Contributions au RGBD-SLAM

Kathia Melbouci

N° d'ordre : D.U. 2798
N° EDSPIC : 793

UNIVERSITÉ CLERMONT AUVERGNE

École Doctorale
Sciences Pour l'Ingénieur de Clermont-Ferrand

Thèse présentée par :
Kathia MELBOUCI

Formation Doctorale :
Électronique et Système

en vue de l'obtention du grade de

DOCTEUR D'UNIVERSITÉ

Spécialité : Vision pour la Robotique

Contributions au RGBD-SLAM

Soutenue publiquement le 2 Mars 2017 devant le jury :

M. Malik MALLEM	Président du jury
Mme. Sylvie TREUILLET	Rapporteur et examinateur
M. El Mustapha MOUADDIB	Rapporteur et examinateur
M.Thierry CHATEAU	Examinateur
Mme. Sylvie NAUDET	Encadrant
M. Vincent GAY-BELLILE	Encadrant
M. Omar AIT-AIDER	Encadrant
M. Michel DHOME	Directeur de thèse

Résumé

Pour assurer la navigation autonome d'un robot mobile, les traitements effectués pour sa localisation doivent être faits en ligne et doivent garantir une précision suffisante pour permettre au robot d'effectuer des tâches de haut niveau pour la navigation et l'évitement d'obstacles. Les auteurs de travaux basés sur le SLAM visuel (Simultaneous Localization And Mapping) tentent depuis quelques années de garantir le meilleur compromis rapidité/précision. La majorité des solutions SLAM visuel existantes sont basées sur une représentation éparse de l'environnement. En suivant des primitives visuelles sur plusieurs images, il est possible d'estimer la position 3D de ces primitives ainsi que les poses de la caméra. La communauté du SLAM visuel a concentré ses efforts sur l'augmentation du nombre de primitives visuelles suivies et sur l'ajustement de la carte 3D, afin d'améliorer l'estimation de la trajectoire de la caméra et les positions 3D des primitives. Cependant, la localisation par SLAM visuel présente souvent des dérives dues au cumul d'erreurs, et dans le cas du SLAM visuel monoculaire, la position de la caméra n'est connue qu'à un facteur d'échelle près. Ce dernier peut être fixé initialement mais dérive au cours du temps.

Pour faire face à ces limitations, nous avons centré nos travaux de thèse sur la problématique suivante : intégrer des informations supplémentaires dans un algorithme de SLAM visuel monoculaire afin de mieux contraindre la trajectoire de la caméra et la reconstruction 3D. Ces contraintes ne doivent pas détériorer les performances calculatoires de l'algorithme initial et leur absence ne doit pas mettre l'algorithme en échec. C'est pour cela que nous avons choisi d'intégrer l'information de profondeur fournie par un capteur 3D (*e.g.* Microsoft Kinect) et des informations géométriques sur la structure de la scène.

La première contribution de cette thèse est de modifier l'algorithme SLAM visuel monoculaire proposé par Mouragnon et al. (2006b) pour prendre en compte la mesure de profondeur fournie par un capteur 3D, en proposant particulièrement un ajustement de faisceaux qui combine, d'une manière simple, des informations visuelles et des informations de profondeur. La deuxième contribution est de proposer une nouvelle fonction de coût du même ajustement de faisceaux qui intègre, en plus des contraintes sur les profondeurs des points, des contraintes géométriques d'appartenance aux plans de la scène.

Les solutions proposées ont été validées sur des séquences de synthèse et sur des séquences réelles, représentant des environnements variés. Ces solutions ont été comparées aux récentes méthodes de l'état de l'art. Les résultats obtenus montrent que les différentes contraintes développées permettent d'améliorer significativement la précision de la localisation du SLAM. De plus les solutions proposées sont faciles à déployer et peu couteuses en temps de calcul.

Mots clés : Localisation et cartographie simultanées par vision, capteur 3D, ajustement de faisceaux, plans, RGBD-SLAM.

Abstract

To guarantee autonomous and safely navigation for a mobile robot, the processing achieved for its localization must be fast and accurate enough to enable the robot to perform high-level tasks for navigation and obstacle avoidance. The authors of Simultaneous Localization And Mapping (SLAM) based works, are trying since year, to ensure the speed/accuracy trade-off.

Most existing works in the field of monocular (SLAM) has largely centered around sparse feature-based representations of the environment. By tracking salient image points across many frames of video, both the positions of the features and the motion of the camera can be inferred live. Within the visual SLAM community, there has been a focus on both increasing the number of features that can be tracked across an image and efficiently managing and adjusting this map of features in order to improve camera trajectory and feature location accuracy. However, visual SLAM suffers from some limitations. Indeed, with a single camera and without any assumptions or prior knowledge about the camera environment, rotation can be retrieved, but the translation is up to scale. Furthermore, visual monocular SLAM is an incremental process prone to small drifts in both pose measurement and scale, which when integrated over time, become increasingly significant over large distances.

To cope with these limitations, we have centered our work around the following issues : integrate additional information into an existing monocular visual SLAM system, in order to constrain the camera localization and the mapping points. Provided that the high speed of the initial SLAM process is kept and the lack of these added constraints should not give rise to the failure of the process. For these last reasons, we have chosen to integrate the depth information provided by a 3D sensor (*e.g.* Microsoft Kinect) and geometric information about scene structure.

The primary contribution of this work consists of modifying the SLAM algorithm proposed by Mouragnon et al. (2006b) to take into account the depth measurement provided by a 3D sensor. This consists of several rather straightforward changes, but also on a way to combine the depth and visual data in the bundle adjustment process. The second contribution is to propose a solution that uses, in addition to the depth and visual data, the constraints lying on points belonging to the plans of the scene.

The proposed solutions have been validated on a synthetic sequences as well as on a real sequences, which depict various environments. These solutions have been compared to the state of art methods. The performances obtained with the previous solutions demonstrate that the additional constraints developed, improves significantly the accuracy and the robustness of

the SLAM localization. Furthermore, these solutions are easy to roll out and not much time consuming.

Key-words: Simultaneous Localisation and Mapping, 3D sensor, bundle adjustment,plans, RGBD-SLAM.

Notations et Acronymes

Mathématiques

$\mathbb{R}^n$	Espace R de dimension n
M	Matrice
$\mathbf{v}$	Vecteur
$[\mathbf{t}]_\times$	Matrice antisymétrique créée à partir du vecteur $\mathbf{t}$
M^+	Pseudo-inverse de la matrice M
$\mathcal{F}$	Fonction mathématique

Géométrie euclidienne et projective

$\sim$	Egalité à un facteur non-nul près
$\mathbf{q}$	Point 2D
$\tilde{\mathbf{q}}$	Coordonnées homogènes du point 2D
$\mathcal{Q}$	Point 3D dans le repère monde
$\tilde{\mathcal{Q}}$	Coordonnées homogènes du point 3D
$\mathbf{d}$	Mesure de profondeur
$\mathbf{\Pi}$	Plan de l'espace
$\mathcal{R}$	Matrice de rotation
$\mathbf{t}$	Vecteur de translation

Caméras et reconstruction 3D

$\mathcal{C}$	Caméra
$\mathcal{I}$	Image capturée par la caméra
P	Matrice de pose
K	Matrice de calibrage
F	Matrice fondamentale
E	Matrice essentielle
$\mathcal{C}_j$	$j^{\text{ème}}$ caméra reconstruite
$\mathcal{Q}^i$	$i^{\text{ème}}$ point 3D reconstruit
$\mathbf{q}_{i,j}$	Observation du $i^{\text{ème}}$ point 3D par la $j^{\text{ème}}$ caméra

Acronymes

CAO	Conception assistée par ordinateur
EKF	Extended Kalman Filter
GPS	Global Positionning System
ICP	Iterative Closest Point
MAD	Median Absolute Deviation
RANSAC	Random Sample Consensus
SfM	Structure from Motion
SLAM	Simultaneous Localization and Mapping

Sommaire

Introduction

Problématique

L'algorithme de localisation et de cartographie simultanée, plus connu en anglais sous le nom de SLAM pour *Simultaneous Localisation And Mapping*, est un algorithme très utilisé en robotique. L'idée principale de cet algorithme est de calculer simultanément la trajectoire de la caméra et la structure de la scène dans laquelle elle évolue.

Les applications du SLAM sont multiples, pouvant aller de l'exploration des zones dangereuses (zones de catastrophes naturelles, zones de combats militaires) à la cartographie d'environnements encore inaccessibles par l'homme (exploration des planètes). La Figure 1. présente des exemples d'environnements où le SLAM peut typiquement être exploité. On le retrouve aussi dans le domaine de la navigation autonome, notamment dans les premiers véhicules autonomes (Figure 2.) et dans les produits du quotidien comme les robots aspirateurs (Figure 3).

Un autre défi récent du SLAM est la localisation de drones. En effet les drones, révèlent leur potentiel depuis quelques années et les avancées technologiques permettent de les rendre encore plus intelligents, plus sécuritaires et presque indispensables déjà pour certains corps de métier. Les applications utilisant les drones sont variées (Développer l'agriculture de précision, livrer des colis, modéliser des sols et des bâtiments en 3D...).

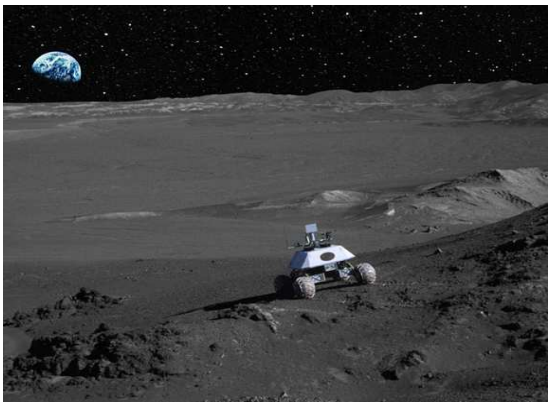
Le principal défi de la robotique mobile est de concevoir un système capable de se localiser de manière autonome, rapide et sûre, dans un environnement inconnu qui peut être dynamique, difficile et large, en se basant seulement sur les données de ses capteurs. Les algorithmes SLAM tentent depuis quelques années d'accomplir cette tache.

Les premières solutions proposées pour le SLAM remontent aux années 1980 avec les travaux de Smith and Cheeseman (1986). Depuis, cet algorithme a atteint un niveau de maturité significatif, avec un nombre important de solutions temps réel rapportées par l'état de l'art, particulièrement, les approches SLAM visuel. En effet, pour garantir le meilleur compromis rapidité/précision, la majorité des algorithmes SLAM visuel se sont basés sur une représentation éparse de l'environnement. Grâce à la mise en correspondance de primitives visuelles sur des images successives, il est possible d'estimer la position 3D de ces primitives et les poses (position, orientation) de la caméra. De nombreux progrès, tant sur la rapidité que sur la qualité

2. How Google's Self-Driving Car Works.
3. http ://www.dyson.fr/

(a) Exploration en milieu intérieur.

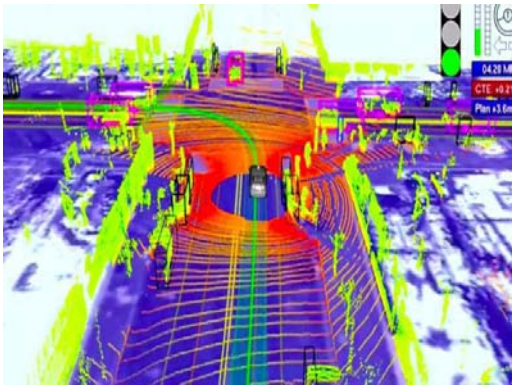
(b) Exploration en milieu sous-marin.



(c) Exploration dans l'espace.

(d) Exploration en milieu souterrain.

FIGURE 1 – Exemples d'environnements où l'algorithme SLAM est appliqué.



(a)

(b) Le SLAM dans le domaines des véhicules autonomes.

FIGURE 2 – **Le SLAM dans le domaines des véhicules autonomes.** (a) : le système de perception de la voiture autonome de Google [2], (b) : le véhicule VIPA développé par l'Institut Pascal

FIGURE 3 – **Le SLAM dans les produits du quotidien.** Le robot aspirateur de Dyson [3].

de la trajectoire et de la reconstruction 3D, ont été réalisés ces dernières années. Cependant la localisation par SLAM visuel, présente encore certaines limitations :

- Une dérive sur les longues trajectoires, due au cumul d'erreurs. Cette dérive est importante dans le cas du SLAM visuel monoculaire car la position de la caméra n'est estimée qu'à un facteur d'échelle près. Ce dernier est fixé initialement mais dérive au cours du temps.
- Une sensibilité aux rotations pures, liée à l'absence de convergence des points de vue qui est non appropriée à la triangulation.
- Un manque de robustesse dans les zones pauvres en textures.

Avec l'apparition des caméras Kinect de Microsoft [4], Primesense Carmine [5] ou Asus Xtion [6]. qui intègrent des capteurs 3D légers et de faible coût, les recherches sur le SLAM ont connu un grand essor. Ce type de capteur appelé caméra RGB-D, fournit non seulement l'image RGB de la scène, mais également une image de profondeur. L'exploitation de l'information de profondeur permet de palier certaines limitations du SLAM visuel monoculaire, notamment diminuer la dérive en facteur d'échelle. Notons, cependant que la majorité des méthodes RGB-D SLAM sont denses, elles utilisent tous les points de l'image et nécessitent des puissances de calculs GPU pour être temps réel.

L'objectif de cette thèse est de proposer une nouvelle approche RGBD-SLAM éparse. Celle-ci repose sur la modification d'un SLAM visuel monoculaire pour y intégrer l'information de profondeur fournie par un capteur RGB-D. Nous nous sommes focalisés sur une solution temps réel dans l'optique d'un fonctionnement sur CPU uniquement, pour des applications sur des structures légères (drones, lunettes ou casques de réalité augmentée...).

Toujours en vue de gagner en précision et robustesse, dans la deuxième partie de la thèse, nous proposons d'exploiter la connaissance a priori et partielle de la structure de la scène pour contraindre l'estimation de la trajectoire de la caméra.

4. https ://www.microsoft.com/fr-fr/
5. http ://www.primesense.com/
6. https ://www.asus.com/fr/3D-Sensor/Xtion-PRO/

Contributions

Cette thèse comporte deux parties :

- La première est consacrée à un nouveau RGBD-SLAM. Celui-ci repose sur la modification d'un SLAM monoculaire visuel pour y intégrer l'information de profondeur fournie par un capteur RGB-D afin d'améliorer la localisation et la reconstruction 3D issus du SLAM.
- La deuxième partie propose une modification de ce RGBD-SLAM pour y intégrer une contrainte additionnelle liée à la connaissance a priori et partielle de la structure de la scène. L'idée consiste à contraindre l'estimation de la trajectoire de la caméra avec des contraintes géométriques liées aux plans de la scène. Nous appellerons cette solution RGBD-SLAM Contraint. Cette contribution a pour but d'améliorer la précision et la robustesse de la localisation, notamment dans des environnements intérieurs avec peu de textures ou peu de structures géométriques.

Organisation du manuscrit

Dans le premier chapitre, nous rappelons quelques notions de bases nécessaires à la compréhension de ces travaux. Nous détaillons notamment la méthode de SLAM monoculaire qui sera revisitée dans le chapitre 3.
Le chapitre 2 est consacré à l'état de l'art des techniques de SLAM visuel monoculaires et des techniques de RGBD-SLAM.
Le chapitre 3 présente notre méthode RGBD-SLAM.
Le chapitre 4 porte sur l'évaluation de cette nouvelle méthode RGBD-SLAM, réalisée sur différentes séquences, avec une comparaison de notre approche par rapport aux récentes solutions de l'état de l'art.
Le chapitre 5 présente le RGBD-SLAM contraint. L'évaluation expérimentale de cette deuxième approche fait l'objet du dernier chapitre.

CHAPITRE 1

Notions de base

Dans ce chapitre, nous présentons quelques notations nécessaires à la compréhension de ce travail, ainsi que les concepts théoriques utilisés pour le SLAM visuel monoculaire (simultaneous localisation and mapping). Nous allons rappeler le modèle de projection d'une caméra suivant un modèle sténopé et les équations associées que nous utiliserons dans la suite de ce manuscrit. Nous décrirons par la suite l'algorithme de localisation et de cartographie simultanée (SLAM) de Mouragnon et al. (2006b), qui nous a servi de base dans notre travail.

1.1 Projection et rétro-projection

1.1.1 Le modèle sténopé

Il décrit la projection des points du repère 3D de la scène, dans le repère 2D de l'image créée par la caméra. Cette fonction de projection π' est définie par :

$$\pi' : \mathbb{R}^3 \rightarrow \mathbb{R}^2 \tag{1.1}$$

Dans cette thèse le modèle de caméra sténopé utilisé est le modèle décrit par Hartley and Zisserman (2003) et par Horaud and Monga (1995) et illustré dans la Figure 1.1. Ce modèle considère que l'ensemble des rayons lumineux passent par un seul et unique point avant d'atteindre le capteur. La projection décrite par ce modèle est connue sous le nom de la *projection perspective*. Le point d'intersection des rayons est appelé le *centre optique* de la caméra $\mathcal{C}$, la distance entre le centre optique et le plan image est appelée la *distance focale* f. La droite qui passe par le centre de projection et qui est perpendiculaire au plan image, est appelée *axe optique*. Le point d'intersection de l'axe optique avec le plan image est le *point principal* $\mathbf{p}$. Le point 3D $\mathcal{Q}$ dont les coordonnées sont $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})^\top$ est lié à son correspondant 2D $\mathbf{q}$ de coordonnées $(\mathbf{x}, \mathbf{y})^\top$ par les équations suivantes :

$$\pi'(\mathbf{X}, \mathbf{Y}, \mathbf{Z})^\top \rightarrow (\mathbf{x}, \mathbf{y})^\top \tag{1.2}$$

$$\mathbf{x} = \frac{\mathbf{X}.f}{\mathbf{Z}} \tag{1.3}$$

$$\mathbf{y} = \frac{\mathbf{Y}.f}{\mathbf{Z}} \tag{1.4}$$

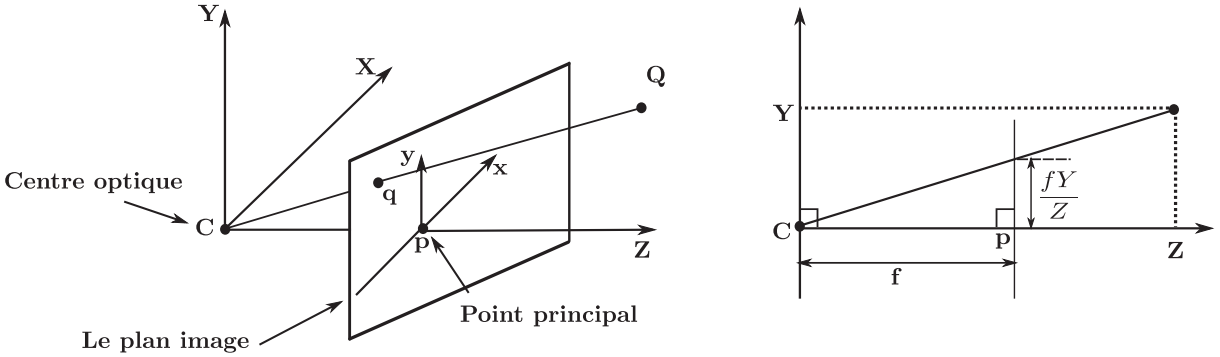


FIGURE 1.1 – Modèle d'une caméra sténopé Hartley and Zisserman (2003).

