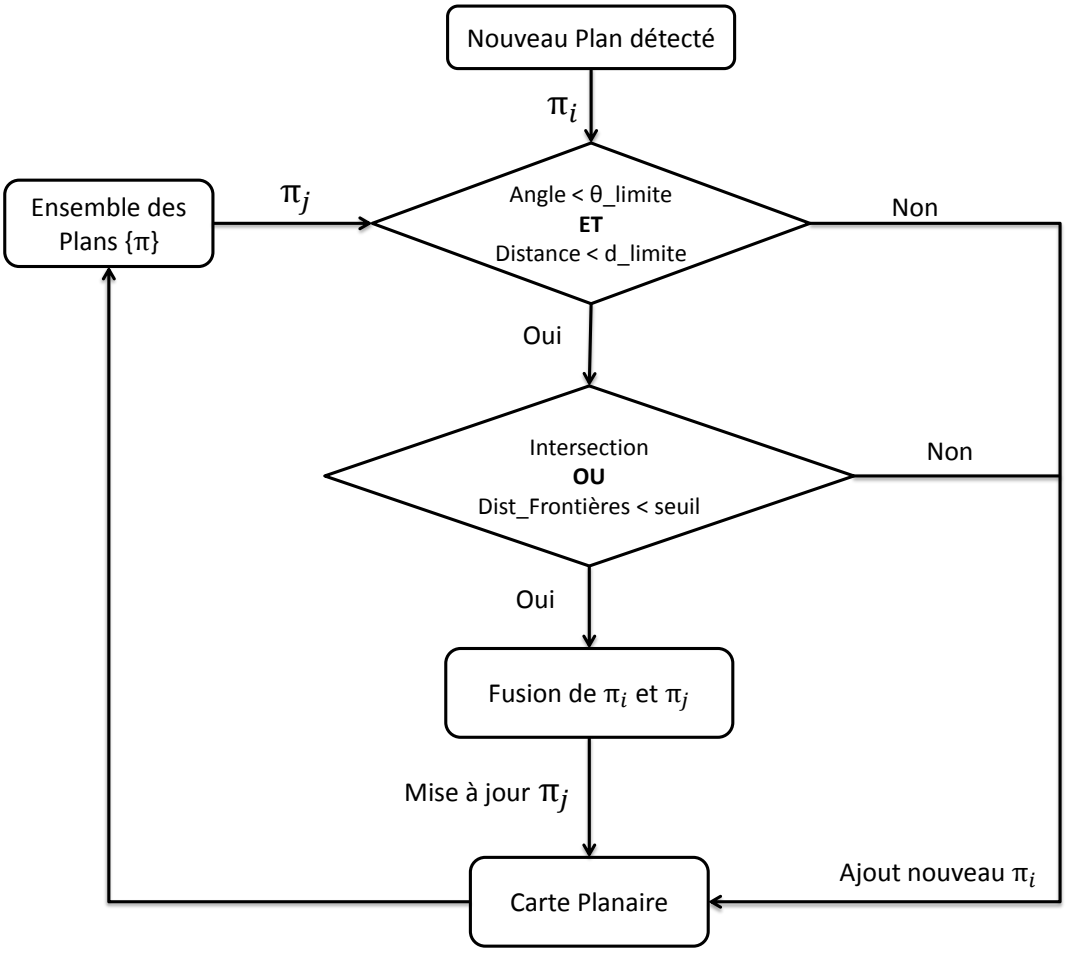



**FIGURE 3.5 –** *Algorithme de construction de la carte planaire.*

Tout d'abord, l'angle entre deux modèles de plans ne doit pas dépasser un seuil fixé à 10 degrés et la distance limite entre eux ne doit pas excéder $5cm$. Ces limites ont été définies empiriquement en tenant compte des éventuelles erreurs d'estimation de pose et de détection de plans. En effet, la transformation estimée n'est pas toujours exacte, et l'erreur de l'estimation peut engendrer un décalage

entre deux plans correspondants à la même région dans deux poses différentes. D'autre part, la détection de plans peut aussi être partiellement erronée (distance ou normale au plan non précises), ce qui produit, pour la même région planaire, des modèles de plans légèrement différents d'une pose à l'autre. Ainsi, ces erreurs sont prises en considération lors de la mise en correspondance des plans, d'où les seuils fixés précédemment.

Lorsque les deux premières conditions sont satisfaites, nous vérifions que les deux régions planaires s'intersectent ou sont au moins spatialement proches. Cette étape est nécessaire pour éviter la fusion des plans qui possèdent des modèles identiques, mais ne représentent pas un plan continu dans le monde réel. Un exemple d'une telle situation peut être une portion de mur discontinue avec une porte ouverte au milieu, où les deux parties du mur ont la même équation. Pour faire face à ce problème, nous vérifions l'intersection des boites englobantes des deux plans concernées. Dans l'exemple de la figure 3.6, les deux plans ($\pi_i$ et $\pi_j$) vont être fusionnés puisqu'ils s'intersectent dans l'espace. Si l'intersection n'est pas trouvée, la distance minimale entre les boîtes englobantes des deux plans est considérée. En effet, la région détectée peut ne pas avoir d'intersection avec la région précédente, même s'il s'agit du même plan, suite à un grand déplacement ou à une mauvaise détection (vue partielle). En prenant en considération ce cas de figure, nous avons défini une distance limite entre les extrémités des deux régions au dessous de laquelle la fusion est effectuée. Cette distance limite a été fixée à $50cm$ en étudiant les discontinuités produites lors des déplacements, notamment à cause des rotations du capteur.

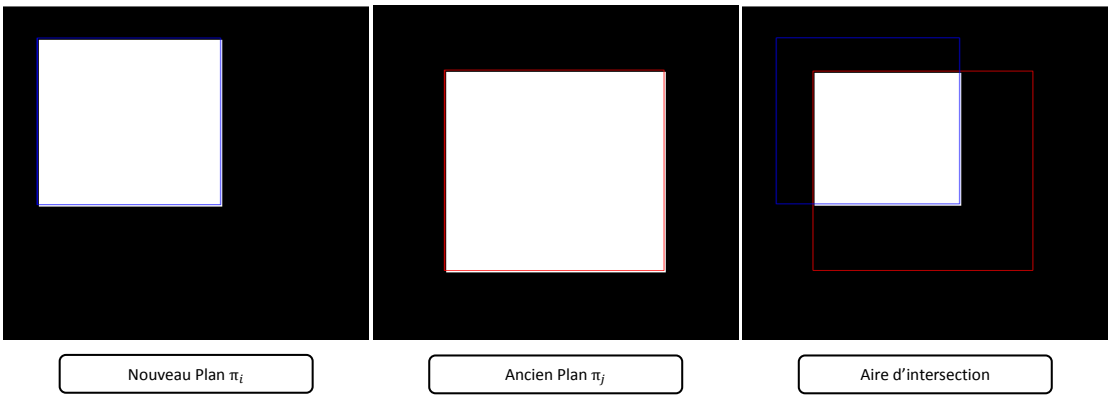



**FIGURE 3.6** – *Chevauchement de plans.*

Ainsi, lorsqu'une nouvelle région planaire $\pi_i$ correspond à une région pla-

naire de la carte $\pi_j$, elles sont fusionnées dans un nouveau plan résultant en tenant compte de leurs populations de points (inliers) respectives. En effet, le modèle de plan d'une grande région planaire, estimé grâce à un grand nombre de points 3D, est plus précis que celui d'une petite région planaire. Par conséquent, nous accordons à la grande région planaire un poids plus élevé lors de la fusion de plans, afin de ne pas dégrader la qualité du plan résultant de cette fusion. Les paramètres du nouveau plan sont définis comme suit :

$$n_{fusion} = \alpha \mathbf{n}_i + (1 - \alpha)\mathbf{n}_j$$

et

$$d_{fusion} = \alpha d_i + (1 - \alpha)d_j$$

Avec

$$\alpha = \frac{N_i}{N_i + N_j}$$

Le plan résultant contiendra la somme des points des deux plans fusionnés : $N_{fusion} = N_i + N_j$. Leurs boîtes englobantes sont aussi comparées et les valeurs extrêmes sont affectées au nouveau plan.

Par ailleurs, notre carte contient aussi les intersections théoriques entre les plans. Ces intersections sont générées en utilisant un critère d'adjacence des plans détectés. Dans la carte, elles sont représentées par des lignes et des points lorsque, respectivement, deux ou trois plans s'intersectent. En outre, à chaque mise à jour d'un plan, nous mettons aussi à jour ses intersections enregistrées en recalculant les intersections théoriques du nouveau plan avec les plans adjacents. Ainsi, nous générons une carte de primitives planaires qui représente une première étape vers une carte plus sémantique. De plus, nous avons modélisé cette carte sous forme d'un graphe où les noeuds sont les plans détectés et où les arêtes représentent les intersections entre les plans adjacents. Étant donné que la manière dont les noeuds et les arêtes sont définis dans les cartes topologiques est relative à leurs utilisations ([24] section 1.3), notre carte peut être considérée aussi comme les prémisses d'une carte topologique où les plans représentent des lieux et leurs intersections correspondent aux relations entre les lieux. Cette représentation hybride rend notre carte plus significative et praticable pour d'autres applications tel que la navigation d'un robot.

## 3.4 Conclusion

Dans ce chapitre nous avons présenté notre méthode RGB-D SLAM basé-plans. Ce système étend l'approche de Endres *et al.* [31, 32] pour utiliser les plans 3D dans le processus. Notre approche profite des données de la profondeur du capteur pour extraire les structures planaires de l'environnement. Comme ils sont fréquents dans les milieux intérieurs et facile à modéliser, nous avons intégré les plans 3D pour générer des points d'intérêt 3D planaires et pour créer la carte de l'environnement.

Dans un premier temps, nous avons proposé une méthode permettant de générer des points d'intérêt 3D planaires. Cette méthode repose sur l'utilisation des modèles de plans détectés à chaque pose pour régulariser les mesures brutes issues du capteur de profondeur de la caméra RGB-D. L'objectif était de diminuer l'erreur des valeurs de profondeur des points d'intérêt 3D. Nous avons vu que dans l'approche d'origine un point d'intérêt 2D est transformé en un point 3D en récupérant la mesure brute de profondeur correspondante. Or les valeurs récupérées peuvent être bruitées ou complètement absentes. En outre, tenant compte des natures dissemblables des espaces de mesures, pixellique pour les points d'intérêt 2D et métrique pour la carte de profondeur, les valeurs accordées aux points 2D sont approximatives. Tout ces problèmes amènent à générer des données hétérogènes, au niveau de la profondeur, pour des points qui peuvent être homogènes dans la réalité (les points d'un plan par exemple). Par conséquent les erreurs dans l'estimation initiale de la pose peuvent être importantes et demanderont beaucoup de temps pour être raffinées.
Comme le modèle d'un plan est estimé par un consensus de tous les points 3D appartenant à une région planaire, la distance estimée est généralement très proche de la réalité. Ainsi, en projetant tous les points d'intérêt 3D qui vérifient l'équation d'un plan dans leur modèle, nous avons généré des points d'intérêt 3D planaires moins bruités par rapport aux points bruts. Ces points vont être utilisés dans l'estimation des trajectoires de la caméra et leur influence sur le processus de la localisation fera l'objet d'une partie de l'étude expérimentale détaillée dans le chapitre suivant.

La deuxième méthode développée permet de créer une représentation de la carte globale basée sur les primitives planaires de l'environnement. Vu les défauts présents dans les cartes basées-points qui sont redondantes et difficile-

ment exploitables, nous avons proposé un algorithme permettant la génération des cartes 3D réduites basée-plans. La construction de la carte commence par la détection des plans 3D dans chaque nouveau nuage de points reçu. Ensuite, une fois la pose correspondante au nuage de points est estimée, chaque plan enregistré localement (dans le repère du nuage) est transformé dans le repère global. Afin d'avoir une représentation compacte et fidèle de l'environnement, les structures planaires continues doivent être représentées par un seul plan dans la carte. Ainsi, notre algorithme procède à la fusion des modèles de plans correspondants à chaque détection en tenant compte des erreurs possibles de l'estimation de pose et de détection de plans. Cette fusion permet de générer un plan résultant à chaque correspondance de deux plans en se basant sur leurs modèles et nombres d'inliers. En cas de discontinuité, les plans sont ajoutés comme de nouvelles primitives dans la carte. Notre méthode permet de créer des cartes planaires denses et légères de l'environnement qui peuvent être utilisées dans des applications de la robotique ou de la réalité augmentée. L'évaluation de ces cartes sera présentée dans le chapitre suivant qui est consacré aux expériences et résultats.

# Chapitre 4

# Évaluations et Résultats

## Sommaire


**4.1 Introduction . . . . . 84**
**4.2 Détection de plans . . . . . 85**
4.2.1 Objectifs . . . . . 85
4.2.2 Protocoles . . . . . 85
4.2.3 Résultats . . . . . 87
**4.3 Benchmarks TUM . . . . . 94**
4.3.1 Objectifs . . . . . 94
4.3.2 Protocoles . . . . . 95
4.3.3 Résultats . . . . . 95
**4.4 Séquence Bureau . . . . . 98**
4.4.1 Représentation Cartographique . . . . . 98
4.4.1.1 Carte 3D basée-points . . . . . 98
4.4.1.2 Carte 3D basée-Plans . . . . . 101
4.4.1.3 Comparaison des représentations de la carte . . 101
4.4.2 Estimation de la pose . . . . . 103
4.4.2.1 Objectifs . . . . . 103
4.4.2.2 Protocoles . . . . . 103
4.4.2.3 Résultats . . . . . 106
4.4.3 Évaluation de la Carte . . . . . 108
4.4.3.1 Objectifs . . . . . 108
4.4.3.2 Protocoles . . . . . 109
4.4.3.3 Résultats . . . . . 109

Nous allons consacrer ce chapitre à l'étude et l'évaluation de l'approche proposée dans le chapitre précédent en détaillant les expériences réalisées et les résultats obtenus. Pour commencer nous présenterons les expériences préliminaires menées dans l'objectif d'étudier les conditions d'utilisation garantissant l'efficacité de notre système ainsi que sa compatibilité pour une application temps-réel. Nous évaluerons par la suite les performances de cette méthode et notamment la précision des trajectoires et de la carte, en deux temps. Tout d'abord nous utiliserons des données de référence avec leurs outils d'évaluation, puis nous proposerons nos propres protocoles expérimentaux adaptés aux scènes de bureau.