1.1.2 Modélisation Algébrique

Dans le cadre de nos travaux, nous utiliserons des caméras perspectives respectant le modèle de *caméra sténopé* idéal (voir Figure 1.2). Nous venons de décrire le modèle géométrique pour une caméra sténopé, nous détaillerons dans ce qui suit les équations algébriques associées, nous présenterons les différents paramètres physiques caractérisant la caméra.

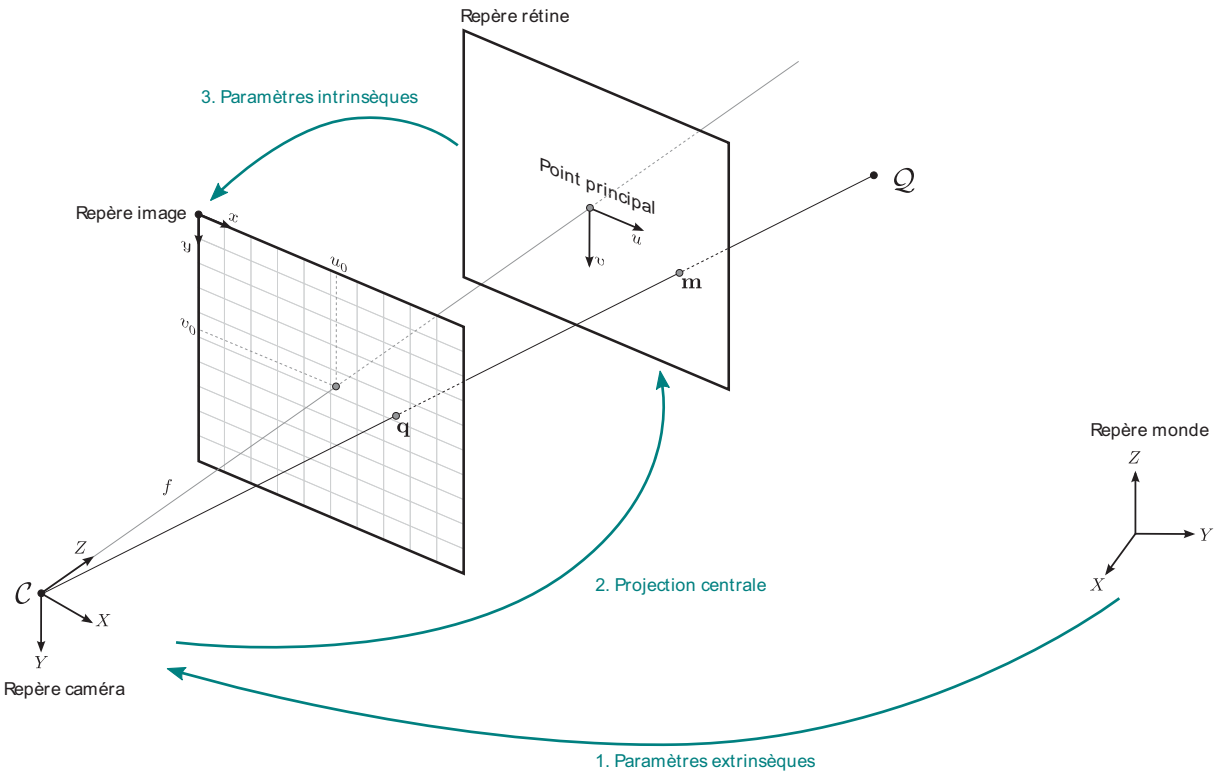


FIGURE 1.2 – **Projection perspective.** Illustration des différentes transformations géométriques appliquées au point 3D dans la projection perspective.

1.1.2.1 Projection d'un point dans l'image

Nous allons définir les équations de projection à l'aide de 4 repères (Figure 1.2). Le premier repère que nous décrivons est un repère 3D attaché à la caméra (le *repère caméra*). Comme origine nous choisissons le centre optique $\mathcal{C}$ et comme axe des $\mathbf{Z}$ l'axe optique. Les axes des $\mathbf{X}$ et des $\mathbf{Y}$ sont choisis comme étant parallèles au plan image et perpendiculaires entre eux. Nous choisissons les conventions suivantes : l'axe des $\mathbf{X}$ est orienté vers la droite, et l'axe des $\mathbf{Y}$ est orienté vers le bas.
Ensuite, nous définissons un repère 2D pour le plan image (le *repère rétine*). Son origine est le point principal. Ses axes $\mathbf{u}$ et $\mathbf{v}$ sont choisis parallèles aux axes des X et Y du repère caméra. Le repère rétine peut être vu comme la projection orthogonale du repère caméra.
La distance focale f est définie comme étant la distance orthogonale entre le centre optique et le plan image de la caméra (donc la distance entre le centre optique et le point principal).

Nous pouvons à présent définir les équations de projection d'un point 3D $\mathcal{Q}$, dont les coordonnées sont données dans le repère caméra (ce qui s'exprime par l'indice c) :

$$\mathcal{Q}_{\mathcal{C}} = \begin{pmatrix} \mathbf{X}_c \\ \mathbf{Y}_c \\ \mathbf{Z}_c \end{pmatrix} \tag{1.5}$$

Les coordonnées $\mathbf{u}$ et $\mathbf{v}$ du point image $\mathbf{m}$, projection du point $\mathcal{Q}_{\mathcal{C}}$, peuvent être calculées à l'aide des relations suivantes :

$$\mathbf{u} = f\frac{\mathbf{X}_c}{\mathbf{Z}_c} \tag{1.6}$$

$$\mathbf{v} = f\frac{\mathbf{Y}_c}{\mathbf{Z}_c} \tag{1.7}$$

En coordonnées homogènes, ces relations peuvent être exprimées par une multiplication matrice-vecteur :

$$\tilde{\mathbf{m}} = \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ 1 \end{pmatrix} \sim \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{X}_c \\ \mathbf{Y}_c \\ \mathbf{Z}_c \\ 1 \end{pmatrix} \tag{1.8}$$

Notons qu'en coordonnées homogènes, cette égalité vectorielle n'est définie qu'à un facteur près, d'où le symbole $\sim$.

1.1.2.2 Du repère rétine au repère pixel

Les images que nous utilisons sont des matrices de valeurs (*e.g.* niveaux de gris, couleurs Red, Green, Blue (RGB)) où les coordonnées sont exprimées en pixels, ceci implique que nous ne pouvons pas directement utiliser les coordonnées 2D définies dans la section précédente (*i.e.* dans le repère rétine), mais nous devons prendre en compte un autre changement de repère, qui vise à passer du repère rétine (repère lié à la physique du capteur et où les coordonnées sont exprimées en millimètres) au repère image (repère pixel) de la caméra (où les coordonnées sont exprimées en pixels).

Nous définissons donc un *repère image* comme illustré dans la Figure 1.2. L'origine est le coin en haut à gauche, l'orientation des axes de ce repère est identique au repère caméra. Les coordonnées pixelliques sont notées $\mathbf{x}$ et $\mathbf{y}$.
Soient $\mathbf{u}_0$ et $\mathbf{v}_0$ les coordonnées du point principal exprimées dans le repère image. Soit k_x la densité de pixels en direction de l'axe des $\mathbf{x}$ et k_y la densité de pixels en direction de l'axe des $\mathbf{y}$ (exprimée en nombre de pixels par millimètre par exemple).
Considérons le point $\mathbf{m}$ avec les coordonnées $\mathbf{u}$ et $\mathbf{v}$ dans le repère rétine. Ces coordonnées par rapport au repère image, sont obtenues en effectuant un changement d'unité et une translation comme suit :

$$\mathbf{x} = k_x\mathbf{u} + \mathbf{u}_0 \tag{1.9}$$

$$\mathbf{y} = k_y\mathbf{v} + \mathbf{v}_0 \tag{1.10}$$

En combinant les équations 1.8, 1.9 et 1.10, nous obtenons la modélisation de la projection complète du repère caméra au repère image :

$$\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\\ 1\end{pmatrix} \sim \begin{pmatrix}k_x f & 0 & u_0 & 0\\ 0 & k_y f & v_0 & 0\\ 0 & 0 & 1 & 0\end{pmatrix} \begin{pmatrix}\mathbf{X}_c\\ \mathbf{Y}_c\\ \mathbf{Z}_c\\ 1\end{pmatrix} \tag{1.11}$$

En posant :

$$\mathbf{a}_x = k_x f$$
$$\mathbf{a}_y = k_y f$$

On obtient K, la matrice des paramètres internes de la caméra, définie par :

$$\mathsf{K} = \begin{pmatrix}\mathbf{a}_x & 0 & \mathbf{u}_0\\ 0 & \mathbf{a}_y & \mathbf{v}_0\\ 0 & 0 & 1\end{pmatrix} \tag{1.12}$$

Avec :

- $\mathbf{a}_x, \mathbf{a}_y$ expriment la distance focale, en nombre de pixels (en direction horizontale et en direction verticale respectivement).
- $\mathbf{u}_0$, $\mathbf{v}_0$ sont les coordonnées du point principal, exprimées dans le repère pixel.

Il est important de noter que les capteurs à courte focale peuvent présenter un phénomène de distorsion important. Ceci se traduit visuellement par une déformation des lignes droites dans l'image sous forme de courbes. Pour corriger cela, il est possible d'ajouter au calibrage de la caméra des paramètres de distorsions permettant de passer de la position observée d'un point 2D dans l'image à sa position réelle, c'est-à-dire corrigée de toute distorsion. En pratique, la distorsion radiale est modélisée en utilisant un polynôme pouvant avoir jusqu'à 5 coefficients. Les autres distorsions étant beaucoup plus faibles sont souvent négligées. Pour plus de renseignements sur le calibrage des caméras, nous invitons le lecteur à se référer à l'article de Lavest et al. (1998).

Dans le cadre de ce mémoire, nous considérerons à la fois que la matrice de calibrage est connue et que les données utilisées dans nos algorithmes ont été préalablement corrigées en distorsion.

1.1.2.3 Introduction des déplacements

Jusqu'à présent, nous avons représenté les points 3D dans un repère attaché à la caméra. Afin de prendre en compte les déplacements de la caméra, nous devons choisir un repère absolu attaché à la scène 3D. Nous introduisons donc un *repère monde* (Figure 1.2), qui peut être choisi de manière arbitraire (dans notre cas le repère monde est confondu avec le repère de la première caméra). La position des points 3D et les poses des caméras sont décrits par rapport à ce repère. La pose de la caméra comprend la *position* du centre optique et l'*orientation* de la caméra.
La position du centre optique est notée :

$$\mathbf{t} = \begin{pmatrix} \mathbf{t}_x \\ \mathbf{t}_y \\ \mathbf{t}_z \end{pmatrix} \tag{1.13}$$

L'orientation de la caméra est exprimée par la rotation $\mathcal{R}$.
Soit $(\mathbf{X}_w, \mathbf{Y}_w, \mathbf{Z}_w)^\top$, les coordonnées du point 3D $\mathcal{Q}_w$ exprimées dans le repère monde et $(\mathbf{X}_c, \mathbf{Y}_c, \mathbf{Z}_c)^\top$ les coordonnées du point 3D $\mathcal{Q}_c$ exprimées dans le repère de la caméra $\mathcal{C}$. le changement de repère, du repère monde au repère caméra peut être exprimé comme suit :

$$\mathcal{Q}_\mathcal{C} = \begin{pmatrix} \mathcal{R}^\mathsf{T} & -\mathcal{R}^\mathsf{T}\mathbf{t} \end{pmatrix} \tilde{\mathcal{Q}}_w \tag{1.14}$$

$$\mathcal{Q}_\mathcal{W} = \begin{pmatrix} \mathcal{R} & \mathbf{t} \end{pmatrix} \tilde{\mathcal{Q}}_c \tag{1.15}$$

Pour simplifier les notations, dans la suite du manuscrit le point 3D exprimé dans le repère monde sera noté $\mathcal{Q}$.

Rappelons que les matrices de rotation sont des matrices *orthonormales*. Chaque rotation peut être décomposée en trois rotations de base, autour des axes du repère :

$$\mathcal{R} = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}$$

Les angles α, β, γ sont associés aux axes $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ respectivement. Il s'agit des angles *roulis-tangage-lacet*.

1.1.2.4 Modèle Complet

Pour obtenir le modèle de projection d'un point 3D, nous combinons les équations précédentes 1.11 et 1.14.
Le modèle complet s'écrit alors :

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \\ 1 \end{pmatrix} \sim \mathsf{K} \begin{pmatrix} \mathcal{R}^\mathsf{T} & -\mathcal{R}^\mathsf{T}\mathbf{t} \end{pmatrix} \begin{pmatrix} \mathbf{X}_w \\ \mathbf{Y}_w \\ \mathbf{Z}_w \\ 1 \end{pmatrix} \tag{1.16}$$

Dans la suite du manuscrit, on notera P_c la matrice de pose de la caméra $\mathcal{C}$ [1] :

$$\mathsf{P}_c = \begin{pmatrix} \mathcal{R}^\mathsf{T} & -\mathcal{R}^\mathsf{T}\mathbf{t} \end{pmatrix}$$

1. En toute rigueur la matrice P_c est la pose inverse de la caméra $\mathcal{C}$, par mesure de simplicité on l'appellera dans la suite du manuscrit la pose de la caméra

Ainsi, on écrira la projection d'un point 3D $\mathcal{Q}$, en un point 2D $\mathbf{q}$ comme suit :

$$\mathbf{q} = \pi\left(\mathsf{K}\mathsf{P}_c\tilde{\mathcal{Q}}\right) \tag{1.17}$$

Avec π une fonction définie comme suit :

$$\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2 \tag{1.18}$$

$$\pi(\mathbf{X}, \mathbf{Y}, \mathbf{Z})^\top \rightarrow (\mathbf{x}, \mathbf{y})^\top \tag{1.19}$$

$$\mathbf{x} = \frac{\mathbf{X}}{\mathbf{Z}} \tag{1.20}$$

$$\mathbf{y} = \frac{\mathbf{Y}}{\mathbf{Z}} \tag{1.21}$$

1.1.3 La rétro-projection

La rétro-projection peut être vue comme l'opération inverse de la projection. Son but est de générer la position d'un point 3D $\mathcal{Q}$ à partir de son observation 2D $\mathbf{q}$ dans l'image et de sa mesure de profondeur. Si la profondeur $\mathbf{d}$ et les paramètres intrinsèques de la caméra sont connus, le point 3D $\mathcal{Q}_\mathcal{C}$ correspondant au point 2D $\mathbf{q}$ peut être obtenu en utilisant les équations suivantes :

$$\mathbf{X}_c = \frac{\mathbf{x} - \mathbf{u}_0}{\mathbf{a}_x}\mathbf{d} \tag{1.22}$$

$$\mathbf{Y}_c = \frac{\mathbf{y} - \mathbf{v}_0}{\mathbf{a}_y}\mathbf{d} \tag{1.23}$$

$$\mathbf{Z}_c = \mathbf{d} \tag{1.24}$$

Comme la matrice des paramètres internes K est une matrice triangulaire supérieure, son inverse est définie comme suit :

$$\mathsf{K}^{-1} = \begin{pmatrix} \frac{1}{\mathbf{a}_x} & 0 & \frac{-\mathbf{u}_0}{\mathbf{a}_x} \\ 0 & \frac{1}{\mathbf{a}_y} & \frac{-\mathbf{v}_0}{\mathbf{a}_y} \\ 0 & 0 & 1 \end{pmatrix} \tag{1.25}$$

Les Équations 1.22, 1.23, 1.24 peuvent s'écrire comme suit :

$$\begin{pmatrix} \mathbf{X}_c \\ \mathbf{Y}_c \\ \mathbf{Z}_c \end{pmatrix} = \mathbf{d} \begin{pmatrix} \frac{1}{\mathbf{a}_x} & 0 & \frac{-\mathbf{u}_0}{\mathbf{a}_x} \\ 0 & \frac{1}{\mathbf{a}_y} & \frac{-\mathbf{v}_0}{\mathbf{a}_y} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{1} \end{pmatrix} \tag{1.26}$$

On notera par π^{-1} la fonction qui construit un point 3D $\mathcal{Q}_\mathcal{C}$ $(\mathbf{X}_c, \mathbf{Y}_c, \mathbf{Z}_c)$ à partir du point 2D $\mathbf{q}(\mathbf{x}, \mathbf{y})$ et sa mesure de profondeur $\mathbf{d}$ tel que :

$$\mathcal{Q}_c = \pi^{-1}(\mathbf{q}, \mathbf{d}) = \mathbf{d}\mathsf{K}^{-1}\tilde{\mathbf{q}} \tag{1.27}$$

Le point 3D dans le repère monde $\mathcal{Q}\ (\mathbf{X}, \mathbf{Y}, \mathbf{Z})^\top$ est alors défini comme suit :

$$\mathcal{Q} = \mathsf{P}_c^{-1} \left(\tilde{\mathcal{Q}}_\mathcal{C}\right)^\top \tag{1.28}$$

En remplaçant $\mathcal{Q}_\mathcal{C}$ par son expression (équation 1.27), on obtient :

$$\mathcal{Q} = \mathsf{P}_c^{-1} \left(\left(\pi^{-1}(\mathbf{q}, \mathbf{d})\right)^\top |1\right)^\top \tag{1.29}$$

Par mesure de simplicité, par la suite nous écrirons cette équation comme suit :

$$\mathcal{Q} = \mathsf{P}_c^{-1}\pi^{-1}(\mathbf{q}, \mathbf{d}) \tag{1.30}$$

1.2 Calcul de la géométrie de l'environnement

Dans cette section, nous présentons quelques outils mathématiques élémentaires qui permettent de calculer les poses des différentes caméras présentes dans la scène, et les points 3D observés. Ces paramètres sont calculés en utilisant les points d'intérêt observés dans les images RGB.

1.2.1 Poses de caméras et déplacement relatif

Le but de cette section est de formaliser la notion de *déplacement relatif* entre deux caméras ainsi que les notations associées. Comme nous l'avons vu précédemment, la pose des caméras peut être vue comme un changement de repère entre le repère monde et les repères attachés aux caméras :

$$\mathcal{Q}_{\mathcal{C}_1} = \left(\mathcal{R}_1{}^\mathsf{T} \quad -\mathcal{R}_1{}^\mathsf{T}\mathbf{t_1}\right) \tilde{\mathcal{Q}} \tag{1.31}$$

$$\mathcal{Q}_{\mathcal{C}_2} = \left(\mathcal{R}_2{}^\mathsf{T} \quad -\mathcal{R}_2{}^\mathsf{T}\mathbf{t_2}\right) \tilde{\mathcal{Q}} \tag{1.32}$$

Avec : $\mathcal{Q}$ les coordonnées du point 3D exprimées dans le repère monde et $\mathcal{Q}_{\mathcal{C}_1}$ et $\mathcal{Q}_{\mathcal{C}_2}$ les coordonnées du même point 3D exprimées dans les repères liés aux caméras $\mathcal{C}_1$ et $\mathcal{C}_2$ respectivement. Afin de fixer les notations, nous appellerons $(\mathcal{R}_{1\to2}, \mathbf{t}_{1\to2})$ le déplacement relatif entre les caméras, c'est-à-dire la transformation permettant de passer du repère lié à $\mathcal{C}_1$ à celui lié à $\mathcal{C}_2$:

$$\mathcal{Q}_{\mathcal{C}_2} = \left(\mathcal{R}_{1\to2} \quad \mathbf{t}_{1\to2}\right) \tilde{\mathcal{Q}}_{\mathcal{C}_1}. \tag{1.33}$$

Des équations 1.31 et 1.32, on peut obtenir le système d'équations suivant :

$$\begin{cases} \mathcal{Q}_\mathcal{W} = \left(\mathcal{R}_1 \quad \mathbf{t_1}\right) \tilde{\mathcal{Q}}_{\mathcal{C}_1} \\ \mathcal{Q}_{\mathcal{C}_2} = \left(\mathcal{R}_2{}^\mathsf{T} \quad -\mathcal{R}_2{}^\mathsf{T}\mathbf{t_2}\right) \tilde{\mathcal{Q}}_\mathcal{W} \end{cases} \tag{1.34}$$

d'où

$$\mathcal{Q}_{\mathcal{C}_2} = \left(\mathcal{R}_2{}^\mathsf{T}\mathcal{R}_1 \quad \mathcal{R}_2{}^\mathsf{T}(\mathbf{t}_1 - \mathbf{t}_2)\right) \tilde{\mathcal{Q}}_{\mathcal{C}_1}. \tag{1.35}$$

Le déplacement relatif entre les caméras est donc défini comme suit :

$$\begin{cases} \mathcal{R}_{1\to2} = \mathcal{R}_2{}^\mathsf{T}\mathcal{R}_1 \\ \mathbf{t}_{1\to2} = \mathcal{R}_2{}^\mathsf{T}(\mathbf{t}_1 - \mathbf{t}_2) \end{cases} \tag{1.36}$$

1.2.2 Reconstruction 3D des points

Nous avons vu que la rétro-projection d'une observation 2D dans l'image permet d'obtenir sa position 3D grâce à sa mesure de profondeur (se référer à la section 1.1.3). Mais dans le cas où la mesure de profondeur est indisponible la position du point 3D peut être estimée à partir des observations 2D de ce point, acquises d'au moins deux points de vue différents. On parle alors de **triangulation** du point. L'idée de la triangulation est de calculer l'intersection des rayons optiques issus de deux observations. En pratique, en raison des bruits sur les différentes données (calibrage, poses des caméras, positions des observations, *etc.*), les rayons ne s'intersectent pas (Figure 1.3). Dans un but de robustesse et de précision, la notion de triangulation est généralisée à plus de deux caméras. Par exemple, dans le cas de trois caméras, il est possible de calculer trois triangulations différentes à partir des couples de caméras (1,2), (2,3) et (1,3). Le résultat final de la triangulation est alors le barycentre de ces trois points. Il existe également une approche linéaire permettant de trianguler un point observé par N-vues en utilisant la méthode DLT (Hartley and Zisserman (2003)).

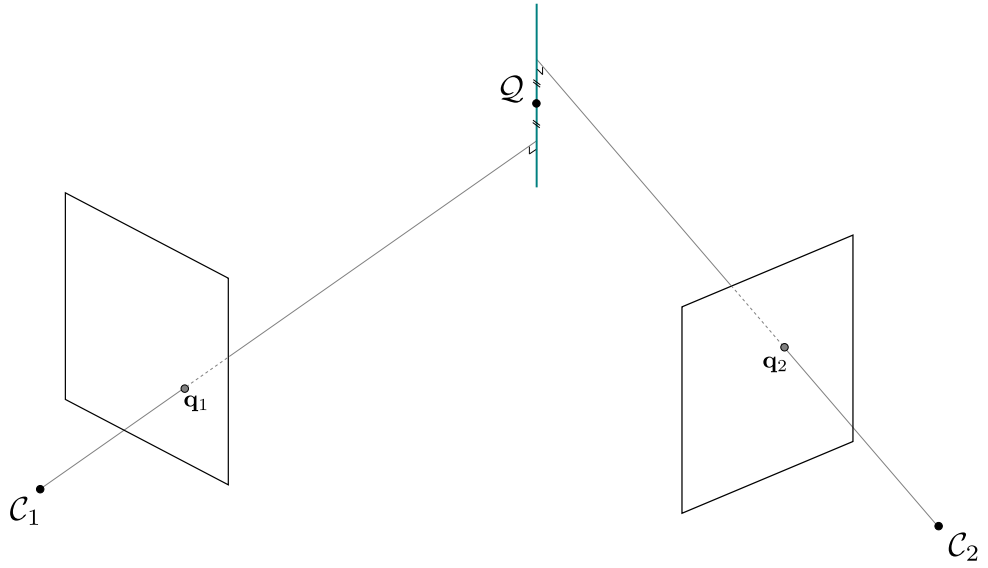


FIGURE 1.3 – **Triangulation de points 3D.** La structure de l'environnement peut être obtenue par triangulation des observations dans les images.

1.2.3 Calcul de pose par association 2D/3D

Dans l'algorithme du SLAM visuel que nous utilisons, la pose de la caméra courante est calculée à partir d'un ensemble d'associations 2D/3D, cette technique est connue sous le nom de *recalage* 2D/3D. L'utilisation d'associations 2D/3D plutôt que 2D/2D présente quelques avantages. Le calcul de pose 2D/3D est beaucoup plus rapide que le calcul de pose 2D/2D (section A.1) (l'estimation de la matrice essentielle étant une étape coûteuse). De plus l'approche 2D/3D permet de garder un même facteur d'échelle tout au long d'une reconstruction incrémentale (de proche en proche).

Une des méthodes utilisées pour le calcul de pose est la méthode "*The Three Point Perspective Pose Estimation*" présentée dans les travaux de Haralick et al. (1994). Le but de cette

approche est de déterminer la pose relative entre la caméra et la scène à partir de trois correspondances 2D/3D connues. On parle de problème minimal PnP (Perspective n-points), avec n=3. Cette méthode se décompose en deux étapes. La première étape consiste à estimer les profondeurs inconnues $\mathrm{d}_\mathrm{i}^{\mathcal{C}}$ de chaque point dans le repère de la caméra $\mathcal{C}$, en utilisant les contraintes sur les sinus, déterminés par le triangle $\mathcal{C}\mathcal{Q}_i\mathcal{Q}_j$. La distance entre les points $\mathcal{Q}_i$ et $\mathcal{Q}_j$ et l'angle entre les directions $\mathcal{C}\mathcal{Q}_i$ et $\mathcal{C}\mathcal{Q}_j$ sont supposés connus. L'estimation des profondeurs des points se fait alors en résolvant un polynôme d'ordre 4. Une fois que les coordonnées des points sont connues dans le repère de la caméra $\mathcal{C}$, la deuxième étape est d'estimer la transformation rigide P_c qui relie le repère caméra au repère monde.

Bien que les solutions P3P soient connues pour le calcul de pose, d'autres solutions qui utilisent un nombre de correspondances >3 (*PnP)* sont habituellement les plus utilisées. En effet la précision de la pose estimée augmente en augmentant le nombre de points utilisées. Par ailleurs, les appariements 2D/3D doivent être établis de façon automatique, ce qui implique des techniques robustes à la présence de faux appariements (ou outliers). C'est pour cela que ces méthodes sont combinés à un processus RANSAC (Random Sample Consensus) Raguram et al. (2008).

1.2.4 Ajustement de faisceaux

Les poses des caméras et la structure de l'environnement calculées avec les méthodes présentées précédemment ne fournissent pas une solution optimale au problème de reconstruction et localisation simultanées. Une étape d'optimisation par ajustement de faisceaux est nécessaire pour raffiner l'ensemble de ces paramètres (à savoir les six paramètres de chaque pose de la caméra et les trois paramètres de la position de chaque point 3D). Cette optimisation repose sur la minimisation des erreurs de projection (Figure 1.4).

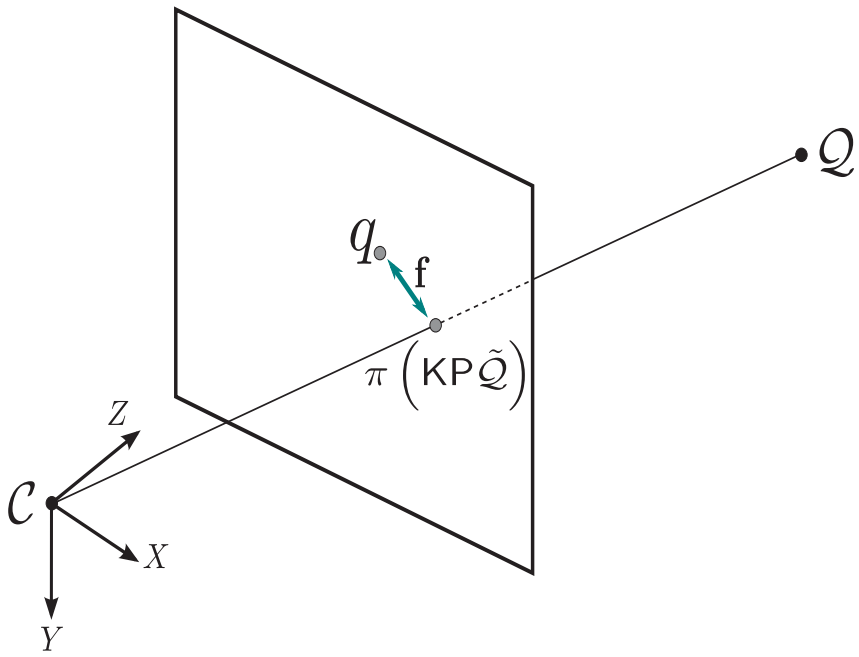


FIGURE 1.4 – **Erreur de projection.** L'erreur de projection est la distance entre l'observation $\mathbf{q}$ d'un point $\mathcal{Q}$ et sa projection dans l'image $\pi(\mathsf{K}\mathsf{P}\tilde{\mathcal{Q}})$.

Cette erreur est la distance 2D entre le point d'intérêt détecté dans l'image et la projection de son point 3D associé.

$$\mathbf{f} = ||\mathbf{q} - \pi(\mathsf{K}\mathsf{P}\tilde{\mathcal{Q}})||. \tag{1.37}$$

La fonction à minimiser dans l'ajustement de faisceaux s'écrit donc :

$$f(\{P_j\}_{j=1}^{\mathcal{N}_c}, \{\mathcal{Q}_i\}_{i=1}^{\mathcal{N}_p}) = \sum_{1\leq i\leq \mathcal{N}_p} \sum_{j\in\mathcal{A}_i} ||\mathbf{q}_{i,j} - \pi(\mathsf{K}\mathsf{P}_j\tilde{\mathcal{Q}}_i)||^2, \tag{1.38}$$

où $\{\mathsf{P}_j\}_{j=1}^{\mathcal{N}_c}$ représente l'ensemble des poses de la caméra et $\{\mathcal{Q}_i\}_{i=1}^{N_p}$ l'ensemble des points 3D à optimiser. $\mathbf{q}_{i,j}$ est l'observation 2D du $i^{ème}$ point 3D dans la $j^{ème}$ caméra. L'ensemble $\mathcal{A}_i$ est l'ensemble des indices des caméras observant le point $\mathcal{Q}_i$. Afin de minimiser cette fonction de coût, on utilise l'algorithme du Levenberg Marquard décrit dans Moré (1978).

L'ajustement de faisceaux représente une étape fondamentale en reconstruction 3D. La majorité des contributions apportées par nos travaux est axée sur ce processus d'optimisation. C'est pour cette raison que nous détaillons son implémentation ci-après :

Construction des matrices Jacobienne et Hessienne Le principal objectif d'une itération d'ajustement de faisceaux consiste à estimer les incréments $\delta = (\delta_{camera}^\top, \delta_{point}^\top)^\top$ à appliquer aux paramètres afin de minimiser la fonction de coût décrite dans l'équation 1.38. Ces incréments sont obtenus par la résolution du système linéaire défini par l'équation normale :

$$\mathsf{J}^\top\mathsf{J} \times \delta = \mathsf{J}^\top \times \mathbf{r}, \tag{1.39}$$

où $\mathbf{r}$ est le vecteur concaténant toutes les erreurs de re-projections des points 3D participant à l'ajustement de faisceaux, et J représente la matrice Jacobienne créée à partir des dérivées partielles de l'erreur de re-projection par rapport aux paramètres de la scène et des poses à optimiser comme suit :

$$\mathsf{J} = \left(\frac{\partial f}{\partial \mathcal{R}_1}, \frac{\partial f}{\partial \mathbf{t}_1} \ldots, \frac{\partial f}{\partial \mathcal{R}_{N_c}}, \frac{\partial f}{\partial \mathbf{t}_{N_c}}, \frac{\partial f}{\partial \mathcal{Q}^1}, \ldots, \frac{\partial f}{\partial \mathcal{Q}^{N_p}}\right). \tag{1.40}$$

Où N_c représente le nombre de caméras, et N_p le nombre des points 3D à optimiser.

Étant donné que tous les points ne sont pas observés par toutes les caméras, la matrice Hessienne, qui s'écrit selon l'approximation de de Gauss-Newton $\mathsf{J}^\top\mathsf{J}$, possède une structure à la fois creuse et par blocs (voir Figure 1.5). Cette matrice peut alors être représentée de la manière suivante :

$$\mathsf{J}^\top\mathsf{J} = \begin{pmatrix} \mathsf{U} & \mathsf{W} \\ \mathsf{W}^\top & \mathsf{V} \end{pmatrix}, \tag{1.41}$$

où :

- $\mathsf{U} = \mathsf{J}_{camera}^\top\mathsf{J}_{camera}$, matrice carrée (de taille $6N_c \times 6N_c$) contenant des blocs diagonaux de taille 6×6 ;
- $\mathsf{V} = \mathsf{J}_{point}^\top\mathsf{J}_{point}$, matrice carrée (de taille $3N_p \times 3N_p$) diagonale par blocs de taille 3×3.
- $\mathsf{W} = \mathsf{J}_{camera}^\top\mathsf{J}_{point}$, matrice (de taille $6N_c \times 3N_p$) exprimant les intercorrélations entre les paramètres des points 3D et les paramètres extrinsèques des caméras. La structure de W est directement liée au fait que les points sont vus ou non dans les images. W contient donc un nombre, non nul égal au nombre de reprojections 2D, de blocs de dimension (6×3) .

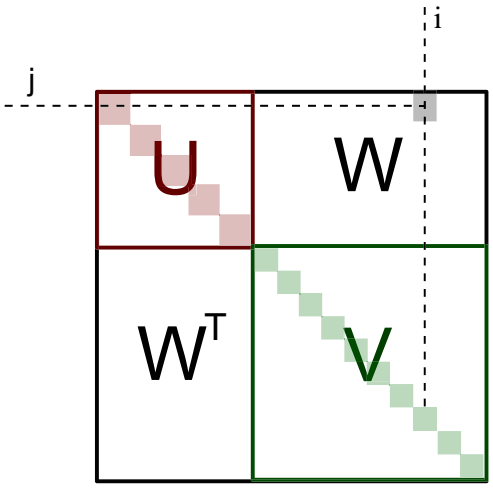


FIGURE 1.5 – Structure de la matrice Hessienne $\mathsf{J}^\top\mathsf{J}$ de l'ajustement de faisceaux.

Résolution éparse des équations normales par le complément de Schur L'approximation de Gauss-Newton de la matrice Hessienne aura la structure représentée dans la Figure 1.5. Cette structure particulière rend possible la résolution de l'équation 1.39 de manière très rapide par le complément de Schur.

$$\begin{pmatrix} \mathsf{U} & \mathsf{W} \\ \mathsf{W}^\top & \mathsf{V} \end{pmatrix} \begin{pmatrix} \delta_{camera} \\ \delta_{point} \end{pmatrix} = \begin{pmatrix} \mathbf{g}_{camera} \\ \mathbf{g}_{point} \end{pmatrix}, \tag{1.42}$$

avec $\mathbf{g}_{camera} = \mathsf{J}_{camera}\mathbf{r}$ et $\mathbf{g}_{point} = \mathsf{J}_{point}\mathbf{r}$.

Inversion d'une matrice par bloc en utilisant le complément de Schur : Soit une matrice M symétrique et inversible, partitionnée comme suit :

$$\mathsf{M} = \begin{pmatrix} \mathsf{U} & \mathsf{W} \\ \mathsf{W}^\top & \mathsf{V} \end{pmatrix}$$

Le complément de Schur du bloc V de la matrice M est $\mathsf{Sch}(\mathsf{V}) = \mathsf{U} - \mathsf{W}\mathsf{V}^{-1}\mathsf{W}^\top$.
Si la matrice M est symétrique et inversible, alors son inverse est défini comme suit :

$$\mathsf{M}^{-1} = \begin{pmatrix} \mathsf{U} & \mathsf{W} \\ \mathsf{W}^\top & \mathsf{V} \end{pmatrix}^{-1} = \begin{pmatrix} \mathsf{Sch}(\mathsf{V})^{-1} & -\mathsf{Sch}(\mathsf{V})^{-1}\mathsf{W}\mathsf{V}^{-1} \\ -\mathsf{V}^{-1}\mathsf{W}^\top\mathsf{Sch}(\mathsf{V})^{-1} & \mathsf{V}^{-1} + \mathsf{V}^{-1}\mathsf{W}^\top\mathsf{Sch}(\mathsf{V})^{-1}\mathsf{W}\mathsf{V}^{-1} \end{pmatrix} \tag{1.43}$$

Cette formule d'inversion est très utile en pratique si V est de grande dimension et s'inverse facilement. Dans notre cas, V sera de grande taille et diagonale par blocs 3×3.