## 4.1 Introduction

Les tâches principales liées à un système de SLAM sont la localisation et la cartographie. Si lors du développement du problème il était acceptable de proposer des solutions hors ligne, aujourd'hui les applications robotiques exigent plus de rapidité dans les traitements pour pouvoir réaliser leurs fonctions en temps réel. En outre, compte tenu des progrès récents de la robotique mobile, il y a une demande croissante de solutions efficaces en termes d'estimation du mouvement et de qualité des cartes résultantes. Ainsi, il est devenu nécessaire qu'un algorithme SLAM soit en mesure de répondre à toutes ces exigences. Comme nous avons proposé une méthode destinée principalement à ce genre d'applications, robotique mobile, une évaluation selon ces critères est nécessaire pour pouvoir juger de la qualité des résultats. Afin d'évaluer les performances de notre approche, nous avons ainsi procédé à plusieurs séries d'expériences. Dans ces expériences nous avons utilisé des séquences de l'ensemble de données TUM [91] ainsi que des données acquises en temps réel avec une Kinect v1 dans notre scène de bureau préalablement préparée pour cet objectif. Toutes les expériences ont été réalisées sur un ordinateur équipé d'un processeur Intel Core i5-2400 CPU 3.10GHz $\times 4$.

Dans notre système, seuls les trois premiers plans principaux ont été considérés lors de la détection des régions planaires. En effet, étant donné que, lors de la détection des plans, ceux ci sont extraits par ordre de populations décroissantes, l'ajout de petits plans supplémentaires peut introduire un grand nombre

de petites portions planaires non pertinentes. En outre, le nombre minimal de points 3D inliers d'une région planaire est fixé à 700 points pour qu'elle soit considérée comme un plan valide. Ces paramètres ont été choisis lors des expériences préliminaires afin de fournir des résultats significatifs lorsque le système tourne en ligne. Les protocoles expérimentaux et les résultats concernant la qualité des plans, l'estimation des trajectoires et la construction de la carte sont présentés dans les sections suivantes.

## 4.2 Détection de plans

### 4.2.1 Objectifs

Les plans 3D détectés dans l'environnement constituent la base de notre système. Par conséquent toute erreur de détection de plans engendrera des erreurs importantes au niveau de l'estimation de la pose, puisque les plans sont utilisés pour générer les points d'intérêt 3D planaires, ainsi que dans la carte globale de l'environnement construite par ces plans .
Nous avons donc réalisé plusieurs expériences pour évaluer la qualité des plans générés, par rapport à une vérité terrain, ce qui nous a permis de déterminer les limites d'utilisation de notre approche.
D'autre part, comme la détection de plans introduit des traitements dans le processus du système, nous avons évalué son influence sur le fonctionnement en temps réel de notre approche.

### 4.2.2 Protocoles

Pour analyser la qualité de la détection de plans, deux critères ont été évalués : la précision des mesures et le temps d'exécution. Dans toutes les expériences de cette section la caméra a été initialement fixée en face d'un plan parmi les différents plans de notre scène bureau.

Nous avons tout d'abord examiné la précision de la détection de plans relativement au seuil d'appartenance des points 3D à un modèle de plan estimé dans un nuage de points. En effet, l'algorithme RANSAC utilisé pour la détection commence par l'estimation d'un modèle de plan 3D initial à partir de points 3D aléatoires. Ce modèle est ensuite raffiné à chaque itération de l'algorithme

par l'ajout d'autres points 3D en tenant compte de leur distance par rapport au modèle initial.
Nous définissons $\varepsilon$ comme étant la distance maximale à laquelle un point 3D est considéré comme appartenant (inlier) ou pas (outlier) à une région planaire lors de sa détection. Nous supposons que les nuages de points brut comprennent des points 3D bruités, notamment leur valeur de profondeur, à cause du bruit caractéristique du capteur de profondeur. Par conséquent, lors de la détection d'une région planaire, nous cherchons à trouver le modèle moyen de plan, incluant la plupart des points 3D appartenant à cette région dans la scène réelle. Pour déterminer le seuil optimal, qui garantit une estimation exacte du modèle et de ses inliers, nous avons fait varier la valeur de $\varepsilon$ lors de la détection de plans. Une estimation exacte est celle qui permettra de générer des régions planaires homogènes contenant la majorité des points 3D appartenant réellement aux plans observés avec des modèles correspondants précis. D'autre part, la distance du plan observé par rapport à la caméra peut influencer le résultat de la détection car l'erreur de profondeur augmente avec la distance. Ainsi, afin de trouver le seuil optimal permettant une bonne détection quelque soit la distance de la caméra, nous avons détecté les plans de la scène depuis différentes distances tout en variant le seuil de la détection $\varepsilon$. Pour pouvoir observer les résultats de la détection directement dans les nuages de points, nous avons marqué les points 3D inliers de chaque région planaire détectée par une couleur différente (tableaux 4.1, 4.2, 4.3, 4.4 (pages 88 et 89)). Quand une région planaire est détectée avec une bonne précision, l'ensemble de points 3D lui appartenant dans la scène réelle, figurent avec la même couleur dans le nuage de points. Cela permet de définir les limites de la détection de plans et les paramètres optimaux.

La deuxième série d'expériences concerne l'impact de la détection de plan sur l'utilisation temps-réel de notre système. Lors des expériences préliminaires, nous avons remarqué que l'extraction d'un plan 3D à partir d'une carte de profondeur, dont la résolution par défaut est $640 \times 480$ (307200 points), demande en moyenne trois secondes. Si ce nuage contient plusieurs plans, la détection de ces plans prendra donc plusieurs secondes. En tenant compte de la haute fréquence d'acquisition des images de la Kinect (30Hz), le processus de détection de plans ralentira tous les traitements ultérieurs du système ce qui nous éloigne considérablement de l'aspect temps réel envisagé et contraint l'utilisation de notre système. Une étape de sous-échantillonnage du nuage de points est donc

indispensable pour limiter la durée du processus de la détection et ainsi alléger le système. Ceci est souvent utilisée par les systèmes sparse pour surmonter le temps de traitement et réduire la taille des nuages de points. Un nuage de points réduit contient moins de points 3D et donc le temps nécessaire pour extraire un plan de ce nuage s'en trouve lui aussi réduit. Ainsi, nous définissons le facteur d'échantillonnage $\lambda$ utilisé lors de la création d'un nuage de points. Ce facteur réduit les dimensions des images reçues de la Kinect. Cependant, si la réduction du nombre de points 3D dans un nuage permet d'accélérer la détection de plans, la précision des plans estimés demeure primordiale pour la réussite des méthodes de localisation et cartographie, proposée dans notre approche, qui sont basée sur ces plans. Nous avons ainsi étudié l'impact des données échantillonnées sur la fidélité et la rapidité de la détection de chaque plan, en prenant en considération les distances de la caméra. Pour chaque distance réelle, nous avons fait varier le facteur $\lambda$ et observé les résultats de détection de plans suivants : la distance estimée, le temps d'extraction, le nombre de plans supplémentaires erronés, le nombre d'inliers des plans principaux détectés et les points d'intérêt leur appartenant.

### 4.2.3 Résultats

Les tableaux 4.1, 4.2, 4.3, 4.4 (pages 88 et 89) présentent les résultats de la détection de plans dans une scène simple composée de trois plans (panneaux). Le plan considéré par l'expérience est celui en face de la caméra lui même perpendiculaire aux deux panneaux latéraux. Après la détection du premier plan, le système continue cependant à extraire les plans principaux possibles de la scène pour vérifier les points 3D affectés à ces plans. Les points 3D ont été constamment observés lors de cette expérience : Les points inliers de la région planaire correspondante au plan considéré sont en bleu alors que les régions supplémentaires détectées sont colorées autrement.

Les deux paramètres influençant directement la détection sont le seuil $\varepsilon$ et la distance de la caméra. Quand le plan considéré est très proche ($1m$), les résultats des plans détectés sont comparables (tableaux 4.1) quelques soit l'épaisseur (seuil d'appartenance au plan $\varepsilon$). La précision de la distance du plan estimée est similaire pour $\varepsilon = 1.5cm$ et $\varepsilon = 1cm$. Cette précision se manifeste dans les points inliers (en bleu) qui figurent dans les plans détectés. À partir de $2m$ des erreurs de détection (les points 3D verts et rouges) commencent à apparaître

**TABLE 4.1** – *Seuil et Qualité de la détection de plan (distance réelle 1m).*

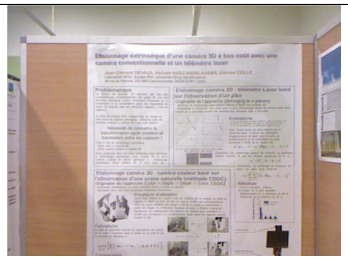
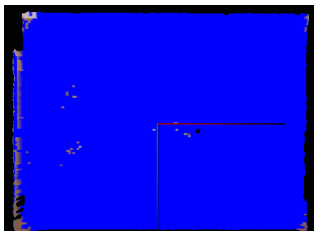
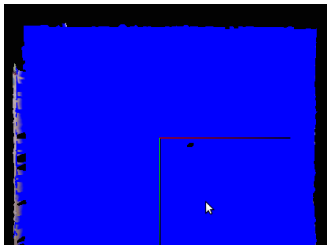
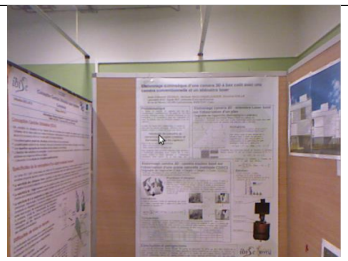
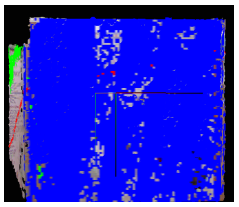
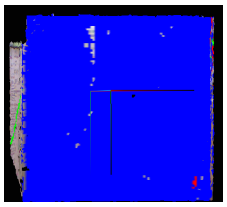
| | | |
|---|---|---|
| Plan observé |  | |
| Seuil de détection $\varepsilon$ | 1.0cm | 1.5cm |
| Plan détecté |  |  |
| Distance estimée | 1.05m | 1.07m |

**TABLE 4.2** – *Seuil et Qualité de la détection de plans (distance réelle 2m).*

| | | |
|---|---|---|
| Plan observé |  | |
| Seuil de détection $\varepsilon$ | 1.0cm | 1.5cm |
| Plan détecté |  |  |
| Distance estimée | 2.10m | 2.08m |

pour les deux seuils $\varepsilon$ utilisés. Ceci est dû à l'erreur caractéristique des données de profondeur qui augmente avec la distance comme nous l'avons expliqué dans le premier chapitre. Cependant, avec un seuil $\varepsilon = 1.5cm$ l'impact de la distance sur le nombre de points 3D affectés au plan principal détecté est faible. Pour $\varepsilon = 1cm$, le plan détecté ne contient pas tous les points 3D du panneau principal, qui est censé être détecté comme la région planaire principale, ce qui

**TABLE 4.3** – *Seuil et Qualité de la détection de plans (distance réelle 2.5m).*

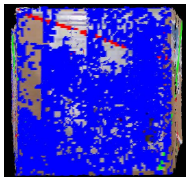
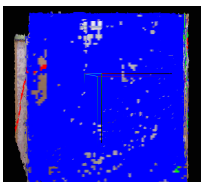
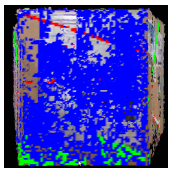
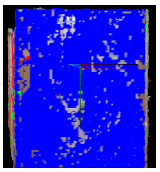
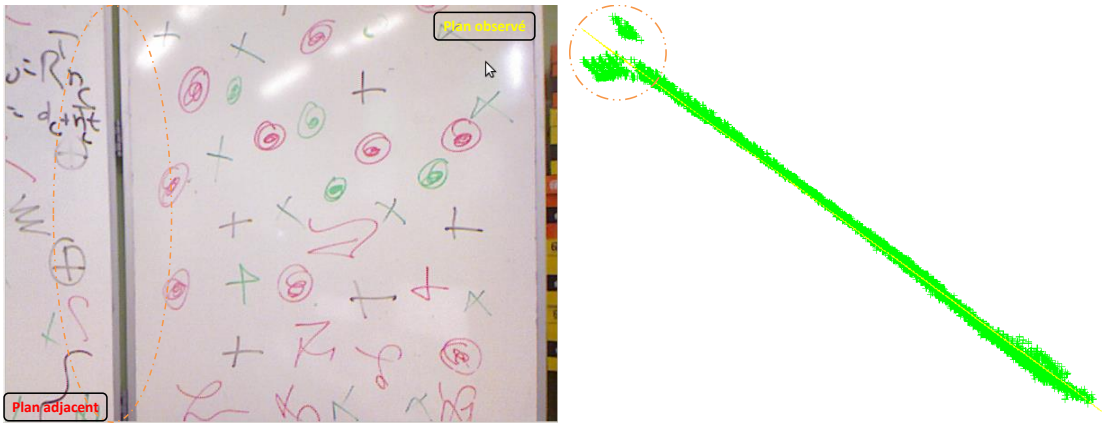
| | | |
|---|---|---|
| Plan observé |  | |
| Seuil de détection $\varepsilon$ | 1.0cm | 1.5cm |
| Plan détecté |  |  |
| Distance estimée | 2.41m | 2.53m |

**TABLE 4.4** – *Seuil et Qualité de la détection de plans (distance réelle 3m).*

| | | |
|---|---|---|
| Plan observé |  | |
| Seuil de détection $\varepsilon$ | 1.0cm | 1.5cm |
| Plan détecté |  |  |
| Distance estimée | 2.78m | 3.09m |

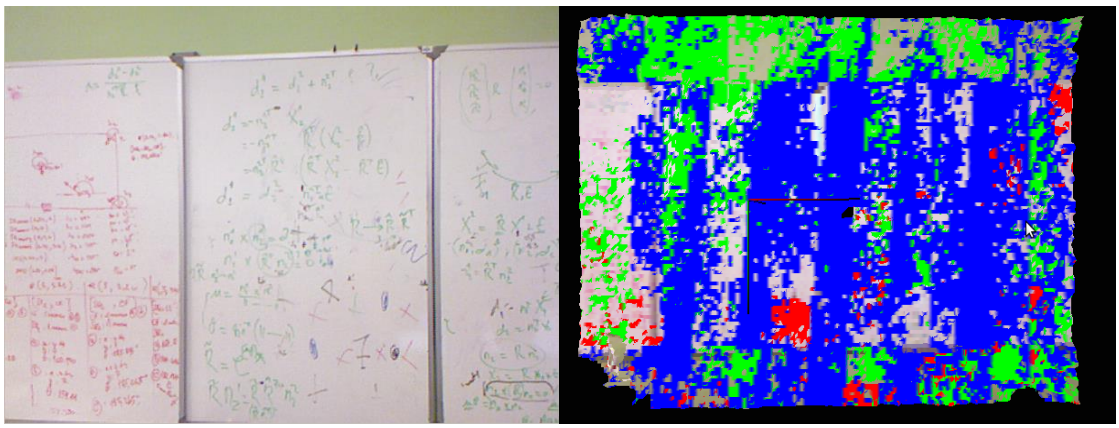
modifie les paramètres du modèle estimé. Lors de la génération de la carte de l'environnement ce genre de problème peut perturber l'étape de fusion de plans lorsque ces paramètres sont comparés à d'autres précédemment enregistrés. Un modèle de plan erroné peut être fusionné avec d'autres plans qui ne lui correspondent pas dans la scène, ce qui peut résulter en un plan inexistant réellement dans l'environnement et par conséquent diminue la fidélité de la carte globale.

Avec un seuil de détection plus élevé ($\varepsilon > 1.5cm$) des problèmes de détection de plans surgissent (figures 4.1 et 4.2). Un problème souvent rencontré, lors de l'apparition progressive d'un nouveau plan de la scène, est l'estimation des points d'un plan comme inliers d'un autre plan adjacent (figure 4.1b) et en l'occurrence le modèle de plan estimé est erroné. Ainsi, les points d'intérêt 3D planaires générés à partir de ce plan serions aberrants et conduirions à une mauvaise estimation de pose. D'autre part, pour les plans loin de la caméra ($> 2m$), la détection génère des groupements de points dispersés qui ne correspondent à aucun plan et ne permettent pas de créer des cartes planaires fiables (figures 4.2).




***(a)*** *Plans observés* ***(b)*** *Modèle avec inliers estimés (vue du dessus)*

**FIGURE 4.1** – *Erreur de l'estimation des inliers au plan*



***(a)*** *Plan observé* ***(b)*** *Régions planaires détection*

**FIGURE 4.2** – *Erreur de détection des régions planaires*

Ainsi, suite à l'analyse de ces expériences et de probables problèmes, le seuil d'appartenance d'un point 3D à un plan lors de la détection dans notre système

**TABLE 4.5** – *Impact des données échantillonnées sur la détection de plans.*

| Distance Réelle (m) | Facteur $\lambda$ | Temps de l'estimation (s) | Distance Moy. Inliers/Modèle | Distance Estimée (m) | Nombre Plans erronés |
|---|---|---|---|---|---|
| 1.5 | 1 | 2.37 | (0.4±0.3)cm | 1.49 | 0 |
| | 2 | 0.23 | (0.4±0.3)cm | 1.48 | 0 |
| | 4 | **0.07** | (0.4±0.3)cm | **1.49** | 0 |
| 2.5 | 1 | 7.26 | (0.6±0.4)cm | 2.44 | 1 |
| | 2 | 1.78 | (0.6±0.4)cm | 2.41 | 2 |
| | 4 | **0.4** | (0.6±0.4)cm | 2.41 | **0** |
| 3.5 | 1 | 7.57 | (0.6±0.3)cm | 3.32 | 2 |
| | 2 | 1.72 | (0.6±0.4)cm | 3.37 | 1 |
| | 4 | **0.4** | (0.7±0.4)cm | 3.36 | **0** |

a été fixé à $\varepsilon = 1.5cm$ pour minimiser l'erreur de cette détection, en termes de paramètres de plans estimés et de points 3D inliers considérés.

La figure 4.3 page 92 montre des exemples de plans détectés (nuage de points colorés) dans notre scène de bureau. Les deux colonnes de gauche regroupent les plans réels observés, alors que les colonnes de droite représentent les régions planaires détectées, correspondant aux plans réels, par notre système avec le seuil $\varepsilon = 1.5cm$. Les figures présentées côte à côte illustrent les plans adjacents dans la scène réelle.

Une fois le seuil optimal de la détection de plans défini, nous sommes passé à l'échantillonnage des nuages de points dans le but d'alléger le système tout en garantissant la même précision que celle obtenue précédemment. Ici l'influence de la taille des données sur le temps d'exécution est mise à l'épreuve. Le tableau 4.5 présente les résultats des expériences réalisées sur un plan de la scène. Nous avons fait varier le facteur d'échantillonnage $\lambda$ lors de la détection d'un plan à différentes distances. Pour chaque expérience, nous avons considéré le temps d'exécution de la détection, les distances estimées et le nombre de plans supplémentaires erronés détectés (petits ensembles de points non significatifs). Une première analyse des résultats indique que le facteur d'échantillonnage $\lambda = 4$ offre le meilleur compromis entre le temps d'exécution et l'exactitude des plans estimés.

Nous avons ensuite étudié l'impact des points 3D inliers dans les plans détectés sur la qualité de l'estimation de plans. Pour le faire, nous calculons l'écart-type et la moyenne de la distance des inliers par rapport au modèle du plan estimé. Même si le nombre d'inliers diminue en raison du sous-échantillonnage

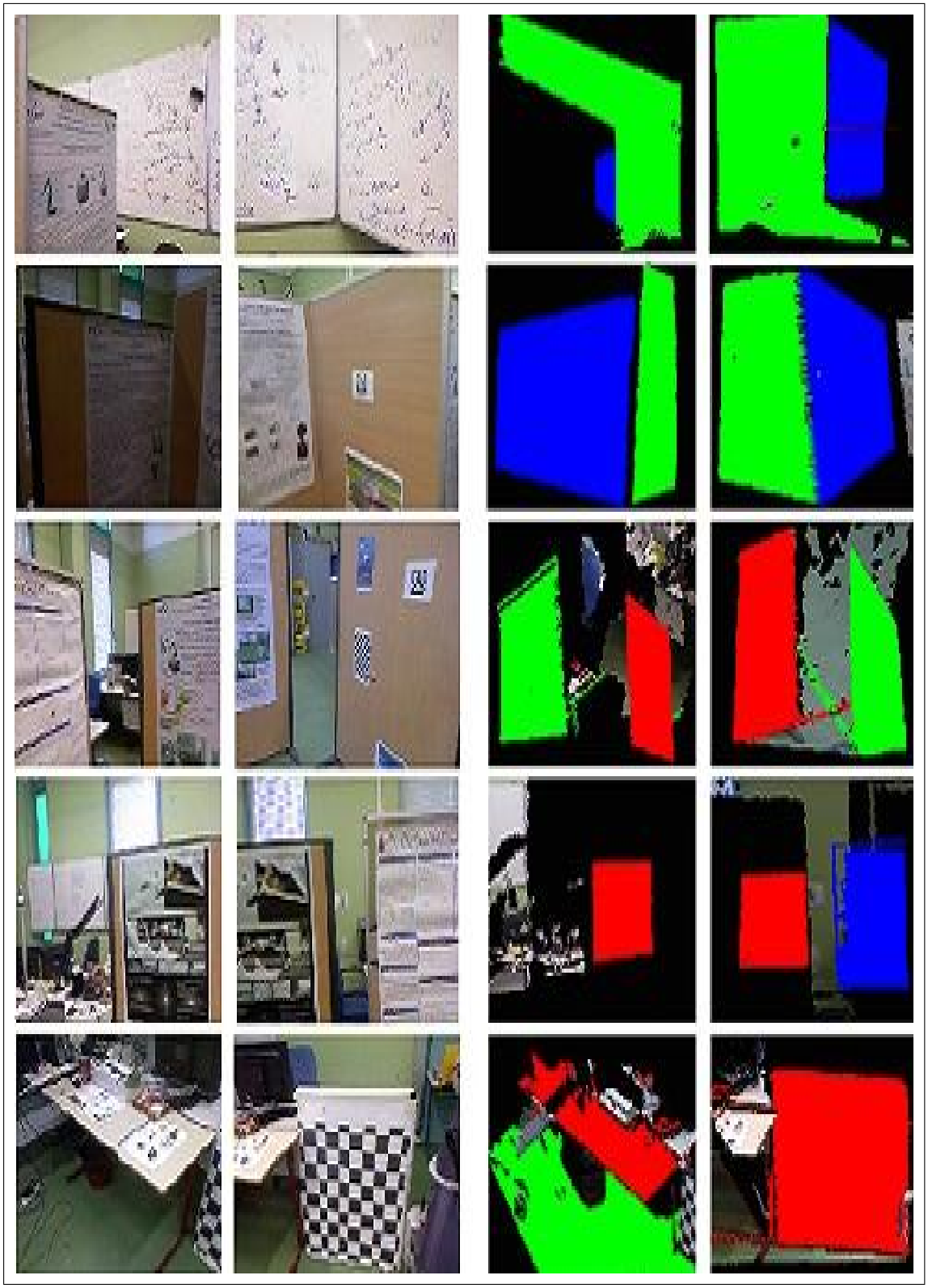
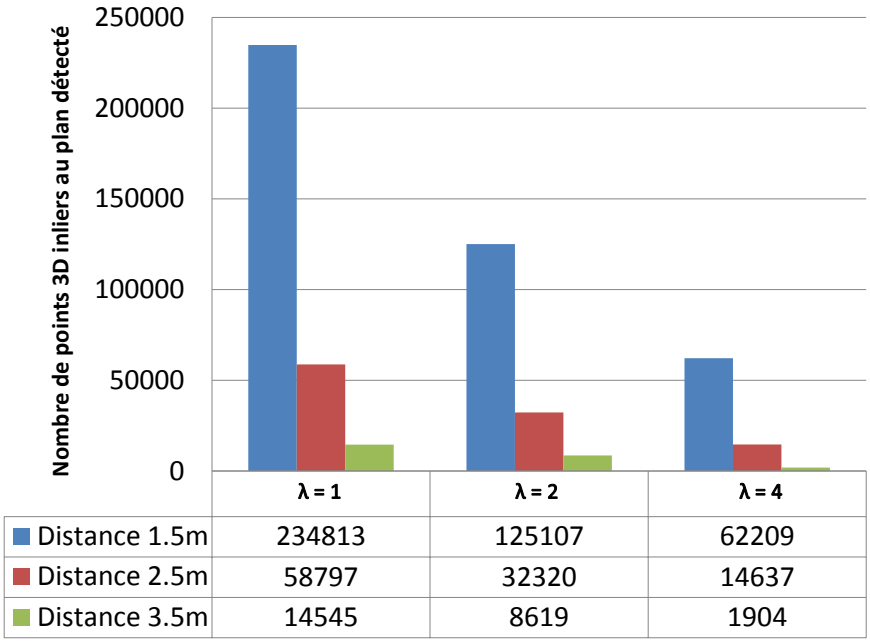


**FIGURE 4.3 –** *Notre scène expérimentale avec les plans détectés.*

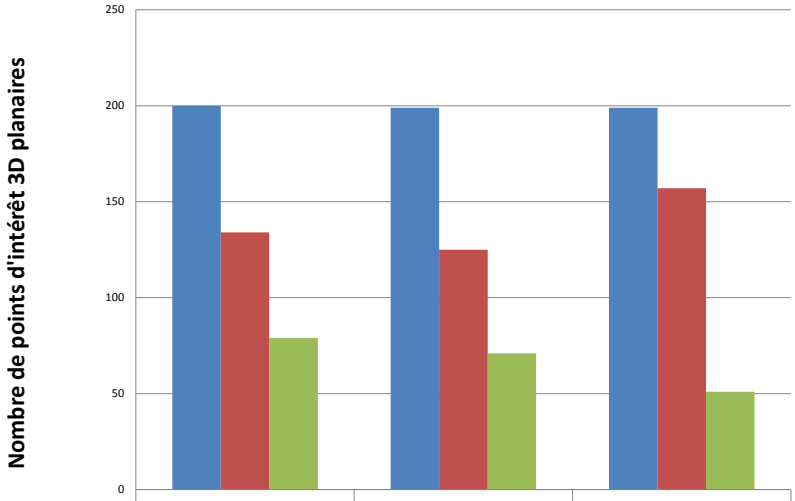
du nuage de points avec $\lambda$ (Figure 4.4), il est évident que cela n'influence pas les résultats qui restent similaires pour tous les plans détectés à la même distance. Ce nombre de points inliers n'a donc aucune incidence sur la distance estimée. Néanmoins, le nombre de points d'intérêt 3D planaires est inversement proportionnelle à la distance du plan (Figure 4.5). Comme le bruit dans les données de profondeur augmente avec la distance, la détection des plans éloignés de $3.5m$ ou plus ne peut pas être effectuée, car elle résulte en plusieurs ensembles de points dispersés qui ne représentent pas un plan réel.




| | λ = 1 | λ = 2 | λ = 4 |
|---|---|---|---|
| Distance 1.5m | 234813 | 125107 | 62209 |
| Distance 2.5m | 58797 | 32320 | 14637 |
| Distance 3.5m | 14545 | 8619 | 1904 |

**FIGURE 4.4** – *Variations du nombre de points inliers au plan détecté en fonction du facteur $\lambda$ et de la distance du plan.*

L'analyse de ces résultats autorise ainsi à procéder à un échantillonnage des nuages de points de la Kinect avec le facteur $\lambda = 4$ puisqu'aucune dégradation de la qualité des plans détectés n'est provoquée par ce sous-échantillonnage tandis qu'il permet de respecter la contrainte du traitement temps réel des données par notre système. Ce choix nous a donc permis d'intégrer l'étape de détection de plans dans le RGB-D SLAM d'origine pour pouvoir réaliser notre objectif de cartographie basée-plans en temps réel. Notons aussi que les erreurs des normales aux plans détectés ne dépassent pas 6 degrés dans tous les cas.




| | λ = 1 | λ = 2 | λ = 4 |
|---|---|---|---|
| Distance 1.5m | 200 | 199 | 199 |
| Distance 2.5m | 134 | 125 | 157 |
| Distance 3.5m | 79 | 71 | 51 |

**FIGURE 4.5** – *Variations du nombre de points d'intérêt 3D planaires (du plan détecté) en fonction du facteur λ et de la distance du plan.*

## 4.3 Benchmarks TUM

### 4.3.1 Objectifs

Afin d'évaluer les performances de notre contribution et plus particulièrement la précision des trajectoires estimées et le temps d'exécution de l'estimation de poses, nous avons procédé à deux séries d'expériences.