Plus de détails sur la résolution de ce système sont disponibles dans l'annexe B et dans l'article de Triggs et al. (1999).

1.2.5 Localisation et cartographie simultanées (SLAM)

Le RGBD-SLAM que nous présentons dans ce mémoire s'appuie sur la méthode de SLAM monoculaire proposée par Mouragnon et al. (2006b) (schématisée sur la Figure 1.6). Les algorithmes de SLAM monoculaire ont pour but de localiser une caméra et de reconstruire en ligne une carte de l'environnement à partir des observations réalisées par cette caméra au cours du

temps. Ainsi, à l'instant $t + 1$, la caméra observe des primitives déjà présentes dans la carte de l'environnement. Ces observations permettent de localiser la caméra. La carte pourra alors être enrichie : la position des primitives existantes pourra être raffinée et de nouvelles primitives seront ajoutées.

Pour une bonne compréhension de la suite de nos travaux, nous détaillerons les différentes étapes de l'algorithme proposé par Mouragnon et al. (2006b). Nous pouvons classifier ces différentes étapes dans deux parties, une partie dont les traitements sont réalisés dans l'image 2D, et une partie qui exploite les données 3D.

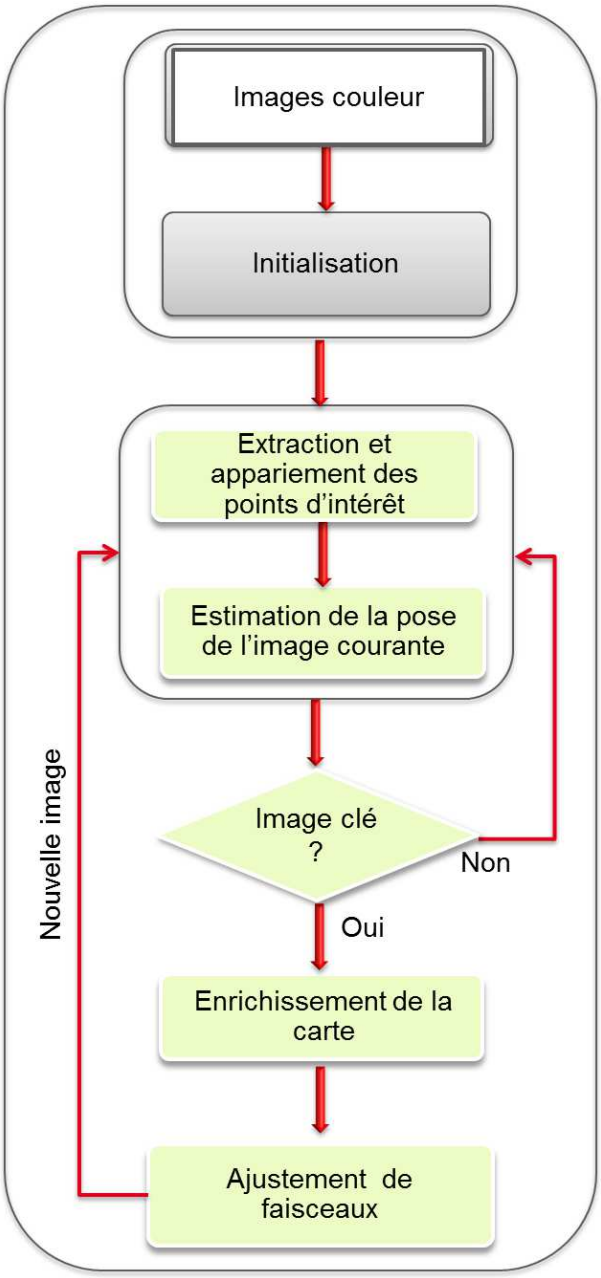


FIGURE 1.6 – **Schéma du fonctionnement du SLAM de Mouragnon et al. (2006b).**

1.2.5.1 Traitements 2D

Le but des traitements 2D (c'est-à-dire au niveau des images) de la méthode de SLAM visuel est triple : détecter les points d'intérêt, les associer dans les images successives et enfin fournir un critère indiquant si l'image courante est une image clé ou non (cette notion d'image clé étant définie plus loin).

Points d'intérêt. Les *points d'intérêt* utilisés dans la méthode de Mouragnon sont des points de Harris (Harris and Stephens (1988)) : il a été montré par Schmid et al. (2000) que ces points d'intérêt offrent une bonne répétabilité, ce qui maximise les chances de pouvoir détecter les mêmes points dans les images successives. Il est à noter que la détection des points d'intérêt est faite par baquets, c'est-à-dire que l'image est découpée en sous-zones et que les points d'intérêt sont recherchés dans chacune de ces zones. Ceci permet de mieux répartir les points d'intérêt dans l'image, ce qui est une configuration nécessaire pour maximiser la qualité des résultats des différents processus de reconstruction 3D. Plus de détails sur les détecteurs de points d'intérêt et leur performance respective peuvent être trouvés dans l'article de Mikolajczyk and Schmid (2002).

Descripteurs associés. Le *descripteur* d'un point d'intérêt est la signature permettant de mesurer sa similarité avec tout autre point d'intérêt. L'idée du descripteur est de calculer la signature du voisinage du point d'intérêt considéré. Pour mettre en correspondance les points d'intérêt, on mesure alors la distance entre leurs descripteurs respectifs (cette mesure étant dépendante du type de descripteur utilisé). La taille du voisinage pris en compte pour le calcul du descripteur peut être choisie automatiquement à partir de l'échelle du point d'intérêt (par exemple pour SIFT, Lowe (2004)).

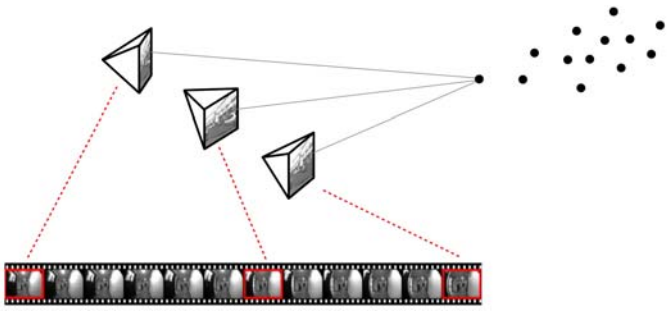
Le descripteur utilisé dans notre étude est un descripteur offrant des performances similaires à SURF (Speeded Up Robust Features, Bay et al. (2006)) qui repose sur la distribution des ondelettes de Haar 2D sur le voisinage des points d'intérêt. D'autres descripteurs ont été présentés et comparés dans l'article de Mikolajczyk and Schmid (2005).

Notions d'image clé. Dans la méthode de SLAM utilisée, toutes les images de la vidéo n'ont pas le même rôle. Toutes les images sont utilisées pour localiser la caméra dans l'environnement précédemment reconstruit mais certaines images, appelées *images clés*, ont un rôle particulier. Nous verrons dans la section suivante que ces images sont utilisées par la brique de reconstruction 3D.
Une image est identifiée comme étant une image clé [2] si le nombre d'appariements 2D entre l'image courante et la dernière image clé est inférieur à un seuil, ou si le nombre de correspondances 2D/3D non aberrantes est en dessous d'un seuil.

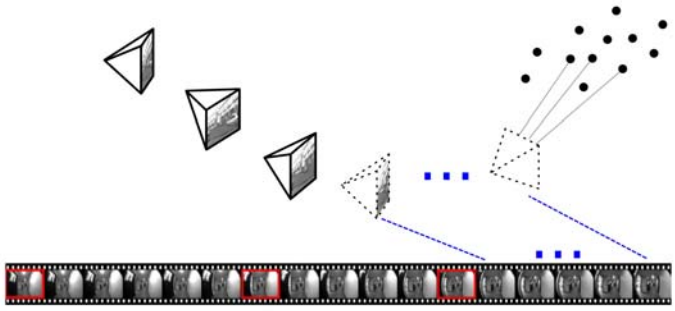
1.2.5.2 Traitements 3D

Initialisation. Le SLAM visuel monoculaire nécessite une étape d'initialisation. En effet au début de l'algorithme aucune information sur la géométrie de la scène n'est disponible. Les seules informations existantes sont les données visuelles 2D, et les matrices de calibrage des caméras. Ces données seront donc exploitées pour estimer les poses des premières caméras clés

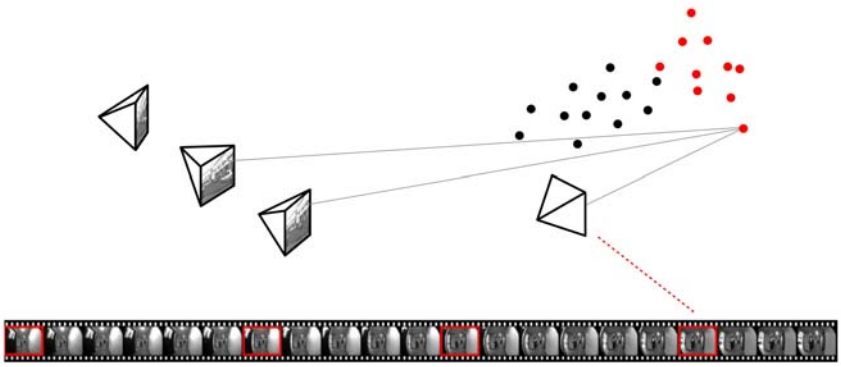
2. C'est l'image précédente qui est sélectionnée car c'est la dernière image ayant eu un nombre de correspondances supérieur au seuil



(a) **Initialisation.** L'initialisation de la structure à partir de 3 images clés est réalisée à l'aide de l'algorithme des 5 points (Nistér (2004)).



(b) **Localisation de la caméra.** Chaque nouvelle image est localisée à partir des correspondances 2D/3D.



(c) **Création d'une nouvelle image clé.** Si l'image courante est clé, elle est alors utilisée pour trianguler de nouveaux points 3D. Les paramètres des $\mathcal{N}_i$ dernières caméras clés et des $\mathcal{N}_p$ points 3D sont alors raffinés à l'aide d'un ajustement de faisceaux.

FIGURE 1.7 – **Résumé de la méthode SLAM (Mouragnon et al. (2006b)).** Les caméras en pointillés sont des caméras classiques et les caméras en trait plein sont des caméras clés. Les images encadrées en rouge sont les images détectées comme étant des images clés.

et les positions des points 3D. Sur les trois premières caméras clés ($\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$), la caméra $\mathcal{C}_1$ est fixée à l'origine (position $(0,0,0)^\top$, orientation $\mathcal{I}_3$). Le déplacement relatif entre les caméras ($\mathcal{C}_1, \mathcal{C}_3$) est calculé à partir de l'ensemble des correspondances 2D/2D disponibles, en passant par l'estimation de la matrice essentielle (voir la section A.1), par la méthode de Nistér (2004). La reconstruction des points 3D vus par les caméras $\mathcal{C}_1, \mathcal{C}_2$ se fait par triangulation (voir la section 1.2.2). Ces points sont mis en correspondance avec les primitives 2D détectées dans l'image de la caméra $\mathcal{C}_3$, donnant lieu à un ensemble d'associations 2D/3D qui vont permettre d'estimer la pose de la caméra $\mathcal{C}_3$ (voir la section 1.2.3). Les poses des caméras et les paramètres des points 3D estimés, sont ensuite optimisés dans un ajustement de faisceaux global (voir la section 1.2.4).

Processus incrémental. Une fois l'initialisation précédente réalisée, les appariements 2D fournis par le module de suivi permettent de remonter à des associations 2D/3D entre les points d'intérêt de l'image courante et les points 3D préalablement reconstruits. La pose de la caméra courante (Figure 1.7(b).) peut alors être estimée à partir de ces appariements (section 1.2.3).

Si la caméra courante est détectée comme étant une caméra clé, elle est alors utilisée pour enrichir la reconstruction de l'environnement (Figure 1.7(c).) :

- Sa pose et ses observations sont utilisées pour trianguler de nouveaux points 3D (section 1.2.2).
- Un ajustement de faisceaux est ensuite appliqué pour raffiner la géométrie de la reconstruction et les poses des caméras.

La particularité des travaux de Mouragnon et al. (2006b) est que l'ajustement de faisceaux ne raffine pas l'ensemble de la reconstruction. En effet, afin d'assurer un traitement temps-réel, l'ajustement de faisceaux est uniquement réalisé sur une sous-partie de la reconstruction (au moyen de l'équation 1.38). Cette sous-partie est constituée des $\mathcal{N}_i$ dernières caméras clés et des points 3D associés. Les $\mathcal{N}_c - \mathcal{N}_i$ autres caméras clés sont fixées, ce qui permet d'apporter les contraintes assurant la cohérence géométrique de la reconstruction de proche en proche. On parle dans ce cas d'ajustement de faisceaux local ou glissant.

Nous verrons par la suite que le SLAM visuel monoculaire présente plusieurs inconvénients ; c'est un processus incrémental donc sujet à l'accumulation d'erreurs. De plus, avec une seule caméra et sans connaissance a priori sur la géométrie de la scène, le SLAM permet d'estimer la rotation, mais la translation n'est connue qu'à un facteur d'échelle près. Ce facteur est fixé à l'initialisation mais il est sujet à la dérive.

1.3 Les capteurs RGB-D

Dans cette section nous présentons quelques caractéristiques techniques des capteurs utilisés. Les capteurs qui nous intéressent dans nos travaux sont les capteurs qui fournissent simultanément une image visuelle (RGB) et une image de profondeur (D). Ces capteurs sont appelés capteurs RGB-D.
Ils sont généralement *actifs*. Ils possèdent leur propre source émettrice par opposition aux capteurs passifs nécessitant une source de rayonnement extérieure.

On distingue au sein de ces capteurs actifs deux modes de fonctionnement distincts : la triangulation active, notion sous laquelle se cache le principe de la lumière structurée, et la mesure du temps de vol ou time-of-flight (ToF).

Portée minimale et maximale pour les mesures de profondeurs	0.5−5m
Champs de vision	57 degrés
Taille des images	640 × 480 pixels
Fréquence des images	30 images par seconde

TABLE 1.1 – Les caractéristiques du capteur Microsoft Kinect V1.

1.3.0.1 Le capteur Kinect V1

Dans cette section, nous présentons brièvement la technologie de la lumière structurée ou codée utilisée dans le capteur *Kinect* V1 de *Microsoft* [3]. Les capteurs exploitant ce type de technologie utilisent une source de lumière qui émet, d'une manière continue une lumière infrarouge en direction de l'objet observé. Cette lumière est filtrée et échantillonnée en un motif connu comme le montre la Figure.1.8. Le motif projeté est ensuite détecté par une caméra infrarouge

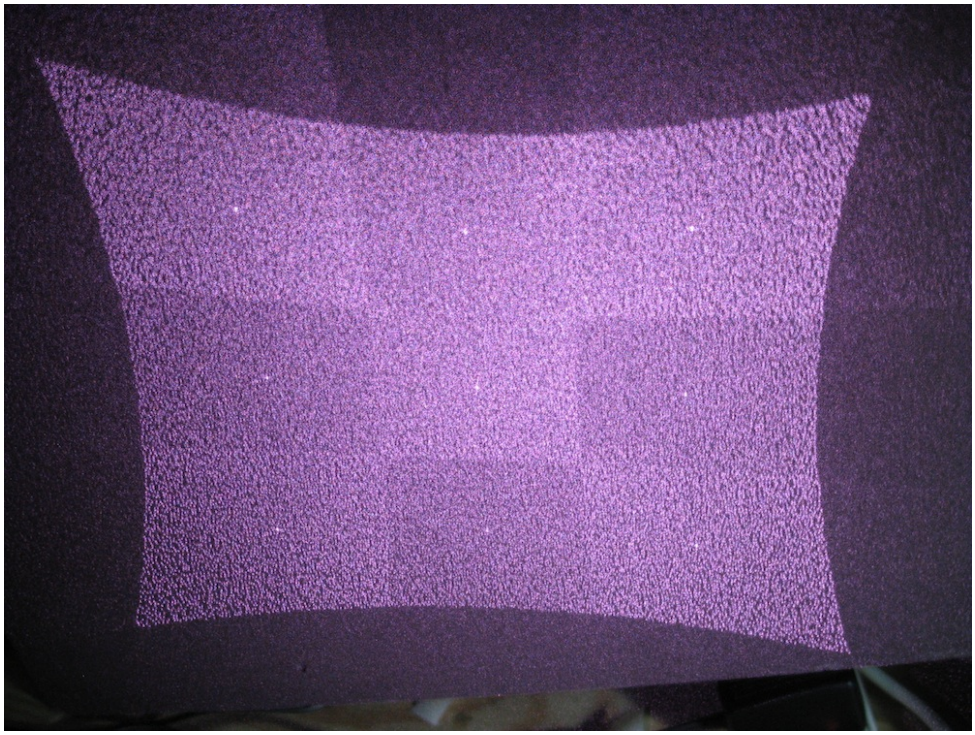


FIGURE 1.8 – Exemple de motif projeté dans la technologie de la lumière codée.

et analysé. L'observation de l'effet de déformation subi par les entités lumineuses projetées au contact de la surface, va permettre alors de déterminer la distance des points par rapport au capteur, comme le montre la Figure 1.9.

Cette technique a été développée par la compagnie (*PrimeSense* [4]) et utilisée dans le capteur (*Microsoft Kinect*V1). Les principales caractéristiques techniques de la caméra sont consignées dans le Tableau 1.1. Pour plus de détails concernant ce capteur, le lecteur peut se référer au site Internet dédié au capteur (Microsoft@).

3. https ://www.microsoft.com/fr-fr/
4. http ://www.primesense.com/

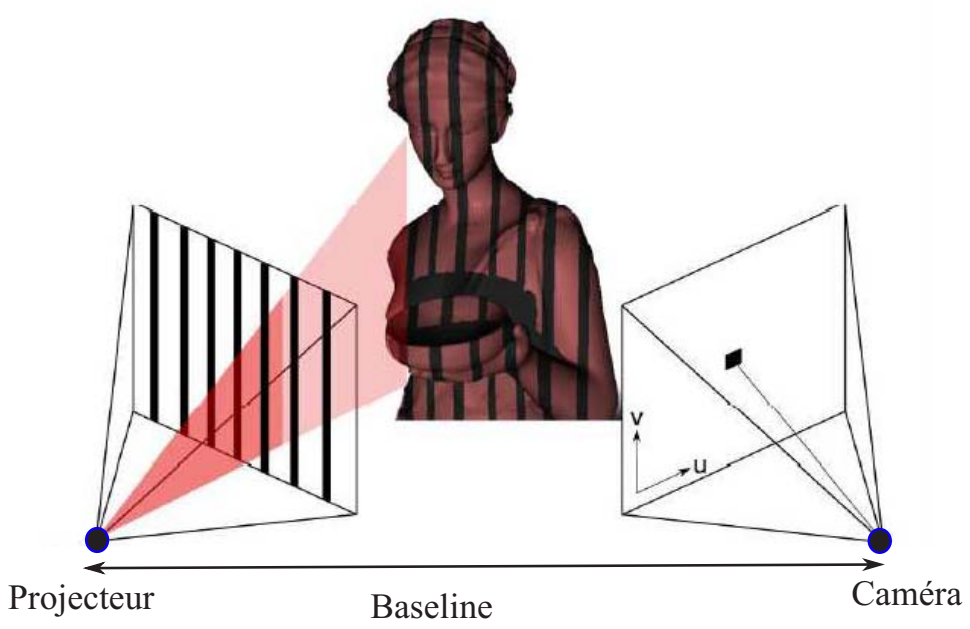


FIGURE 1.9 – Principe de la lumière structurée (source Sarbolandi et al. (2015)).

1.3.0.2 Différents types de données acquises

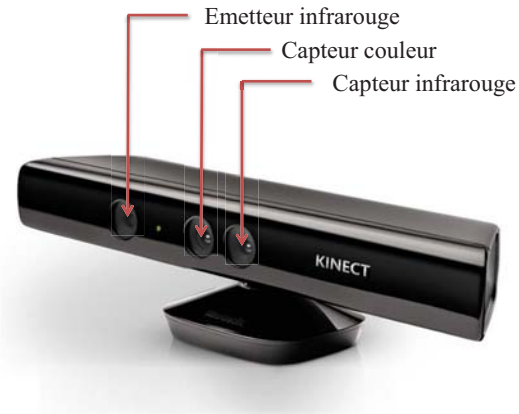


FIGURE 1.10 – Les composants d'une caméra KinectV1.

Le capteur Kinect V1 (Figure 1.10.) permet d'obtenir trois types de données :

- ***Une image infrarouge (IR)*** de la scène, en niveau de gris, qui fournit une information sur l'intensité du faisceau retourné par l'objet observé.
- ***Une carte de profondeur***, ou image de profondeur, émanant de la même caméra que l'image infrarouge, puisque c'est le traitement des faisceaux infrarouges qui permet de réaliser une mesure de distance entre le capteur et l'objet. La carte de profondeur se présente également sous la forme d'une image en niveaux de gris codée sur 16 bits de résolution 640 $\times$ 480 pixels. La particularité est que la valeur de chaque pixel de cette

FIGURE 1.11 – La surface mesurée à différentes distances. Le carré bleu représente le plan choisi pour effectuer les mesures (source Khoshelham and Elberink (2012)).

image correspond directement à une mesure de distance entre le capteur et un point correspondant de la scène.

- ***Une image couleur*** qui correspond à l'image réelle de la scène. Elle est de la même résolution que les images de profondeur (640×480 pixels).

L'arrivée du capteur RGB-D *Kinect* V1 a fortement participé au développement de nouveaux algorithmes en vision par ordinateur. Néanmoins il présente quelques inconvénients. Il a une portée limitée et il ne fournit pas de mesure de profondeurs sur les matériaux très absorbants, très réfléchissants ou transparents. D'autres part les données de profondeurs fournies sont relativement bruitées.

Pour analyser la relation entre l'erreur aléatoire et la distance de l'objet par rapport au capteur, Khoshelham and Elberink (2012) ont mesuré une surface plane à différentes distances du capteur *KinectV1* allant de 0.5m à 5m (qui correspond approximativement à la portée minimale et maximale du capteur), avec un intervalle de 0.5m entre chaque mesure. Pour chaque nuage de points reconstruits, une même partie de la surface est sectionnée pour effectuer les mesures. (voir la Figure 1.11). Pour évaluer cette erreur aléatoire à différentes distances, les auteurs sélectionnent aléatoirement un nombre régulier d'échantillons (4500 échantillons) dans chaque plan, et calculent l'écart type des résidus sur les échantillons sélectionnés.

Les résultats de l'analyse de ces erreurs ont montré que :

- L'erreur aléatoire sur les mesures de profondeur augmente de façon quadratique avec la distance de la scène observée par rapport au capteur. Elle atteint 4cm à la distance maximale de 5 mètres. Le modèle théorique de l'erreur aléatoire est donné par l'équation suivante :

$$\sigma_z = \left(\frac{m}{fb}\right) Z^2 \sigma_{d'} \tag{1.44}$$

  Avec σ_z l'écart type de la mesure de disparité normalisée, et $\sigma_{d'}$ l'écart type de la mesure de profondeur, m le paramètre de la normalisation linéaire, f la distance focale, et b la baseline.
- La résolution de la profondeur diminue de façon quadratique avec l'augmentation de distance par rapport au capteur. A une distance de 5 mètres, l'écart perceptible entre les points dans la direction de profondeur (le long de l'axe optique) est de 7cm, alors qu'à une distance de 1 mètre, l'espacement entre les points est de 2mm seulement.
- Pour les applications de cartographie les données doivent être acquises à une distance de 1-3m du capteur. À des distances plus grandes, la qualité des données est dégradée par le bruit et les faibles résolutions des mesures de profondeur.

Pour plus d'informations sur l'influence de la distance de l'objet par rapport au capteur sur les mesures de profondeur, ainsi que l'influence de la qualité du calibrage sur ces mesures, le

FIGURE 1.12 – Le capteur KinectV2 en marche.

lecteur est invité à se reporter aux travaux réalisés par Khoshelham and Elberink (2012) sur ce sujet.

1.3.0.3 Le capteur Kinect V2

A la différence du premier modèle de Kinect, la version 2 (Figure 1.12.) se base sur la technique de mesure par temps de vol que nous allons expliquer brièvement. Le lecteur peut se référer à l'état de l'art de la thèse de Kahlmann (2007) pour plus de détails sur les systèmes temps de vol.

La technologie d'acquisition 3D par temps de vol est une méthode active et optique, c'est-à-dire qu'elle est basée sur l'émission puis la réception d'un signal lumineux. L'onde employée dans ce procédé peut cependant être modulée de deux manières différentes :

- les ondes pulsées, ou lumière pulsée. Un décalage de temps est alors mesuré (temps de vol). La célérité c de la lumière étant connue et constante, le calcul de la distance se veut directe. La relation qui lie directement la distance d au temps $\triangle$t est donnée par $d = c* \triangle t/2$
- les ondes modulées en amplitude de façon continue. Dans ce cas de figure, c'est un décalage de phase entre onde émise et onde reçue qui sera évalué : la relation qui lie le décalage de phase mesuré à la distance recherchée n'est alors pas directe.

La principale caractéristique qui a été modifiée par rapport à la précédente version est la résolution de l'image couleur qui est plus élevée par rapport l'image de profondeur et à l'image infrarouge. En effet la résolution de l'image de profondeur et de l'images infrarouge est de (512×512) pixels tandis que la résolution de l'image couleur est de (1920× 1018) pixels.
Les travaux de Wasenmüller and Stricker (2016) comparent les performances des deux versions du capteur Kinect en terme de précision des profondeurs fournies, pour une utilisation dans le cas de la reconstruction 3D, du SLAM où de l'odométrie visuelle. Ils ont démontré qu'à la différence de la KinectV1, le bruit sur les profondeurs de la KinectV2 est constant quelque soit la distance, sa modélisation est par conséquent plus simple. Néanmoins sur les surfaces plates, la KinectV1 est plus précise. Des expérimentations similaires ont été publiées dans les travaux

de Sarbolandi et al. (2015)

SDK de Microsoft La gestion du flux de données issues du capteur 3D Kinect est entièrement réalisée par ordinateur. Ainsi, l'installation préalable de ń pilotes ż effectuant le lien entre l'ordinateur et le capteur est nécessaire. Des bibliothèques de code gratuites téléchargeables sur Internet remplissent ce rôle, notamment Libfreenect [5] ou encore le pilote OpenNI [6].
Notons que OpenNI fournit le calibrage extrinsèque entre la caméra infrarouge et la caméra couleur.

5. https ://openkinect.org/wiki
6. http ://openni.ru/openni-sdk/

CHAPITRE 2

État de l'art des méthodes SLAM

L'algorithme SLAM (Simultanuous Localization and Mapping) consiste à estimer une carte d'un environnement tout en localisant simultanément la caméra dans cette même carte. Le SLAM visuel fait référence aux cas dans lesquels une ou plusieurs caméras sont utilisées comme capteurs d'entrée de l'algorithme. Dans ce chapitre, nous donnons un aperçu du problème du SLAM visuel et les solutions les plus largement utilisées à ce jour. Nous nous intéressons ensuite aux méthodes SLAM visuel qui exploitent les capteurs RGBD.

2.1 Le problème de localisation et de cartographie simultanée SLAM

Au cours des 30 dernières années, la communauté scientifique a réalisé des nombreux progrès dans le développement de cet algorithme, permettant des applications à grande échelle dans le monde réel et un transfert constant dans le monde industriel.

Bien que historiquement plus connu dans la communauté robotique, le problème formulé par le terme *SLAM* est fortement lié aux premières recherches dans la photogrammétrie où il est nommé *ajustement de faisceaux*, et dans la vision par ordinateur où il est appelé *structure à partir du mouvement* plus connu en anglais par *Structure from motion*, Triggs et al. (1999). Même si le problème est le même, la différence principale entre le SLAM et les autres formulations réside dans la solution incrémentale et les applications temps réel en robotique. Un exemple de situation où le SLAM serait typiquement appliqué est une plateforme robotique mobile qui nécessite une estimation de sa position dans un environnement inconnu, en temps réel, afin de pouvoir prendre des décisions de haut niveau pour la navigation et l'évitement d'obstacles.

Les premiers travaux sur le SLAM dans la communauté robotique, sont attribués à Smith and Cheeseman (1986), leur travaux ont permis de poser le problème initial du SLAM, en donnant les moyens d'estimer l'incertitude et la corrélation entre la positon du capteur et la structure de l'environnement. Par la suite, les approches par *filtre de kalman étendu EKF* (Davison (2003), Castellanos and Tardos (2012), Newman (1999)) sont devenues des solutions très utilisées pour le SLAM temps réel. Ces solutions présentent un certain nombre de limitations. L'EKF-SLAM

utilise une approximation linéaire. Ceci entraine nécessairement des erreurs et peut rendre le filtre inconsistant (ce qui se traduit par une sous-estimation de l'erreur commise). De plus le vecteur d'état dans ce type d'approches croit linéairement avec l'accroissement de la taille de la carte. La covariance croit d'une manière quadratique. Ce qui provoque l'augmentation de l'espace mémoire nécessaire pour estimer la carte en temps réel ainsi que la complexité des calculs. Cet inconvénient, restreint l'extensibilité de ces solutions sur de grandes cartes. Ces limitations ont conduit un grand nombre de recherches à se diriger vers des algorithmes SLAM plus évolutifs et adaptés aux grandes surfaces, notamment les travaux sur le filtrage local de la carte par Williams (2001) et par Tardós et al. (2002), et les travaux sur le filtrage compressé par Guivant and Nebot (2001).

Cependant ces techniques n'ont pas permis de supprimer la complexité quadratique de l'algorithme ; elles ont juste réduit le temps de calcul requis par un facteur constant. Notons une exception du SLAM hiérarchique proposé par Estrada et al. (2005), qui ont réussi à atteindre une complexité linéaire. Une autre limitation de l'approche EKF est que la distribution est supposée uni-modale ce qui signifie qu'il y a une inaptitude intrinsèque à ce que des hypothèses multiples soient naturellement représentées. Ceci limite l'application du filtre dans le scénario où une localisation globale est nécessaire.

L'avènement des méthodes Monte-Carlo (Dellaert et al. (1999)) comme solution au problème de localisation d'un robot, a apporté un moyen beaucoup plus évolutif et informatiquement favorable au problème de représentation de l'état. Autrement connu comme les *méthodes à filtre à particules*, ces techniques ont fourni une solution aux problèmes d'évolutivité associés aux systèmes de SLAM basés EKF. La plus citée de ces méthodes est le FastSLAM Montecarlo proposé par Montemerlo et al. (2002). L'idée principale dans leur système est la "Rao-Blackwellisation" du problème qui représente la distribution à posteriori comme un filtre à particules sur la trajectoire du robot, tout en allouant un filtre de kalman indépendant à chaque primitive associée à chaque particule. L'approche FastSlam apporte une solution au problème de localisation globale et au problème d'ambiguïté, en permettant l'approximation des distributions multimodales.

Bien que ce système se soit révélé être un succès pour plusieurs applications, Julier and Uhlmann (2001) ont démontré que sur de très longues trajectoires, les estimations de l'état fournies par les méthodes basées filtrage sont peu précises. Plus tard Strasdat et al. (2010) ont conclu que l'ajustement de faisceaux était plus efficace que les approches basées filtrage pour le SLAM visuel.

Une alternative aux méthodes basées EKF ou basées filtres à particules, sont les méthodes de *graphe de poses*. Ce type d'approches est devenu très populaire ces dernières années, montrant des capacités de cartographie à de très grandes échelles par rapport aux méthodes basées filtrage. Dans les méthodes SLAM par graphe de poses, le problème est représenté comme un ensemble de nœuds et d'arêtes, où les nœuds représentent les variables latentes (poses des capteurs, positions des primitives) et les arêtes représentent les mesures relatives entre ces variables. Une fois le graphe reconstruit, le problème revient à trouver la configuration des nœuds qui assure un maximum de cohérence avec les mesures.

La formulation du problème du SLAM par graphe de poses a été proposé par Lu and Milios (1997). Ils ont choisi d'inclure uniquement les poses des caméras dans le graphe. Pour la reconstruction de la carte les auteurs ont utilisé le balayage avec un *lidar*.

La formulation du SLAM par graphe de poses a mis plusieurs années pour être populaire en raison de la complexité relativement élevée de la résolution du problème de minimisation d'erreurs. Les avancées dans le domaine de l'algèbre linéaire et les améliorations des méthodes

d'optimisation ont permis à cette formulation de devenir l'une des plus utilisée. Par la suite un grand nombre de travaux sur les solutions SLAM basées graphes de poses a été publié (Olson et al. (2006), Kummerle et al. (2011), Konolige et al. (2010), Gutmann and Konolige (1999), Dellaert and Kaess (2006)). Une des qualités les plus importantes des approches basées graphe par rapport aux approches basées filtrage est le fait que l'optimisation, pour résoudre le problème du SLAM, conserve toutes les informations sur toutes les poses qui ont été enregistrées. Contrairement aux approches basées filtrage qui donnent moins d'importance aux poses précédentes.
Kaess et al. (2008) ont proposé le *iSAM* qui formule le problème du SLAM sous la forme d'un graphe factoriel (graph factor). Un graphe factoriel (Figure 2.1.) est un graphe bipartite composé de deux types de nœuds : des nœuds représentant les variables et des nœuds représentant les facteurs. Les nœuds variables sont connectés aux nœuds facteurs par des arrêtes dans le graphe, où chaque nœud facteur est fonction de toutes les variables auxquelles il est connecté. Une autre contribution de iSAM est de proposer une méthode efficace et précise pour la mise à jour incrémentale de la factorisation du système, en recalculant seulement les éléments de la matrice qui ont changé. Il n'est plus nécessaire de résoudre le système complet à chaque fois qu'une pose est rajoutée au graphe.

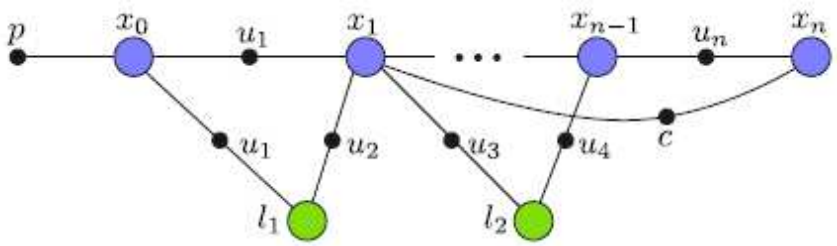


FIGURE 2.1 – **Exemple d'un graphe factoriel représentant le problème du SLAM.** Les cercles colorés représentent les nœuds variables (les x_i représentent les poses des caméras, les l_i représentent les positions des primitives), les cercles noirs représentent les nœuds facteurs (les u_i représentent les mesures d'odometrie), l'arc c représente une contrainte de fermeture de boucle par exemple.

Des travaux plus récents sur le SLAM basé graphe de poses se penchent sur des moyens robustes et efficaces de traiter les valeurs aberrantes dans le graphe (Agarwal et al. (2014), Olson and Agarwal (2013), Sünderhauf and Protzel (2012), Latif et al. (2013)). D'autres travaux sur la réduction de la complexité du SLAM ont été publiés (Johannsson et al. (2013)).

2.2 Le SLAM visuel

Les premiers systèmes SLAM utilisaient des capteurs tels que les radars, sonars et lidars pour percevoir l'environnement. Mais très vite des systèmes utilisant l'information visuelle se sont imposés comme références pour le SLAM. L'un des premiers SLAM basé vision est celui proposé par Davison and Murray (2002) qui pouvait reconstruire une carte précise à partir de primitives visuelles. Le même auteur dans Davison (2003), a proposé le MonoSLAM qui utilise une caméra monoculaire. Le MonoSLAM utilise une approche par EKF présentée précédemment dans le cas d'un SLAM visuel monoculaire. Ce système est capable d'estimer la pose de la caméra en temps réel, en estimant simultanément la position des primitives visuelles, tout en

conservant la covariance pleine du vecteur d'état. Une contribution importante du MonoSLAM consiste à proposer une nouvelle paramétrisation des primitives visuelles par leur profondeur inverse, détaillée dans Montiel et al. (2006).