L'objectif de la première partie de cette évaluation consiste à vérifier que l'introduction des points d'intérêt 3D planaires ne dégrade pas les résultats de la localisation. En effet, notre système se limite à ces points pour estimer les transformations entre les poses successives au lieu d'utiliser tous les points d'intérêt détectés dans la scène. Ceci réduit considérablement l'ensemble de points utilisés et peut avoir un impact sur la qualité de cette estimation.

La deuxième partie de cette évaluation concerne la rapidité de notre système. Même si nous avons étudié le temps nécessaire pour la détection de plans dans la section précédente (Tableau 4.5), il est également important d'évaluer le temps d'exécution global de tout le processus de notre SLAM.

### 4.3.2 Protocoles

Les évaluations de cette section ont été réalisées en utilisant les données de l'université Technologique de Munich (TUM) présentées dans le deuxième chapitre. Les résultats détaillés de cette étude sont publiés dans notre papier [30]. Pour effectuer les tests nous avons sélectionné des séquences où les scènes présentent de nombreux plans suffisamment texturés : bureaux, plancher, tableaux. Pour étudier la déviation de notre trajectoire estimée par rapport la vérité du terrain, nous avons mesuré l'erreur absolue de la trajectoire (ATE).

Dans l'expérience concernant le temps d'exécution du système, nous avons utilisé une seule séquence pour laquelle les temps moyens de l'extraction des plans et de l'estimation du mouvement ont été évalués. Le temps moyen de l'exécution de la détection de plans a été calculé en divisant la somme des temps d'exécution de chaque détection de plan par l'ensemble de plans détectés par le système.
De la même manière, pour l'estimation de la pose, la somme des durées d'estimation est divisée par le nombre de poses calculées. Nous considérons que le traitement commence quand les points d'intérêt appariés sont utilisés pour générer une estimation initiale et se termine avec la dernière itération du raffinement de cette estimation par RANSAC.

### 4.3.3 Résultats

Le tableau 4.6 présente une comparaison de l'erreur des trajectoires estimées par notre approche avec l'implémentation originale du SLAM RGB-D. Les résultats affichent des erreurs comparables pour les deux approches. Cependant, notre système basé sur les points d'intérêt 3D planaires n'utilise en moyenne qu'un quart des points d'intérêt bruts utilisés par l'approche d'origine pour l'estimation de la pose. Cela montre que ces primitives, générés à partir de la projection des points d'intérêt 3D dans leur modèles de plans, offre une précision similaire à celle des points 3D bruts même avec une faible population. Ces résultats confirment nos hypothèses quant à la fiabilité des points 3D planaires et montrent qu'ils peuvent être utilisés pour générer des trajectoires précises.

Pour la séquence "fr3/structure texture far", principalement composée de plans, un petit gain par rapport à l'approche native peut être constaté. La scène est constituée de panneaux en bois et d'un sol recouvert de plusieurs affiches

**TABLE 4.6 –** *Comparaison entre l'erreur absolue de la trajectoire ATE de notre contribution avec le RGB-D SLAM original.*

| Séquence | fr1 xyz | fr1 floor | fr2 large no loop | fr3 structure texture far |
|---|---|---|---|---|
| RGB-D SLAM | 0.01m | 0.04m | 0.86m | 0.039m |
| Notre contribution | 0.02m | 0.05m | 0.83m | 0.036m |

présentant une forte texture.

**TABLE 4.7 –** *Séquence "fr3/structure texture far" utilisée dans l'évaluation de notre approche. [91]*

| Séquence fr3/s.t.far | Caractéristiques | Durée : 31.55s, Dimension : 1.90m x 4.56m x 0.08m |
|---|---|---|
| | Vitesse moyenne de | translation : 0.193m/s, rotation : 4.323deg/s |

Le résultat obtenu dans cette scène permet de prédire le comportement de notre système dans de telles scènes qui pourrait générer des estimations de poses précises. La figure 4.6 montre un exemple de trajectoire générée à partir de cette séquence avec la vérité terrain correspondante (obtenue par un tracking externe, voir section 2.2.3 page 51).

Nous avons ensuite évalué les temps d'exécution de l'extraction des plans 3D et de l'estimation de pose en comparant nos résultats au système d'origine et aux approches proposées dans [38] et [92].

Le tableau 4.8 présente le temps de traitement moyen pour l'extraction des plans et l'estimation de mouvement pour toutes ces approches. Nous nous sommes servi de la même séquence d'évaluation utilisée dans [38] ("fr2/pioneer slam"). Dans [92] les auteurs ont évalué le système en utilisant leur propre scène et ont présenté leurs meilleurs résultats. En comparant les résultats du tableau, nous observons que notre méthode réduit le temps de calcul de l'estimation du mouvement 3D par rapport aux approches comparées. En effet, même si la détection de plans rajoute un traitement supplémentaire au système, nous

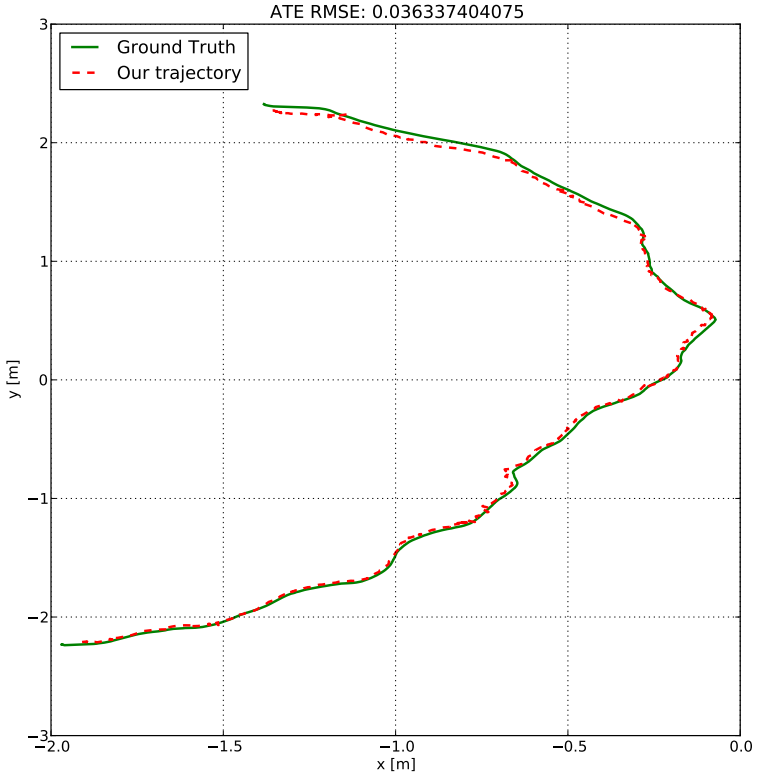



**FIGURE 4.6** – *Trajectoire estimée et Vérité terrain. Séquence fr3/structure texture far.*

gagnons significativement lors de l'estimation de poses en raison du nombre réduit de points d'intérêt 3D planaires utilisés dans cette estimation. La réduction du temps d'exécution (pour le calcul de la pose) provient principalement du petit nombre de points d'intérêt 3D utilisés pour estimer les poses. Avec les résultats de la précision des trajectoires, présentés dans le tableau précédent, notre système peut donc être considéré comme un bon choix pour les scènes composées par des plans.

**TABLE 4.8** – *Temps d'exécution en secondes.*

| Temps d'exécution | Endres [32] | Notre contribution | Gao [38] | Taguchi [92] |
|---|---|---|---|---|
| Extraction de Plans | * | 0.0941 | 0.0181 | 0.210 |
| Estimation de Poses | 0.566 | 0.0057 | 0.0290 | 0.138 |

## 4.4 Séquence Bureau

Dans les expériences de cette section, les données ont été acquises et utilisées pour la localisation et la reconstruction de l'environnement en temps réel. La Kinect a été montée sur un support mobile lui permettant d'être placée partout dans la scène bureau (image 4.7). Cette scène, principalement composée de plans, a été utilisée pour l'évaluation des performances de notre système. Les différentes zones planaires sont réparties dans la scène de façon à créer des discontinuités et des intersections de plans. Lors des expériences de cette section, différentes dispositions de ces plans dans la scène ont été utilisées. Nous avons recouvert ces plans avec des images et des posters pour créer des textures, nécessaires pour la détection des points d'intérêt 2D et la génération des points d'intérêt 3D planaires, et ainsi pouvoir estimer la pose lors des déplacements de la caméra.

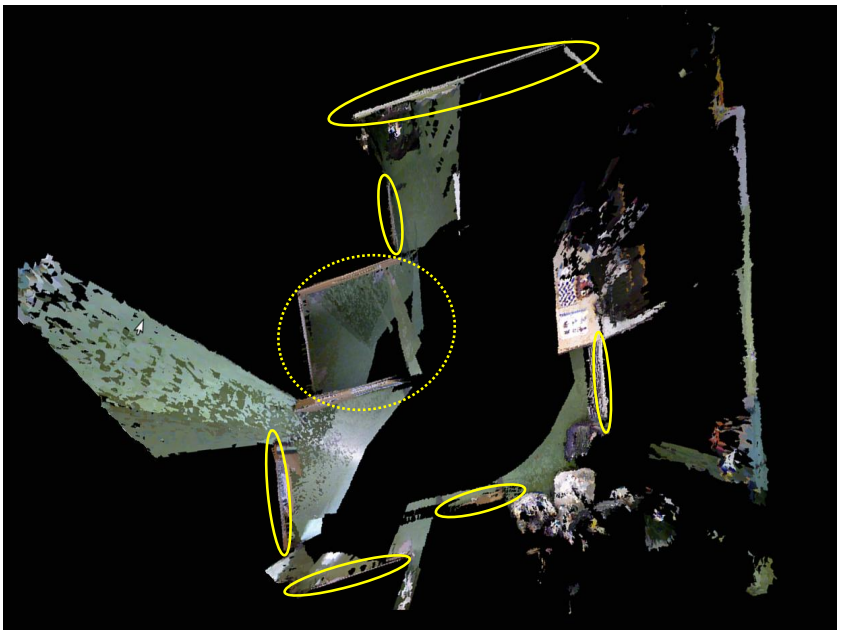
### 4.4.1 Représentation Cartographique

#### 4.4.1.1 Carte 3D basée-points

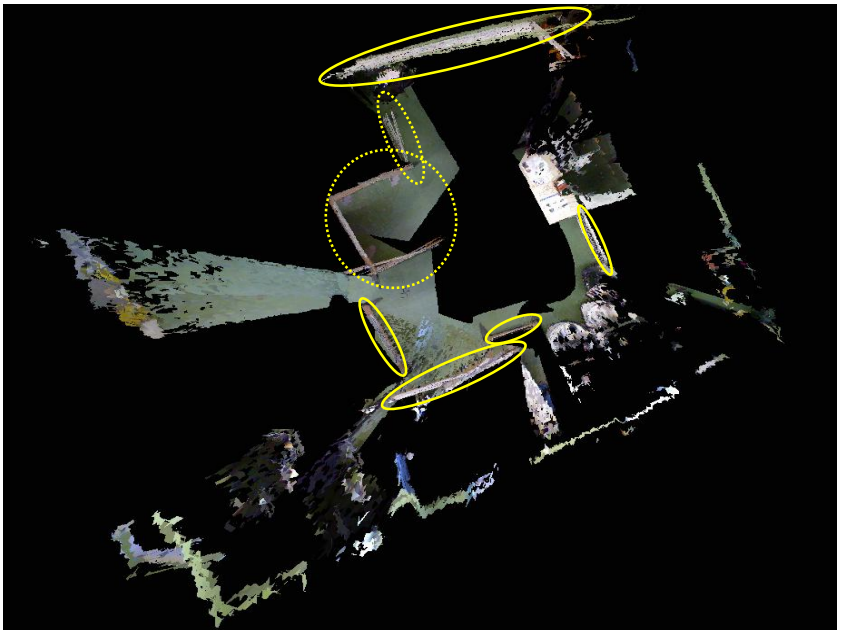
Dans nos études préliminaires, nous avons proposé une première approche [30] étudiant l'impact de l'utilisation des points 3D coplanaires au lieu des points 3D bruts sur la construction de la carte globale, notamment les structures planes, avant même de proposer la carte basée-plans plutôt que basée-points. L'idée de cette première contribution était de modifier les positions des points 3D appartenant aux plans dans la carte brute pour construire des structures planaires continues au lieu de points 3D bruts dispersés. En utilisant le processus de fusion des plans, nous avons proposé d'améliorer la coplanarité des structures en projetant les points coplanaires dans le modèle du plan résultant dès le calcul de la pose. La figure 4.8 montre le résultat de cette technique en comparant la carte brute générée par le RGB-D SLAM natif avec notre carte modifiée. Les emplacements des plans sont entourés en jaune dans les deux cartes. Les régions planaires de notre carte représentent la projection de leurs inliers dans les modèles de plans fusionnés. Comme la carte globale est construite par l'alignement des nuages de points successifs, à travers les transformations estimées, la projection des points coplanaires permet de réduire l'impact des erreurs de ces transformations sur les régions planaires de la carte. Cette méthode a donné


**FIGURE 4.7** – *Notre scène expérimentale Bureau.*

plus de compacité aux régions planaires dans notre carte contrairement à la carte brute où les discontinuités des régions planaires sont plus évidentes. Le résultat montre que nous avons réussi à générer des plans bien distincts dans la carte. Cependant, cette solution ne permet pas de résoudre les difficultés principales liées à l'utilisation de ce genre de représentation, basée-points, à savoir la grande taille de la carte et la difficulté de son exploitation. Ainsi, nous avons choisi de l'abandonner par la suite pour chercher une représentation plus légère et plus sémantique de l'environnement.