Plusieurs systèmes SLAM se sont concentrés sur la contrainte temps réel. Ainsi le système PTAM proposé par Klein and Murray (2007) est un SLAM visuel temps réel qui permet d'accomplir des tâches de réalité augmentée très précises dans une petit espace. Pour ce faire PTAM sépare le processus d'estimation de la pose de la caméra et le processus de reconstruction de la carte dans deux threads parallèles indépendants. Ce choix est motivé par la proposition selon laquelle seules les primitives visuelles doivent être suivies à chaque image, la reconstruction de la carte peut se faire par intermittence dans un thread dont la priorité est moins importante. Leur système évite la propagation des erreurs de linéarisation couramment associées aux approches basées EKF en appliquant un ajustement de faisceaux sur un ensemble d'images clefs.

Une extension du PTAM aux grands environnements est proposée dans Castle et al. (2011). Les auteurs utilisent une approche de subdivision de la carte de l'environnement. Cette subdivision permet un fonctionnement sur des espaces plus grands et une visualisation précise de la réalité augmentée sur plusieurs cartes. D'autres solutions ont été proposées pour étendre le SLAM aux grands environnements, notamment le *FrameSLAM* que nous retrouvons dans les travaux de (Konolige and Agrawal (2008), Konolige and Bowman (2009), Konolige et al. (2010)). Les travaux de Sibley et al. (2010) combinent une approche de reconnaissance de lieux et un ajustement de faisceaux relatif, pour une localisation en extérieur. Enfin, les travaux de McDonald et al. (2013), utilisent les nœuds d'ancrage pour permettre un fonctionnement du SLAM sur les longues trajectoires.

Outre ces développements dans le SLAM monoculaire qui se basent principalement sur la détection et le suivi de primitives visuelles que nous appellerons par la suite approches éparses, de nouvelles approches dites *directes* ont été adoptées. Ces méthodes n'utilisent pas de primitives visuelles, elles se basent sur le suivi de texture. Engel et al. (2014) ont proposé le LSD-SLAM qui est l'acronyme de *Large-Scale Direct Monocular SLAM*. *LSD-SLAM*. C'est une technique qui estime la pose de la caméra par un alignement d'images en utilisant une approche multi-échelle combinée à un filtrage statistique et une estimation de cartes de profondeur semi-denses où seuls les points ayant une forte valeur de gradient sont pris en compte. Une optimisation globale est réalisée dans un graphe de poses composé d'images clés comme sommets et des transformations relatives entre ces images, comme arrêtes. Une extension au LSD-SLAM, *Omni LSD-SLAM*, a été proposée dans Caruso et al. (2015). Le développement de cette méthode a été motivé par l'analyse que le modèle sténopé ne permet pas d'avoir un grand champs de vision, ainsi en utilisant une caméra omnidirectionnelle, le champs de vision est plus grand, dépassant les 180ř. Ceci permet de réaliser des mouvements de rotation extrêmes ce qui n'était pas encore possible avec les systèmes SLAM précédents. Le même LSD-SLAM a été étendu pour prendre en compte une caméra stéréo. Dans le LSD-SLAM stéréo proposé par Engel et al. (2015), plusieurs avantages peuvent être notés : l'initialisation est instantanée, l'échelle est fixée automatiquement et les rotations importantes sont mieux gérées. De plus le LSD-SLAM stéréo inclut une correction de l'exposition automatique ce qui rend la fonction d'erreur invariante aux changements d'illumination.

Une autre approche de SLAM visuel qui nous intéresse dans nos travaux est celle proposée par Mouragnon et al. (2006a). Pour avoir un fonctionnement temps réel, les auteurs proposent un ajustement de faisceaux local qui considère uniquement les dernières images clefs et qui prend en compte un nombre limité de primitives 3D. Une autre contribution de ces travaux est d'exploiter la particularité des matrices engendrées dans l'ajustement de faisceaux telle que

proposée par Triggs et al. (1999). Les auteurs exploitent les structures creuses de ces matrices et utilisent le complément de Schur pour réduire la complexité des équations dans l'ajustement de faisceaux. Ceci permet de résoudre le problème d'une manière rapide et efficace. C'est ce SLAM qui sera utilisé dans nos travaux.

Toutes les méthodes de SLAM visuel listées dans cette section utilisent soit un capteur stéréoscopique soit une caméra monoculaire. Lorsque le capteur *Kinect* de Microsft est devenu disponible pour la communauté scientifique, il a été très vite adopté et utilisé dans plusieurs algorithmes SLAM. Cet engouement pour ce capteur se justifie par ses nombreuses caractéristiques favorables au SLAM. Ce capteur est le premier à fournir simultanément une carte de profondeur à haute résolution (640×480) à une fréquence de 30Hz, ce qui le rend très utile à la résolution d'un certain nombre de problèmes de perception robotique et en particulier dans le cas du SLAM visuel. De plus ce capteur est léger et peu couteux.

Dans ce qui suit, nous proposons une classification des approches exploitant les capteurs RGB-D en deux catégories : les approches denses qui utilisent tous les points de l'image et les approches éparses qui utilisent les points d'intérêt, pour l'estimation de la pose de la caméra. Dans les méthodes RGB-D éparses, il existe celles qui utilisent uniquement les points d'intérêt dont la mesure de profondeur est disponible [1] et celles qui exploitent les points d'intérêt dont la mesure de profondeur est fournie par le capteur et les points d'intérêt dont la mesure de profondeur est inconnue.

2.3 Les méthodes RGBD SLAM

Au cours des dernières années de nombreux articles utilisant les capteurs RGBD ont été publiés. Ces capteurs sont utilisés pour estimer la pose de la caméra et reconstruire la carte de l'environnement. Dans cette section nous présentons quelques-unes de ces méthodes.

2.3.1 Méthodes RGBD-SLAM denses

L'un des premiers systèmes de RGBD-SLAM dense abouti est le *KinectFusion* publié par Newcombe et al. (2011a). Ce système combine la popularité croissante des systèmes GPU avec la qualité élevée des cartes de profondeur fournies par le capteur *Kinect*. KinectFusion a été le premier système capable de produire des reconstructions 3D complètement denses de scènes de taille d'un bureau, en utilisant uniquement le capteur Kinect. Les modèles reconstruits par Kinectfusion sont précis à l'ordre du centimètre. La particularité importante et innovante de ce SLAM est d'utiliser une structure de données volumétrique pour représenter la reconstruction 3D au moyen d'une fonction de distance signée tronquée (*the truncated signed distance function* (TSDF)). La TSDF a été initialement introduite par Curless and Levoy (1996), elle fournit une représentation pratique de la scène qui permet d'avoir des moyens de calcul efficaces pour fusionner des cartes de profondeur. Le résultat de cette fusion est un modèle très lisse. La TSDF comme structure de données est très pratique pour les traitements parallèles, donc adéquate pour une utilisation sur GPU pour améliorer les performances calculatoires. Les poses des caméras sont estimées en alignant deux nuages de points par l'algorithme (Iterative closest point ICP) introduit par Besl and McKay (1992), la métrique utilisée dans Kinectfusion est une distance

1. Elles peuvent cependant exploiter tous les points de l'image pour reconstruire la carte de l'environnement, souvent hors ligne

point-plan comme introduit par Chen and Medioni (1992). Kinectfusion a inspiré plusieurs recherches, visant à robustifier le suivi et étendre la cartographie à des environnements plus larges. Zeng et al. (2012) ont présenté une extension du Kinectfusion à des volumes plus importants en remplaçant la représentation par voxels par une représentation par des octrees, permettant ainsi de reconstruire des environnements allant jusqu'à $8m \times 8m \times 8m$ de volume. Cette méthode augmente la probabilité de dérive dans la carte et n'utilise pas de fermeture de boucle ni d'optimisation globale pour corriger les erreurs accumulées dans la carte. Steinbrucker et al. (2013) proposent une représentation de la TSDF par octree multi-échelle, permettant de fournir une reconstruction dense et texturée d'un environnement aussi large qu'un couloir entier contenant plusieurs chambres couvrant une superficie de $(45m \times 12m \times 3.4m)$. Pour corriger les erreurs cumulées dans la carte, une optimisation de la trajectoire est appliquée dans un graphe de poses, suivi d'une nouvelle reconstruction de la carte. La pose de la caméra courante est estimée par alignement d'images en minimisant une erreur photométrique et une erreur géométrique. Chen et al. (2013) présentent une nouvelle structure de données hiérarchique qui permet de fusionner des cartes de profondeur de manière efficace, en optimisant l'espace utilisé. Cette méthode permet de cartographier des espaces illimités en volume, limités seulement par la mémoire disponible. Cependant cette approche ne permet pas de corriger la dérive et les erreurs dans la carte, aucune optimisation globale n'est appliquée. Enfin Whelan et al. (2013b) ont montré comment KinectFusion pouvait être étendu pour les grands environnements grâce à une représentation plus efficace de la mémoire, en déplaçant le volume représentant la scène avec le déplacement de la caméra.

Une autre alternative au problème du SLAM a été présentée dans Salas-Moreno et al. (2013) (SLAM++), où des objets connus de la scène, sont suivis et cartographiés dans un RGBD-SLAM. Une optimisation de la carte par graphe de poses est réalisée uniquement au niveau des positions des objets détectés, ce qui permet de réaliser une fermeture de boucle. Dans cette approche les auteurs donnent moins d'importance à la reconstruction d'une carte dense. Ils utilisent uniquement le nuage de points reconstruit par rétro-projection. Henry et al. (2013) utilisent des petits volumes "patch" pour segmenter la carte de l'environnement en des fonctions TSDF discrètes, chacune associée à une pose, qui est rigidement optimisée par une fermeture de boucle. Cette approche est similaire au SLAM++ où les volumes "patchs" peuvent être considérés comme analogues aux objets détectés.
D'autres résultats impressionnants sur la reconstruction de scènes en 3D ont été présentés par Zhou et al. (2013). Cette technique cible particulièrement les effets du bruit à hautes fréquences et les effets de la distorsion à basses fréquences, rencontrés souvent dans le cas d'utilisation des caméras RGBD. En reconstruisant des fragments de la scène qui sont ensuite alignés, une reconstruction très précise peut être obtenue. Cependant ce travail peut être réalisé uniquement hors ligne.

Audras et al. (2011) ont proposé un RGBD-SLAM qui calcule la pose de caméra en estimant la déformation (*the warping*) entre deux images successives. Cette méthode consiste à aligner deux images d'intensité. Elle est basée sur l'hypothèse qu'un même point observé par deux caméras, produit la même intensité lumineuse sur les deux images. Ainsi, tous les points de la première image sont rétro-projetés en 3D grâce à leur mesure de profondeur fournie par un capteur RGB-D ; la fonction de warping estime le mouvement de la caméra qui transforme les points dans la deuxième image en minimisant une erreur photométrique, après la projection de ces points sur cette dernière. Des travaux similaires ont été proposés par Kerl et al. (2013b) dans la solution qu'ils ont nommée DVO (Dense Visual Odometry). Ils estiment la pose de la caméra en estimant la fonction de déformation entre deux images successives en intégrant des

probabilités sur le bruit du capteur dans cette fonction de déformation.

D'autres approches denses expoitant la notion d'images clefs ont été proposées dans la littérature. Tykkälä et al. (2013) proposent un RGBD SLAM dense basé image clefs où la pose de la caméra est calculée en minimisant une erreur photométrique. De nouvelles images RGBD sont fusionnées avec les anciennes images clés présentes dans la carte pour améliorer la reconstruction. Une étape d'ajustement de faisceaux, optionnelle peut être appliquée pour optimiser les poses des caméras. Meilland and Comport (2013) proposent un modèle qui unifie les avantages d'une représentation volumétrique dense avec une représentation basée sur des images clés permettant une cartographie de l'environnement dense et précise sur de grandes trajectoires, sans avoir besoin de corriger la dérive ou d'appliquer une fermeture de boucle. Ils proposent de fusionner des images clés qui vont représenter un modèle 3D, afin de prédire une seule image de référence. La pose de la caméra est estimée en minimisant une erreur photométrique et une erreur géométrique, en exploitant uniquement des points saillants dans l'image. Kerl et al. (2013a) optimisent un SLAM basé images clés dans un graphe de poses.

De récents travaux proposent de cartographier des scènes non rigides, ainsi Newcombe et al. (2015) ont proposé le *DynamicFusion*, le premier SLAM dense capable de cartographier des scènes dynamiques en temps réel en utilisant des données de profondeur. Un modèle canonique de référence est reconstruit et raffiné incrémentalement à chaque nouvelle carte de profondeur dans une représentation volumétrique fixe par kinectfusion. Le nouveau rendu 3D est obtenu par le calcul d'une fonction de déformation qui permet de transformer chaque point du modèle canonique de référence vers l'image courante, cette dernière sera ensuite fusionnée au modèle de référence par la transformation inverse qui annule le mouvement de la scène. Cette technique permet d'avoir simultanément un nouveau modèle 3D représentant les nouvelles données, tout en raffinant l'ancien modèle qui a servi de référence. En dépit de son aspect innovant, cette technique présente quelques inconvénients. En utilisant un modèle de référence fixe, le modèle courant peut être très contraint, par conséquent les changements majeurs de forme et de topologie sont difficiles à adapter, de plus cette méthode calcule les correspondances en supposant des mouvements faibles entre les images successives ce qui peut être handicapant dans le cas de grands mouvements. Dou et al. (2016) ont présenté *Fusion4D*, une technique de SLAM dynamique, robuste aux grands mouvements et aux changements importants de topologie. L'originalité de cette approche est d'utiliser la notion de modèle de référence clé qui est réinitialisé suivant l'importance des mouvements dans la scène, dans le but de gérer les éventuels échecs de suivi. L'idée principale de cette méthode est de construire un modèle 3D clé par fusion volumétrique à partir d'un certain nombre de cartes de profondeur. Estimer ensuite, une fonction de déformation par une étape de mise en correspondance non rigide.

Les méthodes listées dans cette section utilisent tous les points de l'image à l'exception des travaux de Meilland and Comport (2013) qui n'utilisent que des points saillants. Ces méthodes sont souvent couteuses en temps de calcul et utilisent pour la plupart des puissances de calcul de type GPU. D'autre part, le volume de la scène à reconstruire est conséquent ce qui rend difficile son optimisation globale. Par conséquent, l'optimisation se limite souvent à la trajectoire uniquement. Enfin ces méthodes ne sont pas robustes dans les zones pauvres en structures géométriques.

L'autre type de solutions pour le RGBD-SLAM est les approches éparses. Celles-ci utilisent les points d'intérêt de l'image. Elles présentent l'avantage d'être rapides. Elles permettent d'optimiser simultanément la trajectoire et la reconstruction 3D. Dans ce qui suit nous présentons quelques-unes des méthodes RGBD-SLAM éparses et nous allons introduire notre méthode.

2.3.2 Méthodes RGBD-SLAM éparses

L'un des premiers travaux sur le RGBD-SLAM éparse est celui de Huang et al. (2011). Pour l'estimation de la pose de la caméra, ce système se base sur le suivi de primitives visuelles en utilisant un descripteur de type FAST. Le nuage de points dense quand à lui, est reconstruit hors ligne. Les paramètres des points 3D et les poses des caméras, sont raffinés par un ajustement de faisceaux en minimisant une erreur de re-projection des primitives de la carte éparse. Henry et al. (2012) dans leur RGBD SLAM utilisent l'appariement de primitives visuelles pour initialiser un algorithme de recalage (*Iterative Closest Point* (ICP)). Une carte de surfels [2] est construite et optimisée par la suite. Dans ces travaux, une comparaison entre une optimisation hors ligne par graphe de poses et par ajustement de faisceaux est présentée. Endres et al. (2012) ont proposé un travail similaire, où les primitives visuelles sont extraites et mises en correspondance pour estimer la pose de la caméra par RANSAC. Pour raffiner les poses estimées, une étape d'optimisation par graphe de poses est appliquée. Une représentation volumétrique basée sur les octrees est utilisée pour stocker la carte, créée en rétro-projetant tous les points de l'image dans le repère monde. Cette représentation de la carte est fournie par le pipeline *Octo-Map* de Hornung et al. (2013) qui inclut une capacité à prendre en compte les incertitudes sur les mesures. Il permet implicitement de représenter les espaces libres et les espaces occupés. Les travaux de Zhou and Koltun (2013), utilisent les mises en correspondance des primitives visuelles pour estimer la trajectoire de la caméra qui est ensuite optimisée dans un graphe de poses, pour permettre une reconstruction de la scène hors ligne. Cette reconstruction est réalisée en alignant des fragments de la scène.

Une représentation voxelique du volume est utilisée par Pirker et al. (2011). Dans leur solution GPSLAM, la pose de la caméra est estimée à partir des primitives visuelles éparses. Dans une représentation par graphe de poses, ils exploitent la reconnaissance de lieux et un ajustement de faiteaux par fenêtre glissante. Pour raffiner les paramètres estimés, ils proposent d'utiliser une grille d'occupation pondérée par des poids pour chaque caméra et pour chaque voxel.

Les méthodes citées précédemment construisent les points 3D à partir de leur profondeur par rétro-projection et utilisent ces points pour estimer les poses des caméras par ICP. Tous les autres points ne sont pas pris en compte. Généralement, les caméras RGB-D ont des portées limitées, des problèmes avec la lumière du soleil, les surfaces réfléchissantes ou très absorbantes. Ceci signifie que la mesure de profondeur n'est pas disponible sur tous les points de l'image. Scherer et al. (2012) ont proposé une extension de l'algorithme PTAM Klein and Murray (2007) pour prendre en compte, en plus des données visuelles, les données de profondeur. Ils ont suggéré plusieurs modifications du PTAM, la plus importante concerne l'ajustement de faisceaux. En effet ils ont étendu l'optimisation par ajustement de faisceaux en prenant en compte la profondeur des points dans la fonction de coût à minimiser. Cette dernière combine l'erreur de re-projection classique avec une erreur sur les profondeurs. Pour être combinées, ces erreurs de différentes métriques (la première en pixel et la deuxième en mètre) doivent être pondérées selon leurs incertitudes. L'incertitude sur les données de profondeur est fonction d'un facteur de pondération, que les auteurs ont fixé expérimentalement, mentionnant que ce paramètre doit être adapté selon la complexité de la scène. Les mêmes auteurs ont proposé dans un second papier (Fang and Scherer (2015)) une extension de ces travaux. En effet une des limitations du PTAM

2. Les surfels (abréviation de Surface elements) sont des paradigmes permettant de réaliser des rendus sur des objets complexes, de manière efficace et rapide, contrairement aux discrétisations de surface classiques, les surfels sont des primitives ponctuelles sans connectivité explicite. Les attributs des surfels comprennent la profondeur, la texture, la couleur, la normale et d'autres.

est le temps de calcul de l'ajustement de faisceaux global. Ce temps est conséquent quand le nombre de points 3D à optimiser devient important. Pour atténuer cette limite, ils proposent dans ce papier de remplacer l'ajustement de faisceaux global par une optimisation par graphe de poses, où la fonction de coût, combinant une erreur visuelle et une erreur sur les profondeurs est utilisée pour raffiner les branches du graphe. Ces branches représentent la pose relative entre chaque paire d'images clés. Des travaux similaires sont présentés dans Zhang et al. (2014). Les auteurs exploitent les données fournies par un capteur RGB-D quand celles-ci sont disponibles, les données issues d'un Lidar et les données purement visuelles sans information de profondeur disponible. Les points 3D peuvent être reconstruits soit par rétro-projection avec leur profondeur associée, soit par triangulation à partir de données visuelles uniquement. La pose de la caméra est estimée par recalage 2D/3D. Un ajustement de faisceaux est appliqué sur un certain nombre d'images, en parallèle des traitements effectués sur les images.