***(a)*** *Carte basée-points générée par notre système.*



***(b)*** *Carte basée-points générée par le RGB-D SLAM natif.*

**FIGURE 4.8** – *Comparaison des régions planaires dans la carte brute du RGB-D SLAM et notre carte.*

#### 4.4.1.2 Carte 3D basée-Plans

La figure 4.9 présente un exemple d'une carte 3D basée-plans produite par notre système. Cette carte représente aussi les intersections entre les plans générés. Chaque plan de la carte a été généré progressivement par le processus de fusion, décrit dans le troisième chapitre (section 3.3.3 page 74 ). Les intersections théoriques entre les plans adjacents ont été également mises à jour après chaque changement du modèle d'un plan. Profitant d'une représentation minimale, nous avons généré une carte globale 3D compacte et légère basée sur les plans 3D de l'environnement contrairement à la représentation lourde basée-points. Cette carte offre une information élaborée sur la scène exploitée et représente une première étape vers une carte sémantique. L'exploitation des plans 3D de la carte peut servir à définir des lieux ou des objets de la scène. Vu sa légèreté, elle peut être utilisée par des robots mobiles pour des tâches complexes telle que la navigation.

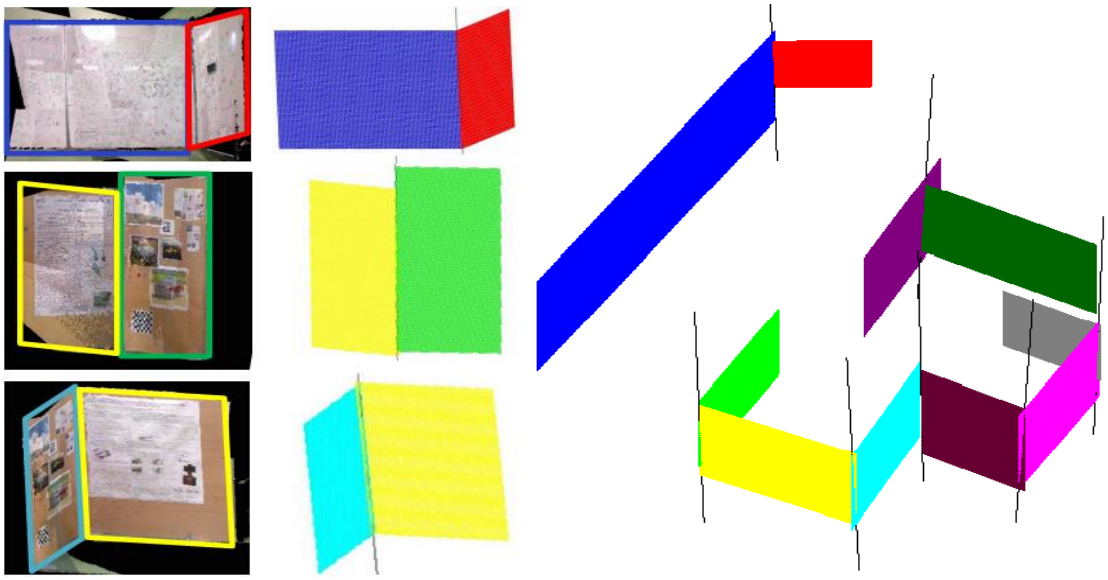
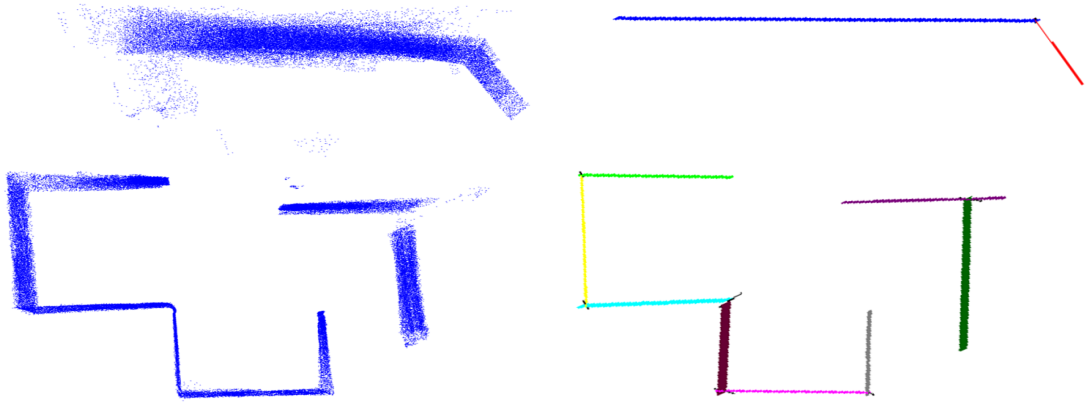


**FIGURE 4.9 –** *Exemple d'une carte 3D planaire générée par notre système.* ***(à gauche)*** *Modèles de plans et leurs nuages de points correspondants.* ***(à droite)*** *Notre carte planaire avec les intersections de plans.*

#### 4.4.1.3 Comparaison des représentations de la carte

Pendant la construction de la carte planaire, nous avons conservé la carte brute des points afin de comparer les deux représentations. La figure 4.10 pré-

sente une vue de dessus des deux types de cartes générées au cours de la même expérience. La carte brute 4.10a contient 1.363.200 points 3D sachant que les nuages utilisés sont sous-échantillonnés par le facteur $\lambda = 4$ définit précédemment (section 4.2 page 85). Une telle représentation conduit à une consommation plus importante de l'espace mémoire (12 mégaoctet), alors que la carte basée-plans générée par notre approche n'occupe que 2 kilo octets de la mémoire. En plus de son poids élevé, le manque d'information sémantique de la scène rend la carte brute difficilement exploitable pour la navigation mobile par exemple. Notre carte basée-plans peut être utilisée par la suite comme base pour construire des cartes à contenu sémantique plus conséquent, permettant par exemple de labelliser les lieux (pièces) où leur contenu (porte, armoire, murs).



***(a)*** *Carte brute basée-points* ***(b)*** *Carte basée-plans*

**FIGURE *4.10*** – *Comparaison des représentations basée-points et basée-plans. (Vue du dessus)*

Au delà de la légèreté, une telle carte peut être plus facilement instrumentée qu'une carte basée points. La figure 4.11 montre une vue approchée du plan coloré en bleu dans la figure 4.10b avec les deux représentations basée-points (en haut) et basée-plans (en bas). La carte brute est représentée par de nombreux points dispersés et qui se chevauchent en raison de l'assemblage de plusieurs nuages de points alors que notre carte affiche une structure planaire 3D continue, légère et prête pour une utilisation sémantique.

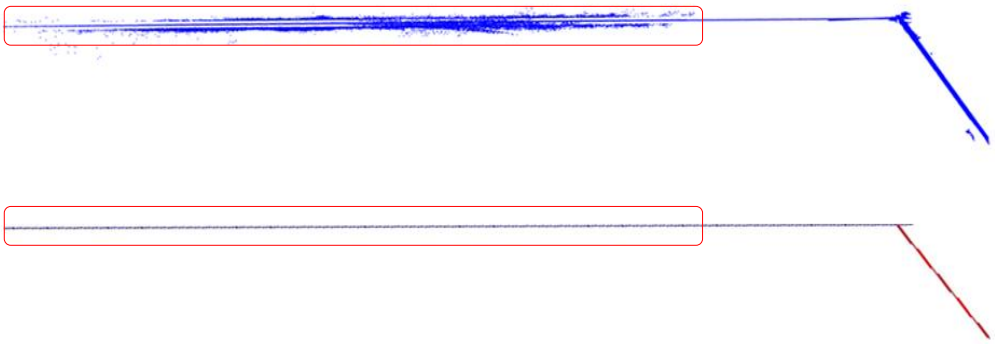


**FIGURE 4.11 –** *Vue approchée d'un plan de la carte 3D : basée-points* ***(en haut)*** *et basée-plans* ***(en bas)***

### 4.4.2 Estimation de la pose

#### 4.4.2.1 Objectifs

Dans la section 4.3.3 nous avons évalué les trajectoires estimées de notre approche en utilisant les séquences de référence de l'université Technologique de Munich (TUM). Les évaluations ont montré que nous avons réduit de manière significative la durée de l'estimation, tout en gardant une bonne précision des trajectoires. Nous avons complété cette étude par d'autres expérimentations menées dans notre scène bureau en utilisant de nouveaux protocoles qui permettent d'évaluer les estimations de poses générées par le système. L'objectif est de pouvoir juger des performances d'un tel système sur une scène réelle sans avoir besoin d'un tracking externe.

#### 4.4.2.2 Protocoles

Nous avons tout d'abord étudié les erreurs de rotations estimées par notre système par rapport à une vérité terrain fournie par un trépied gradué permettant d'effectuer des rotations autour de trois axes. La figure 4.12 montre le dispositif expérimental utilisé dans cette évaluation qui est composé par la caméra Kinect fixée sur le trépied. Nous avons installé ce dispositif en face d'un plan de la scène afin d'évaluer l'utilisation des points d'intérêt 3D planaires, générés par la projection des points d'intérêt 3D bruts dans le modèle de ce plan, dans l'estimation de la rotation. Pour pouvoir juger de la qualité des rotations estimées par notre système, nous avons procédé à plusieurs rotations en utilisant les graduations du trépied comparées aux rotations estimées par


**FIGURE 4.12** – *Dispositif expérimental pour l'évaluation des rotations estimées par notre système.*

le système. Partant du premier nuage de points reçu comme étant la première pose, une nouvelle pose est enregistrée à chaque rotation. La transformation estimée, particulièrement la rotation, est alors comparée à la rotation réelle. Pour valider notre étude, nous avons effectué 30 mesures pour chaque n-degrés de rotation réelle et étudié les rotations estimées de la pose, la moyenne et l'écart-type. D'autre part, afin de définir les conditions de fonctionnement et les limites

de notre système, nous avons également fait varier la distance de la caméra par rapport au premier plan pour étudier son impact sur les rotations estimées.

Nous avons ensuite élaboré un nouveau protocole expérimental [28], consistant à revisiter des endroits particuliers de la scène à plusieurs reprises dans le but d'évaluer la précision des poses estimées. Comme nous ne disposons pas d'un système de suivi externe, nous avons considéré le premier passage par un emplacement comme la vérité terrain et étudié les poses obtenues lors de passages ultérieurs par cet emplacement. Pour cela, nous avons choisi cinq emplacements dispersés dans la scène (les croix dans la figure 4.14b page 4.14) et réalisé une série d'expériences où la caméra a été déplacée à travers ces endroits. La figure 4.13 montre des exemples de ces emplacements d'évaluation dans notre scène réelle. Les marquages au sol (en haut) correspondent aux indicateurs des emplacements, alors que les plans (en bas) représentent la partie de la scène visualisée par la caméra au passage par l'un de ces emplacements.

**FIGURE 4.13** – *Exemples des emplacements d'évaluation dans notre scène bureau.*

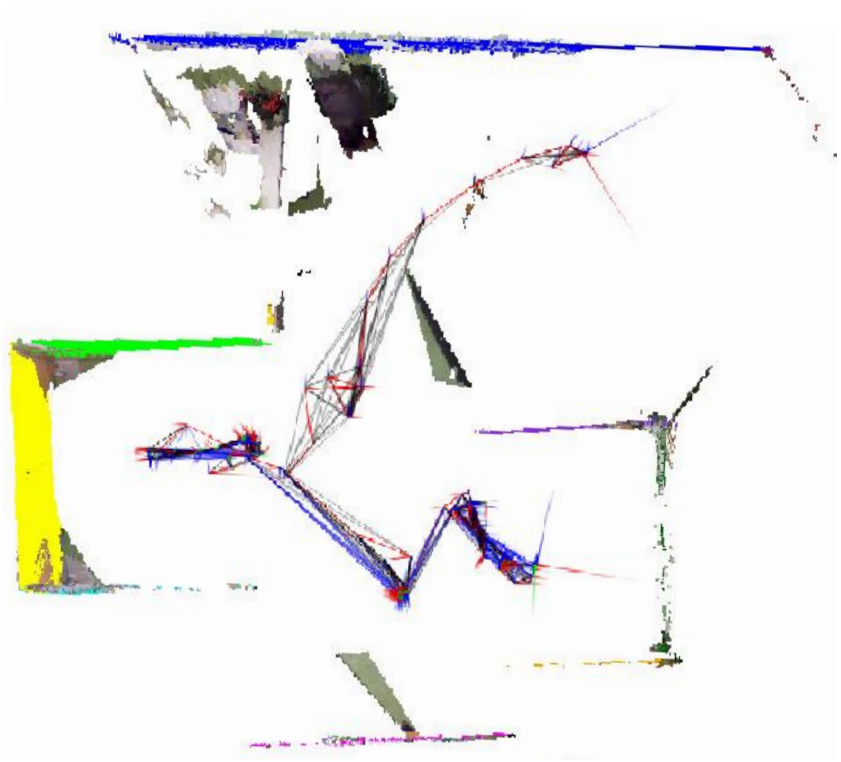
Ainsi, à chaque passage par un emplacement d'évaluation, nous avons enregistré la pose estimée. Pour évaluer la robustesse à l'erreur de notre système, nous avons effectué plusieurs déplacements aléatoires, de la caméra, avant de passer d'un emplacement d'évaluation à l'autre. Cependant, afin d'estimer l'er-

reur absolue de la trajectoire (ATE) (section 2.2.3 page 51), nous avons procédé à un passage successif par l'ensemble des emplacements d'évaluation à chaque fois, c.à.d le retour à un emplacement ne peut être effectué qu'après transition par tout les autres emplacements consécutivement. L'erreur relative de pose (RPE) représente l'erreur locale entre chaque pose estimée et la première pose correspondante enregistrée à cet emplacement. Les trajectoires évaluées correspondent alors aux 5 poses estimées lors de chaque passage par les 5 emplacements d'évaluation consécutifs. Notre protocole d'évaluation peut être très utile lors de l'indisponibilité d'un dispositif externe pour la vérité terrain. En plus de l'évaluation des estimations dans les emplacements revisités, la fermeture de la boucle est également vérifiée et par conséquent la précision des poses estimées par le système.