2.4 Conclusion

Comme présenté précédemment, il existe une multitude de systèmes SLAM utilisant les capteurs RGB-D. Les méthodes denses se focalisent sur la reconstruction d'une carte complète de l'environnement. Ces approches sont plus adaptées pour des applications de réalité augmentée ou des applications où le rendu 3D est nécessaire. Comme expliqué précédemment ces méthodes sont couteuses en temps de calcul, elles nécessitent des processeurs graphiques pour être temps réel. Plus l'environnement exploré est grand, plus le volume de la scène à reconstruire est important, faisant appel à une capacité de mémoire conséquente. En outre, la taille de la carte rend impossible une optimisation globale, par conséquent ces approches optimisent uniquement la trajectoire des caméras. De plus ces méthodes exploitent les cartes de profondeur et sont donc fragiles dans les zones pauvres en géométrie 3D.

La plupart des méthodes RGBD-SLAM éparses proposent d'utiliser uniquement les primitives visuelles ayant une information de profondeur associée. Les primitives visuelles dont l'information de profondeurs n'est pas connue, sont ignorées. Comme la portée des capteurs RGBD est limitée, ces méthodes ne sont pas robustes sur les grands espaces et dans les zones pauvres en géométrie et en texture. Quelques-unes de ces méthodes proposent donc d'utiliser les données visuelles ayant des informations de profondeurs associées et des données visuelles dont l'information de profondeur est inconnue (Scherer et al. (2012), Zhang et al. (2014)).

Dans la première partie de nos travaux, nous proposons une méthode RGBD-SLAM qui, de manière similaire à l'approche proposée par Scherer et al. (2012), repose sur un SLAM visuel revisité permettant d'intégrer l'information de profondeur. Ce RGBD-SLAM combine des primitives visuelles avec une information de profondeur connue et des primitives visuelles dont l'information de profondeur n'est pas disponible. Ceci permet de prendre en compte un maximum de points dans le processus de localisation et de reconstruction 3D. Tous ces points seront intégrés dans un ajustement de faisceaux local, au sein d'une fonction de coût, minimisant une erreur de re-projection 2D des points visuels et une erreur de re-projection fonction des profondeurs des points. Dans la deuxième partie nous allons exploiter la connaissance au préalable d'une partie de l'environnement pour améliorer la localisation du RGBD-SLAM proposé en première partie. Nous proposons de rajouter dans la fonction de coût de l'ajustement de faisceaux du RGBD-SLAM, deux autres contraintes ; la première va dépendre de l'appartenance des points 3D visuel à un plan connu de la scène et la deuxième contrainte va intégrer une erreur sur les profondeurs des autres points en combinaison avec cette contrainte par plans.

CHAPITRE 3

RGBD-SLAM : SLAM augmenté par l'information de profondeur

Nous présentons dans ce chapitre une nouvelle méthode RGBD-SLAM. L'information de profondeur est intégrée au niveau des différentes étapes du SLAM visuel. La principale modification porte sur l'ajustement de faisceaux, nous présentons un nouvel ajustement de faisceaux local qui permet de combiner des données de profondeur et des données visuelles dans une même fonction de coût.

Nous commençons ce chapitre par une discussion sur l'intérêt d'intégrer la mesure de profondeur dans le SLAM. Nous détaillons par la suite les différentes étapes où cette information interviendra et nous expliquons le choix d'une fonction de coût complètement exprimée en pixel dans l'ajustement de faisceaux.
Ces travaux ont donné lieu à une publication internationale Melbouci et al. (2015).

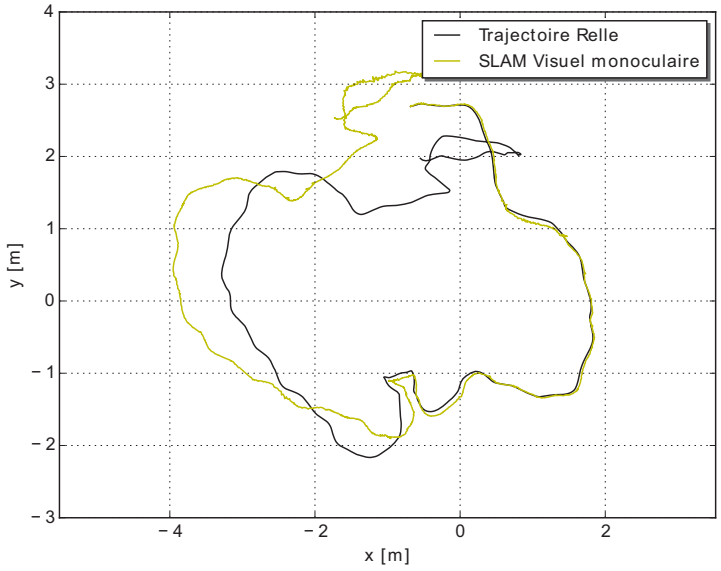
3.1 Introduction

Avec une seule caméra et sans connaissance a priori sur la géométrie de la scène, le SLAM visuel permet de localiser la caméra à un facteur d'échelle près seulement. Ce dernier est fixé initialement mais dérive au cours du temps (Figure 3.1).

Pour des applications d'aide à la navigation en milieu intérieur en particulier, il est nécessaire d'avoir une localisation à la bonne échelle et la plus précise possible. Le facteur d'échelle peut être fixé précisément si nous connaissons la distance de chaque point de l'image par rapport au capteur (*i.e.*profondeur de chaque point.). C'est cette information que les capteur RGBD fournissent instantanément et que nous allons exploiter pour fixer le facteur d'échelle et améliorer la localisation du SLAM.

Dans le cadre de nos travaux, nous visons une localisation en ligne, afin de permettre, à titre d'exemple, la navigation autonome de robots ou de drones, en intérieur. Par conséquent l'intégration de la mesure de profondeur dans le SLAM ne doit pas détériorer les performances calculatoires de ce dernier. D'autre part, pour garantir des capacités d'embarquabilité, l'algorithme doit fonctionner sur une architecture de calcul légère de type CPU, sans utilisation de proces-

(a)



(b)

FIGURE 3.1 – **Illustration de la dérive du SLAM visuel monoculaire sur les grandes trajectoires**. (a) : Exemple d'une image couleur de la scène observée, (b) : Évolution de l'erreur de position de la caméra.

seur graphique (*GPU*). Une autre difficulté découle, ainsi que nous l'avons vue précédemment, de l'indisponibilité de la mesure de profondeur dans certains environnements, en raison de la portée limitée du capteur et de la nature de l'environnement (la nature des matériaux des objets observés) comme le montre la Figure 3.2.

Ainsi pour ces raisons, nous avons choisi de développer un RGBD-SLAM pouvant s'affran-

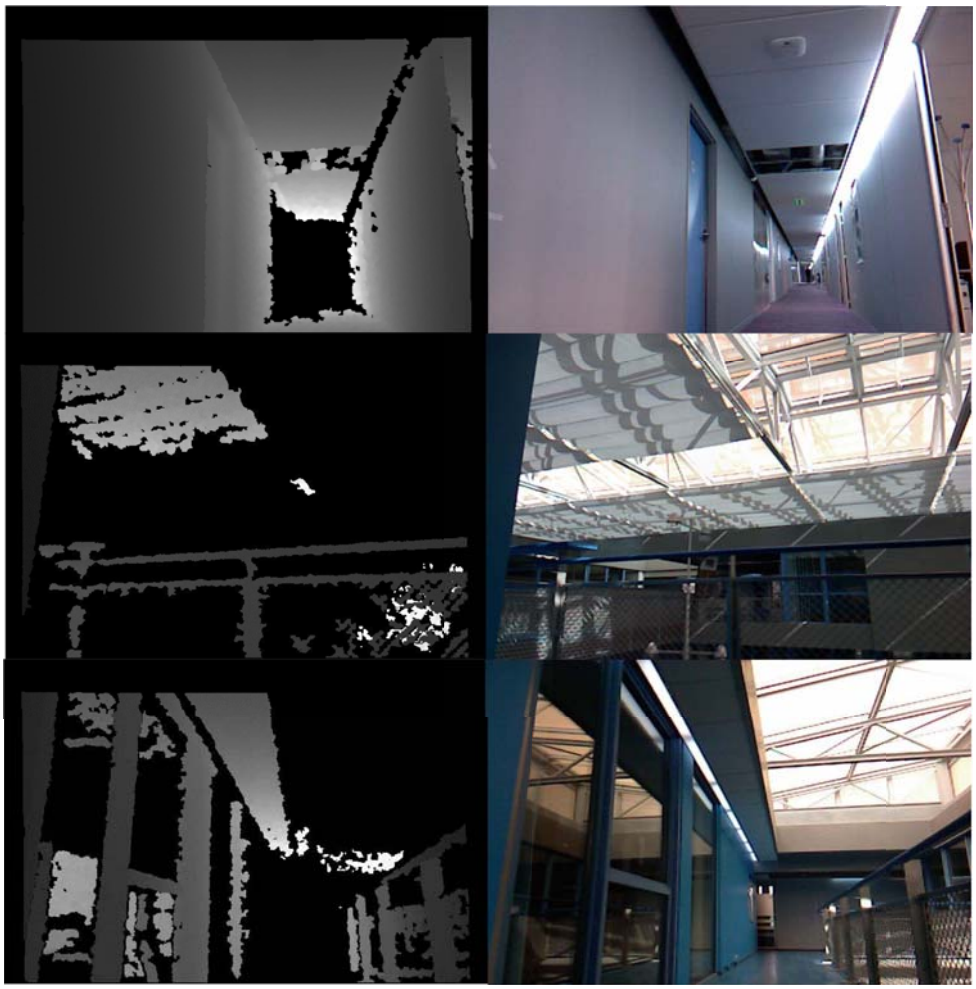


FIGURE 3.2 – Exemple d'environnements difficiles, où la mesure de profondeur est absente à certains endroits (grands environnements, surfaces vitrées, surfaces très absorbantes). A droite : les images couleurs, à gauche : les images de profondeur associées.

chir de l'information de profondeur quand celle-ci n'est pas disponible.
Pour les points dont la mesure de profondeur est indisponible, nous utilisons les équations classiques de la triangulation (voir section 1.2.2).

Adopter cette stratégie permet de bénéficier de plusieurs avantages :

- pouvoir étendre ces travaux à d'autres types de capteurs (comme les capteurs laser combinés à des caméras) qui fournissent l'information de profondeur à certains points seulement.
- appliquer notre algorithme aux grands environnements ou aux environnements extérieurs où l'information visuelle est plus pertinente que l'information de profondeur.
- pouvoir basculer en mode SLAM visuel monoculaire automatiquement, dès que la mesure de profondeur est absente.
- possibilité de combiner une localisation dans des environnements intérieurs et extérieurs.

Dans la suite de cette section, nous allons détailler les différentes étapes du SLAM modifié pour prendre en compte l'information de profondeur. Nous détaillerons la nouvelle fonction de coût de l'ajustement de faisceaux et les choix réalisés pour respecter la contrainte de temps réel. Une étude comparative avec d'autres approches de l'état de l'art, en terme de précision et temps de calcul, sera présentée dans la section 4.3.2.2.

3.2 Intégration de la mesure de profondeur

Comme introduit précédemment, la mesure de profondeur [1] est intégrée dans différentes étapes du SLAM (voir la Figure 3.3).

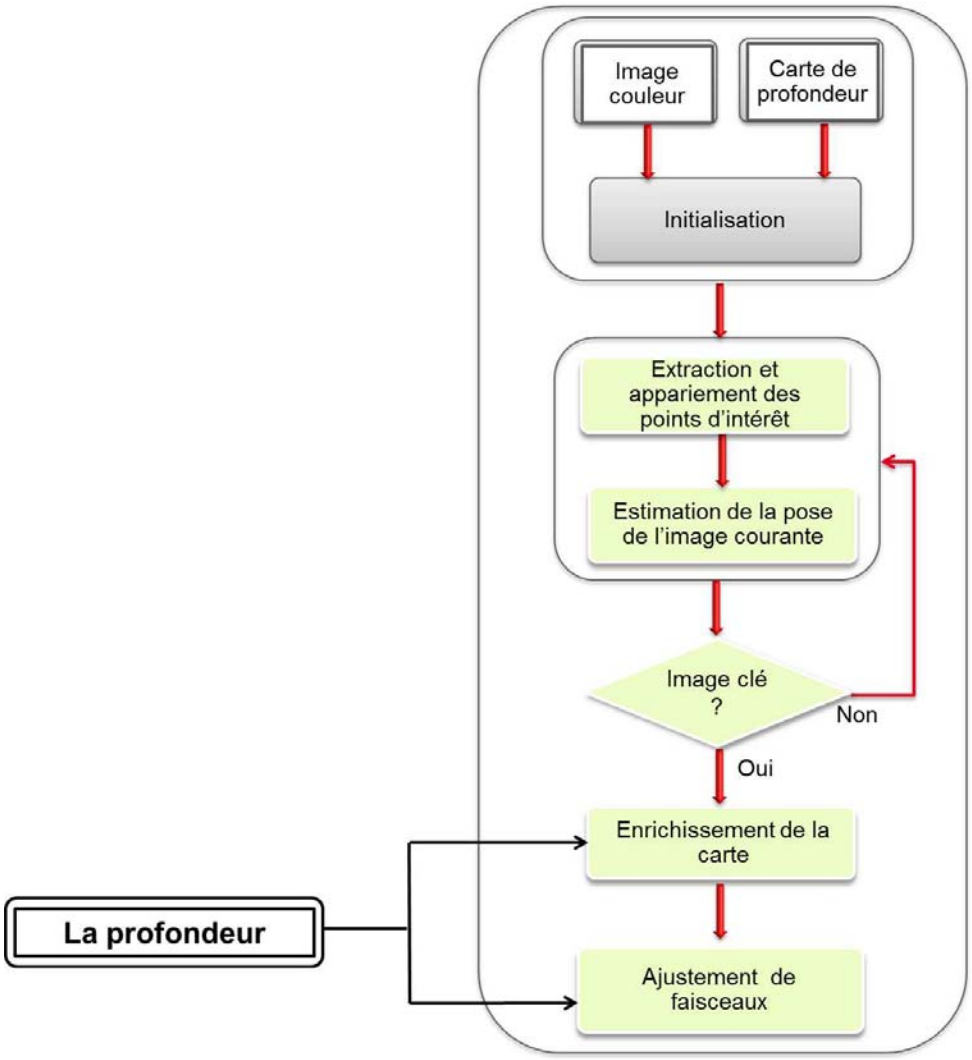


FIGURE 3.3 – RGBD-SLAM : Le SLAM visuel intégrant la mesure de profondeur.

3.2.1 Initialisation et mise à jour de la carte

— **Initialisation**
Le SLAM visuel monoculaire nécessite au moins trois images clés pour s'initialiser au moyen de ***l'algorithme des 5 points*** proposé par *Nister et al.* Nistér (2004).
La disponibilité d'une carte de profondeur va permettre de s'affranchir de cette étape. Une seule image et une seule carte de profondeur sont nécessaires pour construire le

1. Notons qu'une étape d'association profondeur/point est nécessaire pour pouvoir exploiter les profondeurs fournies par un capteur 3D. Le but étant de pouvoir associer directement à chaque pixel représentant la scène capturée une information de distance au capteur, sous forme matricielle. Tout au long de nos travaux nous utilisons le framework $OpenNI^6$ pour réaliser ces associations et ainsi exploiter les données fournies par le capteur *Kinect*.

nuage de points initial. Nous créons donc ce nuage de points à partir du premier couple d'images RGB et D acquis. Pour chaque point d'intérêt dont l'information de profondeur est disponible, un point 3D est généré à partir de ses coordonnées image et sa mesure de profondeur avec l'équation 3.1 (figure 3.4). Ce point 3D exprimé dans le repère de la caméra est ensuite transformé dans le repère monde avec l'équation 1.28. De cette

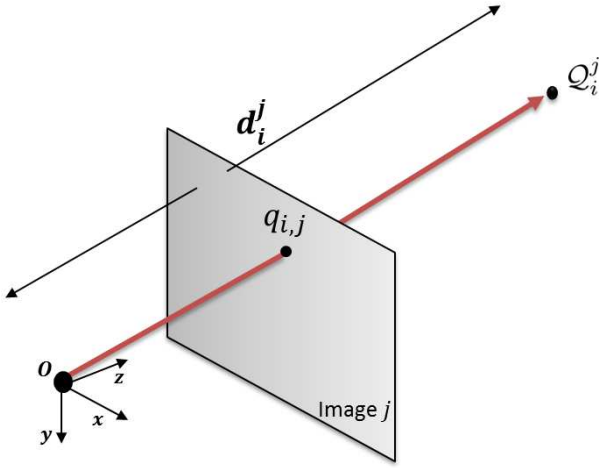


FIGURE 3.4 – Initialisation du RGBD-SLAM par l'information de profondeur par rétro-projection des primitives : La position 3D $\mathcal{Q}_i^j$ de chaque primitive $q_{i,j}$ est obtenue à partir de sa coordonnée image et de sa mesure de profondeur d_i^j.

manière, le facteur d'échelle est donné par les profondeurs des points et les poses des caméras seront à la bonne échelle dès le départ de l'algorithme.