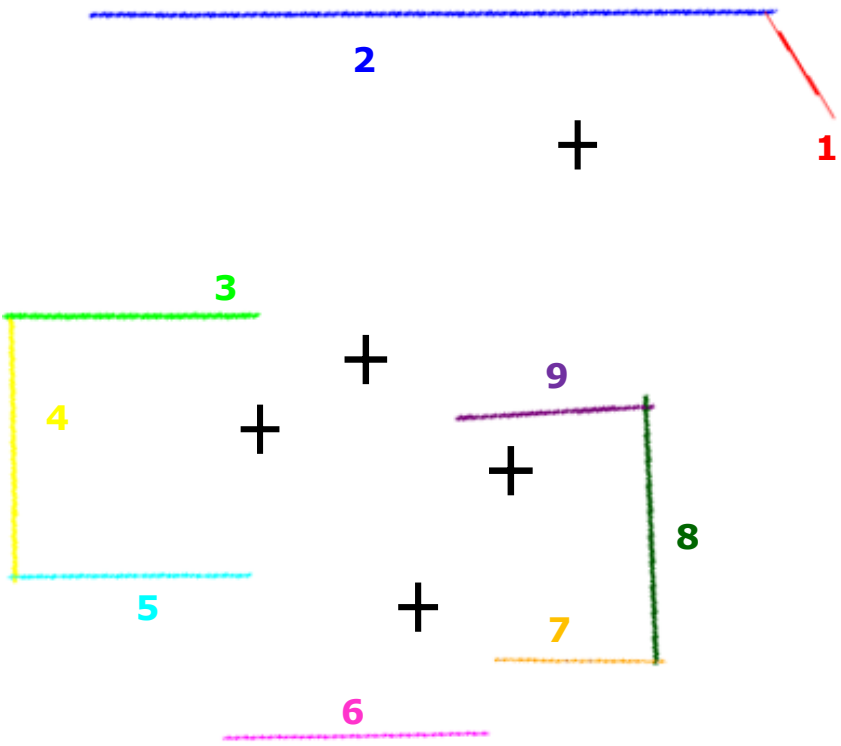
#### 4.4.2.3 Résultats

Le tableau 4.9 présente les résultats des expérimentations comparant les rotations estimées par notre système par rapport à celles mesurées sur le trépied gradué. Notons que l'erreur de mesure sur le trépied est de l'ordre d'un degré. Pour pouvoir estimer des transformations précises, le nombre minimal de points d'intérêt correspondants dans les ensembles de points détectés dans les deux poses a été fixé à 8. En effet, même si, théoriquement, trois points appariés sont suffisants pour estimer une transformation, les expériences préliminaires que nous avons réalisé ont montrées qu'en dessous de huit appariements, la transformation estimée est généralement erronée. Dans les expériences, la rotation limite permettant l'obtention de ce nombre minimal est de 45 degrés. Généralement, les résultats montrent une bonne précision d'estimation compte tenu de l'erreur du trépied. Cependant, comme précédemment, l'erreur de la rotation estimée est proportionnelle à la distance et au degré de la rotation effectuée. Les estimations effectuées sur des plans proches de la distance limite de détection (estimée précédemment à $3.5m$) sont très sensibles aux rotations de la caméra qui ne doivent pas dépasser 20 degrés. Notons que l'estimation de la pose devient impossible pour des rotations supérieures ou égale à 45 degrés, en face d'un plan très éloigné de la caméra, car de telles rotations conduisent à détecter d'autres plans de la scène contenants des points d'intérêt qui ne peuvent pas être matchés à ceux du premier plan détecté.

Le tableau 4.10 résume les résultats de l'évaluation de l'erreur absolue de la



***(a)*** *Carte basée-points avec la trajectoire dedans.*




***(b)*** *Carte basée-plans avec les emplacements d'évaluation (croix).*

**FIGURE *4.14*** **–** *Vue du dessus des reconstructions de notre scène bureau.*

**TABLE 4.9 –** *Résultats des rotations estimées par notre système.*

| Rotation réelle (Degrés) | Rotation estimée (Moy.±Écart-type.) | | |
|---|---|---|---|
| | 1.25(m) | 2.1(m) | 3.3(m) |
| $5° \pm 1°$ | $5.1° \pm 0.4°$ | $4.8° \pm 0.4°$ | $4.7° \pm 0.7°$ |
| $10° \pm 1°$ | $9.3° \pm 0.5°$ | $10.7° \pm 0.9°$ | $10.4° \pm 0.9°$ |
| $20° \pm 1°$ | $19.4° \pm 0.3°$ | $19.6° \pm 0.6°$ | $19.1° \pm 1.2°$ |
| $30° \pm 1°$ | $30.5° \pm 0.7°$ | $30.7° \pm 1.1°$ | $29.9° \pm 2.3°$ |
| $45° \pm 1°$ | $44.0° \pm 0.3°$ | $46.9° \pm 1.9°$ | * |

trajectoire (ATE) et l'erreur relative de la pose (RPE) suivant le protocole expérimental que nous avons élaboré, où la vérité terrain utilisée est le premier passage par les emplacements considérés. Pour pouvoir évaluer la précision des poses estimées chacun de ces emplacements a été revisité 20 fois au cours de l'expérience. Ces passages multiples permettent d'avoir plusieurs estimations pour la même pose réelle et ainsi de mesurer l'erreur moyenne. En outre, lors du déplacement dans la scène nous avons visité différentes zones pour enregistrer d'autres poses de façon à augmenter la probabilité de l'erreur. Les résultats du tableau montrent des erreurs minimales de la trajectoire estimée pour l'erreur absolue (ATE) aussi bien que pour l'erreur relative (RPE). Ces erreurs minimales indiquent que notre système parvient à reconnaître avec une bonne précision les emplacements d'évaluation à chaque passage ce qui prouve que les poses estimées lors des déplacements sont précises. Cela confirme la fiabilité des points d'intérêt 3D planaire et rejoint les conclusions de la section précédente (Benchmarks TUM) concernant l'utilisation de ces points pour l'estimation robuste de la pose.

**TABLE 4.10 –** *Les erreurs ATE et RPE de l'estimation de poses.*

| | |
|---|---|
| ATE (m) | $0.006 \pm 0.003$ |
| RPE - Translation(m) | $0.014 \pm 0.002$ |
| RPE - Rotation(deg) | $1.30° \pm 0.54°$ |

### 4.4.3 Évaluation de la Carte

#### 4.4.3.1 Objectifs

Les plans 3D de la carte générée par notre système sont ajoutés et mis à jour au fil du temps ce qui amène à ajuster itérativement leurs paramètres (orienta-

tion, distance) lorsque de nouveaux points coplanaires sont détectés. Une erreur dans l'estimation de ces paramètres peut engendrer un dysfonctionnement de l'application l'utilisant. Par exemple, si un robot mobile ne dispose pas d'un outil de détection d'obstacles, une mesure erronée de la distance d'un plan peut résulter en une collision. Ainsi, il est important d'évaluer la qualité de la carte construite par rapport à la scène réelle. Nous avons étudié sa fidélité et la précision des plans estimés dans notre carte en utilisant les relations géométriques mesurées entre les plans de la scène réelle.

#### 4.4.3.2 Protocoles

La méthode proposée pour l'évaluation consiste à examiner les distances et les angles entre les plans de la carte. Une reconstruction fidèle de la scène, dans la carte basée-plans, conduirait à des mesures comparables à celles de la scène réelle. Ainsi, nous avons évalué les distances entre les plans parallèles et les angles entre les plans croisés connaissant les plans de la scène. Pour chaque plan de la carte construite, nous disposons de sa distance à l'origine et de son orientation dans le repère global considéré comme le repère du premier nuage de points reçu. Pour trouver la distance entre deux plans il suffit de faire la somme de leurs distances à l'origine si cet origine se trouve entre les deux plans, ou la soustraction autrement. L'angle entre deux plans peut être mesuré en utilisant les vecteurs normaux à ces plans. Notons que, par convention, lors de la détection locale de plans nous avons considéré que la normale au plan est toujours dirigée vers la caméra.

#### 4.4.3.3 Résultats

Les plans utilisés dans ces expériences sont ceux numérotés dans la figure 4.14 (page 107). Les tableaux 4.11 et 4.12 présentent respectivement les mesures des inter-distances et des angles entre les plans estimés. Les résultats sont ainsi confrontés aux valeurs réelles mesurées à partir des plans de la scène de bureau avec une erreur de mesure d'environ $2cm$ pour la distance et de $2^\circ$ pour l'angle. La comparaison entre les mesures réelles et estimées montre que les erreurs ne dépassent pas 6% pour les distances et 1% pour les angles. Ces résultats sont satisfaisants et montrent que notre carte peut convenir à des applications nécessitant une bonne précision. Par conséquent, cette carte légère représente

un bon compromis entre la qualité et la consommation de l'espace mémoire ce qui est requis par des robots mobiles à faible capacité.

**TABLE 4.11 –** *Les distances réelles et estimées entre les plans parallèles.*

| Plan $i$ | Plan $j$ | Distance Réelle (Moy. $\pm$ Écart-type) | Estimation (Moy. $\pm$ Écart-type) |
|---|---|---|---|
| 2 | 3 | $1.50m \pm 0.02m$ | $1.45m \pm 0.06m$ |
| 2 | 5 | $2.82m \pm 0.02m$ | $2.70m \pm 0.08m$ |
| 2 | 6 | $3.42m \pm 0.02m$ | $3.40m \pm 0.06m$ |
| 2 | 7 | $3.20m \pm 0.02m$ | $3.00m \pm 0.10m$ |
| 2 | 9 | $1.95m \pm 0.02m$ | $1.81m \pm 0.10m$ |
| 4 | 8 | $2.95m \pm 0.02m$ | $2.88m \pm 0.05m$ |

**TABLE 4.12 –** *Les angles réel et estimé entre les plans.*

| Plan $i$ | Plan $j$ | Angle réel (Moy. $\pm$ Écart-type) | Estimation (Moy. $\pm$ Écart-type) |
|---|---|---|---|
| 1 | 2 | $56^\circ \pm 2^\circ$ | $56.80^\circ \pm 1.5^\circ$ |
| 3 | 4 | $90^\circ \pm 2^\circ$ | $92.37^\circ \pm 2.1^\circ$ |
| 4 | 5 | $90^\circ \pm 2^\circ$ | $90.12^\circ \pm 0.7^\circ$ |
| 7 | 8 | $90^\circ \pm 2^\circ$ | $92.52^\circ \pm 1.4^\circ$ |
| 8 | 9 | $90^\circ \pm 2^\circ$ | $91.75^\circ \pm 0.9^\circ$ |

Malheureusement, les systèmes similaires à notre approche ne proposent pas d'implémentations accessibles ce qui rend une étude comparative difficile avec ceux-ci vu que nous avons proposé un nouveau protocole pour l'évaluation de la carte.

## 4.5 Conclusion

Dans ce chapitre nous avons présenté les résultats des différentes évaluations de notre méthode RGB-D SLAM basée sur les plans 3D. Nous avons évalué, tout d'abord, la précision des régions planaires détectées. Les expériences ont montré que cette précision est liée au seuil de la tolérance de la détection d'un plan. Si ce seuil est très petit ($1cm$), le processus peut échouer en générant des petits ensembles de points qui ne correspondent pas à des plans réels. Cependant, quand la tolérance de détection est grande ($\geq 2cm$), le plan estimé peut intégrer des points appartenant à d'autres plans de la scène observée ce qui génère ainsi un modèle de plan erroné. En outre, plus la caméra est éloignée des plans plus les erreurs de détection sont importantes en raison du bruit de me-

sure de profondeur. Ainsi, un seuil optimal pour la détection efficace des inliers d'une région planaire a été fixé à l'épaisseur de 1.5$cm$. Il est cependant conseillé de ne pas dépasser une distance de 3$m$ lors de la détection pour garantir de bons résultats.

Nous avons ensuite étudié l'impact de la taille des nuages de points sur la vitesse du système. En effet, l'utilisation d'un nuage complet entraîne le ralentissement du processus de la détection ce qui retarde par conséquent des traitements ultérieurs liés aux résultats de ce processus. Il a donc été nécessaire ainsi de sous-échantillonner les données de profondeur pour pouvoir surmonter cet obstacle sans perdre dans la précision des plans détectés.

Une fois les paramètres garantissant un traitement optimal des données définis, nous avons procédé à l'étude des performances de notre approche. Deux types de séquences ont été utilisées pour cette étude, d'une part les données de référence de TUM [91], et d'autre part une scène bureau capturée en temps réel avec une caméra Kinect. À travers ces données nous avons évalué la précision des trajectoires estimées de la caméra ainsi que la qualité de la carte de l'environnement construite. Les résultats ont montré que nous arrivons à estimer des trajectoires avec la même précision que le système natif malgré le fait que notre méthode n'utilise que les points d'intérêt 3D planaires générés à travers les plans 3D détectés dans les cartes de profondeurs. Cependant, notre estimation de pose est beaucoup plus rapide comparée aux approches similaires. Ceci est dû au nombre réduit de points d'intérêt 3D planaires utilisés dans l'estimation des transformations ce qui montre la fiabilité de ces points et valorise leur utilisation par rapport aux points 3D bruts.

Dans l'évaluation de la carte 3D basée-plans construite par notre système, nous avons comparé cette représentation à la représentation basée-points classique. Notre carte construite par la fusion des modèles planaires détectés dans la scène est beaucoup plus légère que les cartes directes qui cumulent les nuages de points et présentent beaucoup de redondances. En outre, notre représentation compacte de l'environnement fournit une information avancée sur la structure de la scène sans recourir à une approche dense. Elle présente une première étape vers une carte plus sémantique où les plans de la scène pourront être associés à des objets (table, mur,...). Aussi la carte peut servir à générer une représentation topologique où les plans permettront de définir les lieux de la scène.

# Conclusion Générale

Les travaux présentés dans cette thèse ont abordé le problème de Localisation et Cartographie Simultanées (SLAM) dans les environnements d'intérieur. Particulièrement, nous nous sommes intéressé au SLAM visuel utilisant une caméra RGB-D : Le RGB-D SLAM. Notre objectif consistait à concevoir une approche efficace qui offre une représentation légère de l'environnement. En effet, un problème commun aux systèmes RGB-D SLAM existants est la grande taille et la complexité de la carte globale. La solution choisie dans notre thèse consiste à utiliser des structures planaires, majoritaires dans les scènes d'intérieur, pour construire des cartes 3D réduites.

Pour mettre en oeuvre notre solution, nous avons adopté un GraphSLAM basé sur les points d'intérêt. C'est une approche RGB-D SLAM *sparse* utilisant le schéma traditionnel de la détection, description et appariement des points d'intérêt. Elle présente un avantage principal par rapport à notre objectif de thèse, qui est la légèreté de l'implémentation et du traitement de données acquises par la caméra. L'utilisation des points d'intérêt permet de réduire la quantité de données 3D manipulées et stockées par notre système.

L'originalité dans notre approche réside dans l'introduction des plans 3D dans les différents processus du RGB-D SLAM. Par conséquent, la détection de ces plans présente une étape cruciale. Nous avons utilisé l'algorithme RANSAC pour segmenter, de manière incrémentale, les cartes de profondeur en régions planaires.

Pour améliorer l'estimation de pose, nous avons élaboré un mécanisme de pré-traitement de données de profondeur du capteur. Pour estimer une transformation entre deux poses, notre système s'appuie sur les points d'intérêt 3D situés dans les plans détectés localement, au lieu des points d'intérêt 3D bruts, générés par l'association des valeurs de profondeurs bruts aux points d'intérêt visuels 2D correspondants. Ce mécanisme exploite les plans 3D détectés dans

un nuage de points pour améliorer la précision des points d'intérêt 3D et par conséquent l'estimation de pose. Étant donné que ces plans sont estimés à partir d'un ensemble de points 3D coplanaires (un consensus), la probabilité de l'erreur dans les points d'intérêt 3D "planaires" est moins élevée relativement aux points 3D bruts. En outre, une étape de régularisation a été mise en place, consistant à projeter chaque point 3D dans le modèle de plan lui correspondant, pour corriger les erreurs de ces points.

Pour pallier aux défauts des cartes 3D produites par les systèmes RGB-D SLAM (grandes tailles ou redondances de points), nous avons proposé une représentation basée sur les plans 3D de l'environnement. Dans cette représentation, chaque région planaire détectée (ensemble de points 3D coplanaires) est simplifiée en un modèle de plan avec sa boite englobante (points 3D des extrémités). Cela permet de réduire la carte 3D globale résultante qui ne contient que les plans 3D de la scène ainsi que leurs intersections théoriques. Afin de remédier aux éventuelles erreurs d'estimation de poses, nous avons élaboré une méthode de fusion de plans correspondants dans la carte. Si un nouveau plan détecté correspond à un autre plan enregistré dans la carte, ils sont fusionnés en un seul plan résultant. La correspondance de plans est vérifiée par rapport à leurs proximités et leurs orientations. Cette méthode permet de construire itérativement des structures planaires continues et offre une représentation compacte de l'environnement. Par conséquent, la mise à jour de la carte globale est soit le résultat d'une fusion de plans correspondants, soit de l'ajout de nouveaux modèles de plan à cette carte.

Ainsi, nous avons réussi à développer une approche RGB-D SLAM basée uniquement sur des modèles de plans 3D. Afin d'étudier l'efficacité de notre approche, nous avons procédé à une série d'expériences en utilisant des données de références (Benchmarks) pour l'estimation des poses et des données réelles d'une scène expérimentale pour la reconstruction de la carte. Les résultats obtenus pour l'estimation de la pose, en utilisant les points d'intérêt 3D planaires, sont très satisfaisants. Nous avons réussi à produire des trajectoires précises tout en réduisant le temps de calcul de l'estimation de la pose. Dans la carte basée-plans produite par notre système, les modèles de plans remplacent les points 3D. Cette représentation a permis de réduire la taille de la carte et de générer des cartes 3D compactes comparées aux cartes contenant des nuages de points bruts. Les évaluations réalisées ont montré une bonne précision de notre carte par rapport à la vérité terrain ; notamment pour les distances et pour les

angles inter-plans. Par conséquent, elle peut être utilisée par des applications de robotique mobile ou de réalité augmentée.

Dans notre approche, l'hypothèse de base est qu'une scène d'intérieur est constituée majoritairement de portions de plans 3D (tables, murs, portes,...). Cependant, notre système peut rencontrer des difficultés lors de l'absence de structures planaires dans la scène ou le manque de textures sur les plans détectés.

Parmi les perspectives de cette thèse, nous citons les points suivants :

- Une suite logique de nos travaux consiste à construire une carte topologique en exploitant les plans 3D de notre carte. peut être bénéfique pour la construction d'une carte topologique sans avoir recours à la sémantique d'objet, habituellement utilisée pour définir les lieux dans ce type de carte. En partant de l'hypothèse qu'un environnement d'intérieur, une pièce (chambre, bureau,...), est généralement constitué par quatres murs, nous pouvons définir un lieu à partir de l'adjacence de quatre plans et leurs intersections (deux à deux). Ainsi, la carte topologique résultante contiendrait les lieux visités et les chemins entre eux.
- D'autre part, une amélioration envisagée consiste à étiqueter les structures planaires détectées en relation avec les objets réels (tables, portes,...). Cela permettrait d'intégrer des informations sémantiques dans notre carte par la labellisation de plans détectés. En outre, la connaissance de la nature des régions planes peut aider à segmenter de nouveaux objets, si nous considérons que la plupart des objets intéressants ont un plan de support. L'intérêt de l'information sémantique pour la navigation mobile par exemple serait la reconnaissance des lieux visités. Ainsi, la combinaison d'une carte topologique avec la connaissance des objets (labellisation) produirait une carte sémantique.
- Enfin, notre carte 3D peut être enrichie en intégrant d'autres primitives géométriques 3D, elles aussi simplifiées par leurs représentations paramétriques, telles que les sphères, cylindres,.... Par conséquent, la carte basée-primitives 3D fournirait une représentation étendue de l'environnement. Si ces primitives représenteraient une grande partie de l'environnement dans la carte, les points 3D bruts, qui ne sont pas modélisables par une forme géométrique, pourront être réintégrer dans cette carte. Comme ces points 3D seraient minoritaires, la carte résultante ne présenterait pas les inconvénients d'une carte brute (grande taille et redondance des points).

Ainsi, une solution hybride, combinant la carte basée-primitives avec la carte directe (basée-points), est envisageable pour la cartographie dans notre RGB-D SLAM.

# Bibliographie


[1] Motion Capture Systems | VICON. https://www.vicon.com/.

[2] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Richard Szeliski. Building rome in a day. *Commun. ACM*, 54(10) :105–112, October 2011.

[3] Abdennour Aouina, Michel Devy, and Antonio Marin-Hernandez. Comparison of active sensors for 3d modeling of indoor environments. In *Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics*, pages 442–449, 2013.

[4] David Arthur and Sergei Vassilvitskii. Worst-case and smoothed analysis of the ICP algorithm, with an application to the k-means method. In *47th Annual IEEE Symposium on Foundations of Computer Science, 2006. FOCS '06.*, pages 153–164, Berkeley, CA, USA, oct 2006. IEEE.

[5] David Arthur and Sergei Vassilvitskii. Worst-case and smoothed analysis of the ICP algorithm, with an application to the k-means method. *SIAM Journal on Computing*, 39(2) :766–782, July 2009.

[6] E. Ataer-Cansizoglu, Y. Taguchi, S. Ramalingam, and T. Garaas. Tracking an rgb-d camera using points and planes. In *2013 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 51–58, Dec 2013.

[7] S. Y. Bao, M. Bagra, Y. W. Chao, and S. Savarese. Semantic structure from motion with points, regions, and objects. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2703–2710, June 2012.

[8] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3) :346–359, June 2008.

[9] S. Betge-Brezetz, P. Hebert, R. Chatila, and M. Devy. Uncertain map making in natural environments. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 2, pages 1048–1053 vol.2, Apr 1996.

[10] O. Booij, B. Terwijn, Z. Zivkovic, and B. Krose. Navigation using an appearance based topological map. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3927–3932, April 2007.

[11] Jose A. Castellanos and Juan D. Tardos. *Mobile Robot Localization and Map Building : A Multisensor Fusion Approach*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[12] R. Chatila and J. Laumond. Position referencing and consistent world modeling for mobile robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation.*, volume 2, pages 138–145, Mar 1985.

[13] Denis Chekhlov, Andrew P. Gee, Andrew Calway, and Walterio Mayol-Cuevas. Ninja on a plane : Automatic discovery of physical planes for augmented reality using visual slam. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '07, pages 1–4, Washington, DC, USA, 2007. IEEE Computer Society.

[14] Sunglok Choi, Taemin Kim, and Wonpil Yu. Performance evaluation of ransac family. In *Proceedings of the British Machine Vision Conference*, pages 81.1–81.12. BMVA Press, 2009. doi :10.5244/C.23.81.

[15] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized ransac. In *Joint Pattern Recognition Symposium*, pages 236–243. Springer, 2003.

[16] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 303–312, New York, NY, USA, August 1996. ACM.

[17] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1403–1410 vol.2, Oct 2003.

[18] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam : Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6) :1052–1067, June 2007.

[19] Andrew J. Davison and David W. Murray. *Mobile robot localisation using active vision*, pages 809–825. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[20] Frank Dellaert. Factor graphs and gtsam : A hands-on introduction. Technical Report GT-RIM-CP&R-2012-002, GT RIM, Sept 2012.

[21] Frank Dellaert and Michael Kaess. Square root sam : Simultaneous localization and mapping via square root information smoothing. *Int. J. Rob. Res.*, 25(12) :1181–1203, December 2006.

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

[23] Mingsong Dou, Li Guan, Jan-Michael Frahm, and Henry Fuchs. Exploring high-level plane primitives for indoor 3d reconstruction with a handheld rgb-d camera. In *Proceedings of the 11th International Conference on Computer Vision - Volume 2*, ACCV'12, pages 94–108, Berlin, Heidelberg, 2013. Springer-Verlag.

[24] Romain Drouilly. *Cartographie hybride métrique topologique et sémantique pour la navigation dans de grands environnements*. PhD thesis, Université de Nice-Sophia Antipolis, 2015. 2015NICE4037.

[25] Delphine Dufourd. Des cartes combinatoires pour la construction automatique de modèles d'environnement par un robot mobile. November 2005.

[26] Staffan Ekvall, Danica Kragic, and Patric Jensfelt. Object detection and mapping for service robot tasks. *Robotica*, 25(2) :175–187, March 2007.

[27] Oussama El Hamzaoui. *Simultaneous Localization and Mapping for a mobile robot with a laser scanner : CoreSLAM*. Theses, Ecole Nationale Supérieure des Mines de Paris, September 2012.

[28] Hakim ElChaoui ElGhor, David Roussel, Fakhreddine Ababsa, and El-Houssine Bouyakhf. *3D Planar RGB-D SLAM System*, pages 486–497. Springer International Publishing, 2016.

[29] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6) :46–57, June 1989.

[30] Hakim Elchaoui Elghor, David Roussel, Fakhreddine Ababsa, and El Houssine Bouyakhf. Planes detection for robust localization and mapping in rgb-d slam systems. In *Proceedings of the 2015 International Conference on 3D Vision*, 3DV '15, pages 452–459, Washington, DC, USA, 2015. IEEE Computer Society.

[31] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *IEEE International Conference on Robotics and Automation (ICRA 2012)*, pages 1691–1696, May 2012.

[32] Felix Endres, Jurgen Hess, Jurgen Sturm, Daniel Cremers, and Wolfram Burgard. 3D mapping with an RGB-D camera. *IEEE Transactions on Robotics*, 30(1) :177–187, February 2014.

[33] Elisabetta Fabrizi and Alessandro Saffiotti. Augmenting topology-based maps with geometric information. *Robotics and Autonomous Systems*, 40(2-3) :91 – 97, 2002. Intelligent Autonomous Systems - {IAS} -6.

[34] David Filliat and Jean-Arcady Meyer. Map-based navigation in mobile robots : : I. a review of localization strategies. *Cognitive Systems Research*, 4(4) :243 – 282, 2003.

[35] Martin A. Fischler and Robert C. Bolles. Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6) :381–395, June 1981.

[36] A. W. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *European Conference on Computer Vision*, pages 311–326. Springer-Verlag, 1998.

[37] F. Fraundorfer, C. Engels, and D. Nister. Topological mapping, localization and navigation using image collections. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3872–3877, Oct 2007.

[38] Xiang Gao and Tao Zhang. Robust rgb-d simultaneous localization and mapping using planar point features. *Robotics and Autonomous Systems*, 72 :1 – 14, 2015.

[39] A. P. Gee, D. Chekhlov, A. Calway, and W. Mayol-Cuevas. Discovering higher level structure in visual slam. *IEEE Transactions on Robotics*, 24(5) :980–990, Oct 2008.

[40] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2(4) :31–43, winter 2010.

[41] G. Grisetti, C. Stachniss, and W. Burgard. Nonlinear constraint network optimization for efficient map learning. *IEEE Transactions on Intelligent Transportation Systems*, 10(3) :428–439, Sept 2009.

[42] J. Han, L. Shao, D. Xu, and J. Shotton. Enhanced computer vision with microsoft kinect sensor : A review. *IEEE Transactions on Cybernetics*, 43(5) :1318–1334, Oct 2013.

[43] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping : Using depth cameras for dense 3d modeling of indoor environments. In *In the 12th International Symposium on Experimental Robotics (ISER*, 2010.

[44] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D mapping : Using kinect-style depth cameras for dense 3D modeling of indoor environments. *International Journal of Robotics Research (IJRR)*, 31(5) :647–663, April 2012.

[45] Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, and Sven Behnke. Robot soccer world cup xv. In Thomas Röfer, Norbert Michael Mayer, Jesus Savage, and Uluç Saranlı, editors, *Robot Soccer World Cup XV*, chapter Real-time Plane Segmentation Using RGB-D Cameras, pages 306–317. Springer-Verlag, Berlin, Heidelberg, 2012.

[46] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap : An efficient probabilistic 3d mapping framework based on octrees. *Auton. Robots*, 34(3) :189–206, April 2013.

[47] G. Hu, S. Huang, L. Zhao, A. Alempijevic, and G. Dissanayake. A robust rgb-d slam algorithm. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1714–1719, Oct 2012.

[48] Albert S. Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *International Symposium on Robotics Research (ISRR)*, pages 1320–1343, 2011.

[49] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion : Real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 559–568, New York, NY, USA, October 2011. ACM.

[50] Adrian Kaehler and Gary Bradski. *Learning OpenCV, 2Nd Edition*. O'Reilly Media, Inc., 2014.

[51] M. Kaess. Simultaneous localization and mapping with infinite planes. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4605–4611, May 2015.

[52] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *2013 International Conference on 3D Vision - 3DV 2013*, 3DV '13, pages 1–8, Washington, DC, USA, 2013. IEEE Computer Society.

[53] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2) :1437–1454, 2012.

[54] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234, Nov 2007.

[55] Benjamin Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2(2) :129 – 153, 1978.

[56] Benjamin Kuipers and Yung-Tai Byun. Toward learning robots. chapter A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations, pages 47–63. MIT Press, Cambridge, MA, USA, 1993.

[57] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o : A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation (ICRA 2011)*, pages 3607–3613. IEEE, May 2011.

[58] J. J. Leonard and H. F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3) :376–382, Jun 1991.

[59] DavidG. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60 :91–110, 2004.

[60] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4) :333–349, 1997.

[61] Robert Maier, Jürgen Sturm, and Daniel Cremers. *Submap-Based Bundle Adjustment for 3D Reconstruction from RGB-D Data*, pages 54–65. Springer International Publishing, Cham, 2014.

[62] Donald Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2) :129 – 147, 1982.

[63] M. Meilland and A. I. Comport. On unifying key-frame and voxel-based dense visual slam at large scales. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3677–3683, Nov 2013.

[64] Jean-Arcady Meyer and David Filliat. Map-based navigation in mobile robots : : Ii. a review of map-learning and path-planning strategies. *Cognitive Systems Research*, 4(4) :283 – 317, 2003.

[65] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. 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, 2003. IJCAI.

[66] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam : A factored solution to the simultaneous localization and mapping problem. In *Aaai/iaai*, pages 593–598, 2002.

[67] Philippe Moutarlier and Raja Chatila. Stochastic multisensory data fusion for mobile robot location and environment modeling. In *5th International Symposium on Robotics Research*, pages 85–94. Tokyo, 1989.

[68] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *In VISAPP International Conference on Computer Vision Theory and Applications*, pages 331–340, 2009.

[69] Richard Newcombe. *Dense visual SLAM*. PhD thesis, Imperial College London, 2012.

[70] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion : Real-time dense surface mapping and tracking. In *10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2011)*, ISMAR '11, pages 127–136. IEEE Computer Society, October 2011.

[71] Paul Newman. On the structure and solution of the simultaneous localisation and map building problem. *Doctoral diss., University of Sydney*, 41, 1999.

[72] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–652–I–659 Vol.1, June 2004.

[73] S. Park, H. Cheong, and S. K. Park. Coarse-to-fine global localization for mobile robots with hybrid maps of objects and spatial layouts. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3993–4000, Oct 2009.

[74] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels : Surface elements as rendering primitives. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 335–342, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[75] Andrzej Pronobis, Oscar M Mozos, Barbara Caputo, Patric Jensfelt, et al. Multi-modal semantic place classification. *International Journal of Robotics Research (IJRR)*, 29(2-3) :298–320, 2010.

[76] Gustavo A. Puerto-Souza, Alberto Castaño-Bardawil, and Gian-Luca Mariottini. *Real-Time Feature Matching for the Accurate Recovery of Augmented-Reality Display in Laparoscopic Videos*, pages 153–166. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[77] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros : an open-source robot

operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[78] Rahul Raguram, Jan-Michael Frahm, and Marc Pollefeys. A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus. In *Proceedings of the 10th European Conference on Computer Vision : Part II*, ECCV '08, pages 500–513, Berlin, Heidelberg, 2008. Springer-Verlag.

[79] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb : An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, Nov 2011.

[80] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Close-range scene segmentation and reconstruction of 3d point cloud maps for mobile manipulation in domestic environments. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS'09, pages 1–6, Piscataway, NJ, USA, 2009. IEEE Press.

[81] R.B. Rusu and S. Cousins. 3d is here : Point cloud library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA 2011)*, pages 1–4. IEEE, May 2011.

[82] R. F. Salas-Moreno, B. Glocken, P. H. J. Kelly, and A. J. Davison. Dense planar slam. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 157–164, Sept 2014.

[83] Nizar K. Sallem and Michel Devy. Extended grabcut for 3d and rgb-d point clouds. In *15th International Conference on Advanced Concepts for Intelligent Vision Systems - Volume 8192*, ACIVS 2013, pages 354–365, New York, NY, USA, 2013. Springer-Verlag New York, Inc.

[84] Alexander Shpunt and Zeev Zalevsky. Depth-varying light fields for three dimensional sensing. Grant Patent US 8050461 B2, US Patent Office, 13 March 2007. Published Jun. 30 2011.

[85] Alexander Shpunt and Zeev Zalevsky. Three-dimensional sensing using speckle patterns. Patent Grant US 8390821 B2, US Patent Office, 8 March 2007. Published Mar. 5, 2013.

[86] Sudipta N Sinha, Jan-Michael Frahm, Marc Pollefeys, and Yakup Genc. Gpu-based video feature tracking and matching. In *EDGE, Workshop on Edge Computing Using New Commodity Architectures*, volume 278, page 4321, 2006.

[87] Randall C. Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainly. *Int. J. Rob. Res.*, 5(4) :56–68, December 1986.

[88] C. Stachniss, U. Frese, and G. Grisetti. OpenSLAM. `https://openslam.org/`.

[89] H. Strasdat, J. M. M. Montiel, and A. Davison. Scale drift-aware large scale monocular slam. In *Proceedings of Robotics : Science and Systems*, Zaragoza, Spain, June 2010.

[90] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Real-time monocular slam : Why filter ? In *2010 IEEE International Conference on Robotics and Automation*, pages 2657–2664, May 2010.

[91] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, pages 573–580, October 2012.

[92] Y. Taguchi, Yong-Dian Jian, S. Ramalingam, and Chen Feng. Point-plane SLAM for hand-held 3D sensors. In *IEEE International Conference on Robotics and Automation (ICRA 2013)*, pages 5182–5189, May 2013.

[93] S. Thrun and M. Montemerlo. The GraphSLAM algorithm with applications to large-scale mapping of urban structures. *International Journal on Robotics Research*, 25(5/6) :403–430, 2005.

[94] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In *IEEE International Conference on Robotics and Automation (ICRA-2000*, pages 321–328, 2000.

[95] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[96] A. J. B. Trevor, J. G. Rogers, and H. I. Christensen. Planar surface slam with 3d and 2d sensors. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3041–3048, May 2012.

[97] Alexander JB Trevor, Suat Gedikli, Radu Bogdan Rusu, and Henrik I Christensen. Efficient organized point cloud segmentation with connected components. *Semantic Perception Mapping and Exploration (SPME)*, 2013.

[98] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4) :376–380, April 1991.

[99] Trung-Dung Vu. *Vehicle Perception : Localization, Mapping with Detection, Classification and Tracking of Moving Objects*. PhD thesis, Institut National Polytechnique de Grenoble, 2009.

[100] J. Weingarten and R. Siegwart. 3d slam using planar segments. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3062–3067, Oct 2006.

[101] T. Whelan, H. Johannsson, M. Kaess, J.J. Leonard, and J. McDonald. Robust real-time visual odometry for dense RGB-D mapping. In *IEEE International Conference on Robotics and Automation (ICRA 2013)*, pages 5724–5731, May 2013.

[102] T. Whelan, M. Kaess, J. J. Leonard, and J. McDonald. Deformation-based loop closure for large scale dense rgb-d slam. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 548–555, Nov 2013.

[103] T. Whelan, L. Ma, E. Bondarev, P.H.N. de With, and J. McDonald. Incremental and batch planar simplification of dense point cloud maps. *Robot. Auton. Syst.*, 69(C) :3–14, July 2015.

[104] Thomas Whelan, Michael Kaess, Maurice Fallon, Hordur Johannsson, John Leonard, and John McDonald. Kintinuous : Spatially extended kinectfusion. In *RSS Workshop on RGB-D : Advanced Reasoning with Depth Cameras*, 2012.

[105] Zeev Zalevsky, Alexander Shpunt, Aviad Maizels, Javier Garcia, and Others. Method and system for object reconstruction. WO Patent 2007043036, World Intellectual Property Organization, 20 April 2007.

[106] Zeev Zalevsky, Alexander Shpunt, Aviad Malzels, and Javier Garcia. Method and system for object reconstruction. Grant Patent US 8400494 B2, US Patent Office, 14 March 2006. Published Mar. 19 2013.

[107] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2) :119–152, 1994.

[108] S. Zug, F. Penzlin, A. Dietrich, T. T. Nguyen, and S. Albert. Are laser scanners replaceable by kinect sensors in robotic applications ? In *Robotic and Sensors Environments (ROSE), 2012 IEEE International Symposium on*, pages 144–149, Nov 2012.

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

**Titre :**
**Modélisation Planaire pour un RGB-D SLAM: Localisation éparse et Cartographie réduite**

**Keywords :** SLAM temps-réel, caméra RGB-D, GraphSLAM, Segmentation de données 3D, Points d'intérêt 3D planaires, Cartes 3D basée-plans.

**Résumé :** Cette thèse traite du problème de la Localisation et Cartographie Simultanées (SLAM) dans les environnements d'intérieur. Dans ce contexte, nous avons choisi un SLAM visuel en utilisant les données d'un capteur RGB-D de type Kinect pour estimer la trajectoire de la caméra et construire une carte 3D de l'environnement en temps réel. Malgré les avantages des caméras RGB-D (faible coût, images couleurs et cartes de profondeur), les données de profondeur issues de ce genre de capteur peuvent être de mauvaise qualité ce qui affecte l'estimation de la pose. En outre, la taille des nuages de points engendre une carte globale lourde et contenant de nombreux points 3D redondants. Afin de diminuer l'impact de ces faiblesses sur la résolution du SLAM, nous proposons d'utiliser des plans 3D, majoritaires dans les scènes d'intérieur, dans le processus d'estimation de poses de la caméra pour construire des cartes 3D basées-plans. Les plans 3D servent alors à générer des points d'intérêt 3D planaires moins bruités que les points bruts déduits directement des nuages de points. En rectifiant les valeurs de profondeur des points d'intérêt 3D bruts appartenant à ces plans, nous améliorons ainsi l'estimation de pose quand la scène est composée essentiellement de plans. Par la suite, les plans 3D détectés sont utilisés pour construire une carte 3D globale légère. La carte est élaborée en fusionnant itérativement les régions planaires détectées dans la scène avec celles déjà présentes dans la carte ou en ajoutant de nouveaux plans. Contrairement à la représentation classique basée-point, nous réduisons ainsi la taille de la carte 3D et construisons des cartes compactes. Ces cartes sont exploitables par des applications de robotique mobile et de navigation. Pour montrer les bénéfices des travaux proposés dans cette thèse, les expérimentations réalisées évaluent la précision de la localisation, l'influence de l'échantillonnage des données RGB-D sur la détection des plans ainsi que la qualité de la carte basée-plans 3D par rapport à la scène réelle. La carte ainsi constituée de plans présente une première étape vers une carte plus sémantique.

**Université Paris-Saclay**
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

**Title :**
**Planar Modeling for an RGB-D SLAM: sparse Localisation and reduced Mapping**

**Keywords :** Real-time SLAM, Kinect camera, GraphSLAM, Plane detection, 3D planar features, 3D plane-based Maps.

Abstract : This thesis deals with the Simultaneous Localisation and Mapping (SLAM) problem in indoor environments. In this context, we chose a visual SLAM using an RGB-D sensor (Kinect) to estimate the camera trajectory and to build a 3D map of the environment in real time. Despite RGB-D cameras advantages (low cost, color images and depth maps), depth data resulting from this kind of sensors may be noisy, which affects pose estimation. In addition, due to points clouds sizes, the resulting global map is heavyweight and contains many redundant 3D points. In order to reduce the impact of these weaknesses on resolving the SLAM problem, we propose to use 3D planes, which are dominant in indoor scenes, for both camera poses estimations and 3D based-planes maps building process. Hence, 3D planes are used to generate 3D planar feature featuring less depth noise than the raw points extracted directly from points clouds. By regularizing depth values of raw 3D feature points belonging to these planes, we improve pose estimation when the scene is mainly composed of planes. Then, the detected 3D planes are used to build the global 3D map, creating a light representation of the environment based on these planes. The map is iteratively built from each new camera pose either by merging new planes to the existing ones or by adding new planes to the map. Thus, unlike conventional point-based representation, the size of the resulting 3D map is considerably reduced and the built map is more compact compared to point-based maps.These maps may be used by mobile robotics and navigation applications. To show the benefits of our works, the conducted experiments to evaluate localisation accuracy, the influence of subsampled RGB-D data on plane detection, as well as quality of 3D plane-based maps against real scenes. Such plane-based maps represents a first step towards semantic maps.

**Université Paris-Saclay**
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France