— **Enrichissement de la carte de l'environnement**
Dans la version du SLAM précédente, l'enrichissement de la carte est réalisé à chaque image clef, en triangulant les points d'intérêt et en utilisant les poses d'au moins deux caméras clés (voir la section 1.2.2). Pour plus de robustesse et de précision dans les calculs numériques, il est préférable de réaliser cette triangulation sur plus de deux images. Ceci présente l'inconvénient que les points d'intérêt soient mis en correspondance avec toutes les images clés intervenant dans le calcul, ce qui réduit le nombre de points pouvant contribuer à l'enrichissement de la carte.
Dans le cas où l'information de profondeur est disponible, cette étape de triangulation est évitée. En effet à chaque nouvelle image clé, pour les points d'intérêt dont l'information de profondeur est disponible, des points 3D seront rajoutés à la carte par rétro-projection. Ceci est réalisé avec l'équation 3.1. Nous obtenons ainsi, un ensemble de points 3D auxquels une information de profondeur est associée. Toutefois dans le cas où cette information de profondeur n'est pas fournie, nous continuerons à reconstruire les points par triangulation.

$$\mathcal{Q}_i = \mathsf{P}_j{}^{-1}\pi^{-1}(\mathbf{q}_{i,j},\ \mathrm{d}_i^j) \tag{3.1}$$

où : $\mathcal{Q}_i$ est le point 3D exprimé dans le repère monde, P_j est la pose de la caméra j, $\mathbf{q}_{i,j}$ est le point 2D observé dans l'image j et d_i^j est la profondeur associée au point $\mathbf{q}_{i,j}$.

3.2.2 Ajustement de faisceaux

Dans cette section, nous nous intéressons à la manière d'intégrer la mesure de profondeur dans le processus d'optimisation par ajustement de faisceaux. Nous introduisons une nouvelle fonction de coût tenant compte à la fois des données visuelles sans information de profondeur associée et celles dont la mesure de profondeur est fournie par le capteur.

Ainsi que nous l'avons expliqué dans la section 1.2.4, l'ajustement de faisceaux local optimise les dernières poses de la caméra et la structure 3D de la scène (N_p points 3D), en minimisant une erreur de projection 2D dans les dernières images clefs. Cette erreur est la différence entre la projection d'un point 3D $\mathcal{Q}_i$ dans l'image I_j , et son observation correspondante $\mathbf{q}_{i,j}$ sur cette même image.
Nous rappelons ci-dessous la fonction de coût définie dans l'équation 1.38 :

$$\begin{aligned} f(\{\mathsf{P}_j\}_{j=1}^{\mathcal{N}_c}, \{\mathcal{Q}_i\}_{i=1}^{\mathcal{N}_p}) &= \sum_{1\leq i\leq \mathcal{N}_p} \sum_{j\in\mathcal{A}_i} \left\| \mathbf{q}_{i,j} - \pi(\mathsf{K}\mathsf{P}_j \tilde{\mathcal{Q}}_i) \right\|^2 \\ &= \sum_{1\leq i\leq \mathcal{N}_p} \sum_{j\in\mathcal{A}_i} \left\| \mathbf{f}_{i,j} \right\|^2 \end{aligned} \tag{3.2}$$

où $\mathcal{A}_i$ est l'ensemble des images observant le point $\mathcal{Q}_i$.

La mesure de profondeur est intégrée comme une contrainte additionnelle dans la fonction de coût. Nous avons choisi d'intégrer cette contrainte sous forme d'une erreur de reprojection. Le principe de cette contrainte est illustré par la figure 3.5 et s'explique ainsi :

Une caméra k génère un point 3D $\mathcal{Q}_i^k$ à partir du point 2D $\mathbf{q}_{i,k}$ et de sa profondeur d_i^k. $\mathcal{Q}_i^k$ est exprimé dans le repère de la caméra k. Ce point 3D est ensuite transformé dans le repère monde et projeté sur l'image j l'observant. En utilisant les équations 1.17 et 1.30, nous obtenons l'erreur 2D définie par :

$$\mathcal{E}_{2D} = \mathbf{q}_{i,j} - \pi\left(\mathsf{K}\mathsf{P}_j\mathsf{P}_k^{-1}\pi^{-1}(\mathbf{q}_{i,k},\ \mathrm{d}_i^k)\right) \tag{3.3}$$

Ce même point peut être généré par une autre caméra, et de la même manière il sera projeté sur l'ensemble des caméras de la plage $\mathcal{A}_i$ qui l'observent. Nous réitérons cette opération sur les $\mathcal{N}_c$ caméras à optimiser et les $\mathcal{N}_p$ points 3D pris en compte dans l'optimisation, dont la mesure de profondeur est disponible. Ceci donne une nouvelle fonction de coût de la forme suivante :

$$\begin{aligned} g\left(\{\mathsf{P}_j\}_{j=1}^{\mathcal{N}_c}\right) &= \sum_{1\leq i\leq \mathcal{N}_p} \sum_{j\in\mathcal{A}_i} \sum_{k\in\mathcal{A}_i}^{k\neq j} \left\| \mathbf{q}_{i,j} - \pi\left(\mathsf{K}\mathsf{P}_j\mathsf{P}_k^{-1}\pi^{-1}\left(\mathbf{q}_{i,k},\ \mathrm{d}_i^k\right)\right) \right\|^2 \\ &= \sum_{1\leq i\leq \mathcal{N}_p} \sum_{j\in\mathcal{A}_i} \left\| \mathbf{g}_{i,j}^k \right\|^2 \end{aligned} \tag{3.4}$$

La fonction de coût résultante est une combinaison de l'erreur de re-projection classique (équation 3.2) et l'erreur de re-projection associée aux points contraints par l'information de profondeur (équation 3.4), ce qui donne l'équation suivante :

$$h(\{\mathsf{P}_j\}_{j=1}^{\mathcal{N}_c}, \{\mathcal{Q}_i\}_{i=1}^{\mathcal{N}_p}) = \sum_{1\leq i\leq \mathcal{N}_p} \sum_{j\in\mathcal{A}_i} \left\| \mathbf{f}_{i,j} \right\|^2 + \sum_{1\leq i\leq \mathcal{N}_p} \sum_{j\in\mathcal{A}_i} \sum_{k\in\mathcal{A}_i}^{k\neq j} \left\| \mathbf{g}_{i,j}^k \right\|^2, \tag{3.5}$$

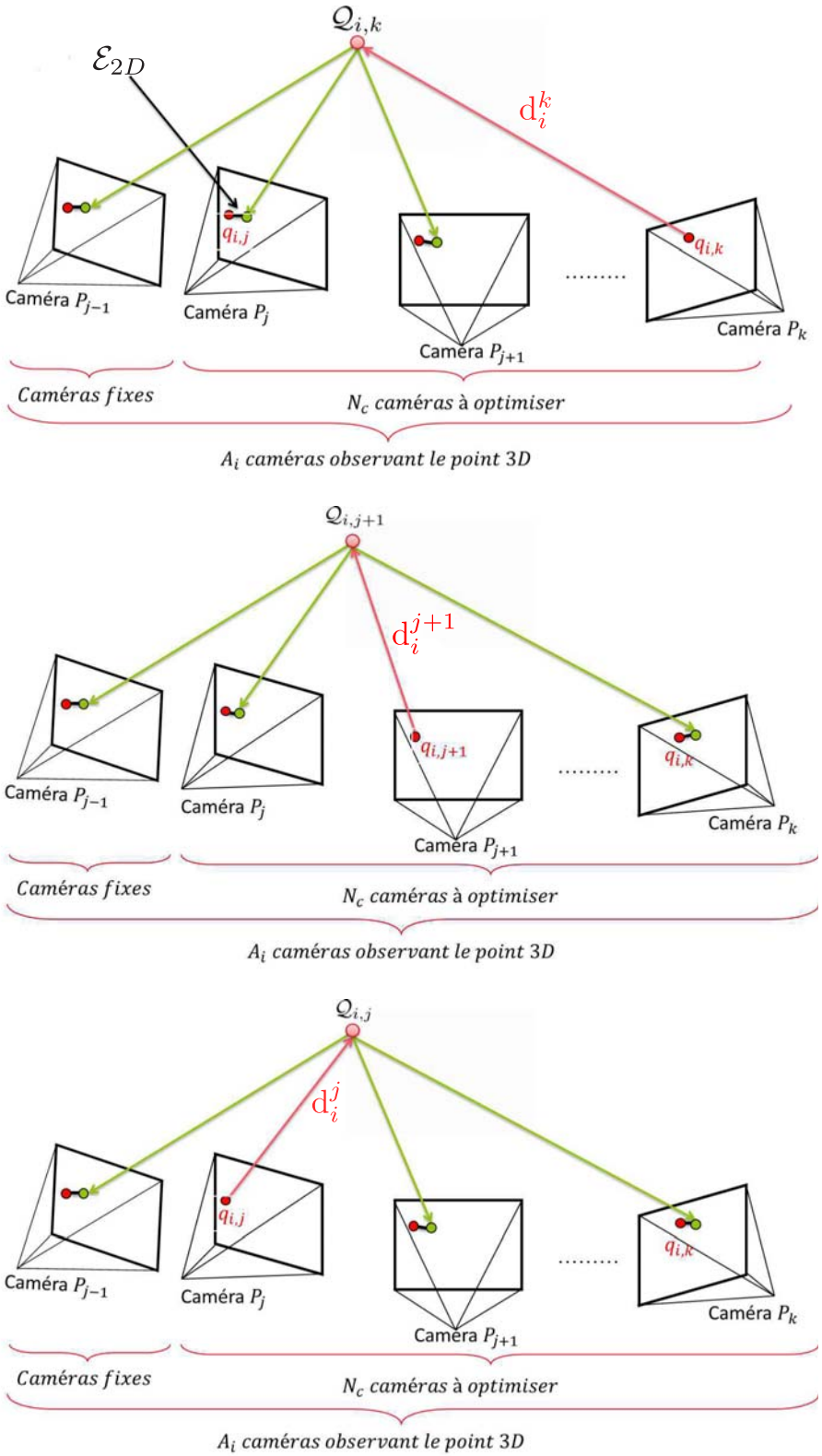


FIGURE 3.5 – **Ajustement de faisceaux contraint par la profondeur des points.** Un point 3D est généré par rétro-projection à partir de son observation 2D (flèches rouges) et sa mesure de profondeur. Ce point est projeté sur l'ensemble des images qui l'observent (flèches vertes) donnant lieu à une erreur de projection 2D dans l'image. Ce processus est réitéré sur l'ensemble des images $\mathcal{A}_i$.

L'équation 3.5 est minimisée par l'algorithme Levenberg- Marquart (Moré, 1978). La résolution complète de ce système ainsi que les équations engendrées, sont détaillées en annexe B.2.

Pour gérer les mesures aberrantes dues aux bruits sur les données ou aux erreurs dans le processus de détection et de mise en correspondance des primitives, l'erreur de projection dans l'équation 3.5 est remplacée par une erreur robuste en utilisant un M-estimateur.
L'estimateur robuste utilisé dans nos travaux est l'estimateur de Geman-McClure. Ainsi la norme $\mathbf{L}_2$ dans l'équation 3.2 et dans l'équation 3.4 est remplacée par une norme robuste définie par :

$$\begin{aligned} \rho(r,c) :&\mathbb{R} \longrightarrow [0..1] \\ & r \longrightarrow \frac{r^2}{(r^2+c^2)}, \end{aligned} \tag{3.6}$$

où c est le seuil de rejet du M-estimateur.

Le seuil c est estimé directement comme suit :

$$c = \text{médiane}(\mathbf{r}) + 1.41 \times \text{MAD}(\mathbf{r}), \tag{3.7}$$

où $\mathbf{r} = (r_1 \dots r_{N_p})^T$ est le vecteur des résidus de taille N_p concaténant les erreurs sur tous les points 3D pris en compte dans l'optimisation et la fonction $\text{MAD}(\mathbf{r})$ est définie par :

$$\text{MAD}(\mathbf{r}) = \text{médiane} \begin{pmatrix} |r_0 - \text{médiane}(\mathbf{r})| \\ . \\ . \\ |r_{N_p} - \text{médiane}(\mathbf{r})| \end{pmatrix}. \tag{3.8}$$

La fonction de coût de l'équation 3.5, combine des erreurs de même nature (pixels) par conséquent aucun facteur de pondération n'est nécessaire contrairement aux travaux de Scherer et al. (2012).
La nouvelle fonction de coût est donnée par :

$$h(\{\mathsf{P}_j\}_{j=1}^{\mathcal{N}_c}, \{\mathcal{Q}_i\}_{i=1}^{\mathcal{N}_p}) = \sum_{1\leq i\leq \mathcal{N}_p} \sum_{j\in\mathcal{A}_i} \rho\left(\|\mathbf{f}_{i,j}\|, c_f\right) + \sum_{1\leq i\leq \mathcal{N}_p} \sum_{j\in\mathcal{A}_i} \sum_{k\in\mathcal{A}_i}^{k\neq j} \rho\left(\left\|\mathbf{g}_{i,j}^k\right\|, c_g\right) \tag{3.9}$$

où $\rho(.,c))$ est l'estimateur de Geman-McClure, avec c le seuil de rejet estimé par le calcul de l'écart moyen absolu (median absolute deviation (MAD)).

3.2.3 Construction du système creux

Dans cette section nous expliquons le processus de construction de la matrice Jacobienne et de la matrice Hessienne intervenant dans l'ajustement de faisceaux du RGBD-SLAM. Nous montrons, qu'introduire la mesure de profondeur dans la fonction de coût de l'ajustement de faisceaux permet de conserver la structure creuse du système à résoudre.
La Figure 3.6. illustre un exemple d'un ajustement de faisceaux dans le cas du RGBD-SLAM où la fonction de coût combine deux erreurs de re-projection : une erreur de re-projection associée aux points sans information de profondeur et une erreur de re-projection associée aux points dont la mesure de profondeur est connue.

Nous nous intéressons à la structure de la matrice Hessienne associée à la fonction de coût $\mathbf{h}$ (équation 3.9). Rappelons que l'approximation de la matrice Hessienne associée à la fonction

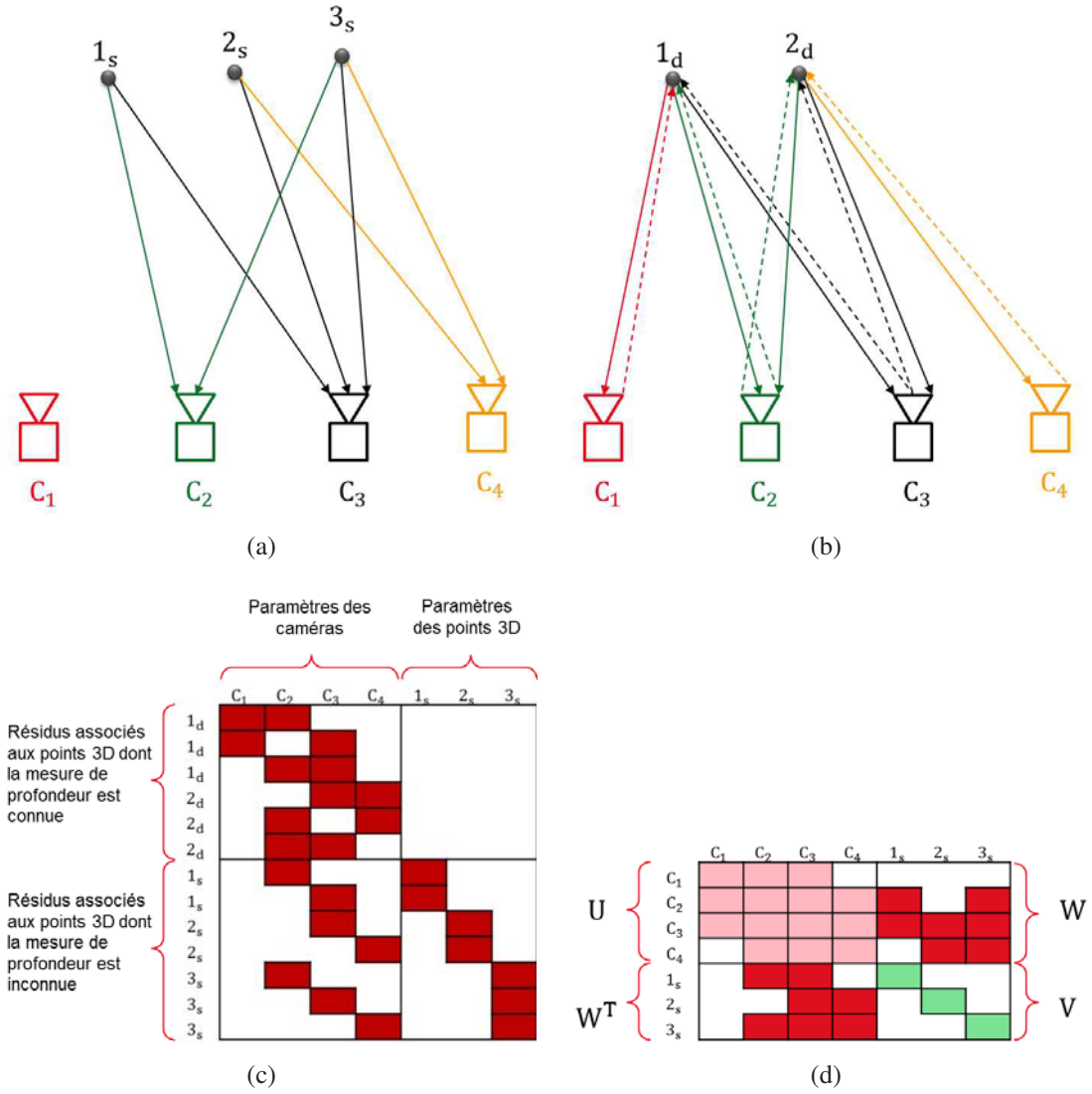


FIGURE 3.6 – **Exemple de matrices Jacobienne et Hessienne d'un ajustement de faisceaux du RGBD-SLAM.** (a) : les points 3D sans information de profondeur, (b) : les points 3D avec une information de profondeur : les flèches en traits discontinus représentent la génération des point 3D par rétro-projection et les flèches en traits continus représentent la re-projection des points sur les caméras les observant, (c) présente la matrice Jacobienne obtenue pour l'ensemble des points intervenant dans l'ajustement de faisceaux, (d) est la matrice Hessienne résultante selon la structure de Triggs et al. (1999).

de coût $\mathbf{h}$ est définie par $\mathbf{H}_h \approx 2\mathbf{J}_h^\top \mathbf{J}_h$, $\mathbf{J}_h$ étant la jacobienne associée à cette même fonction de coût.

En fusionnant les deux erreurs f (équation 3.2.) et g (équation 3.4.) associées respectivement à l'ajustement de faisceaux standard et à l'ajustement de faisceaux contraint par la profondeur, la structure de la matrice Hessienne (Figure 3.6.) est modifiée par rapport à la structure obtenue avec la fonction de coût standard de la Figure 1.5. La matrice U n'est plus diagonale par bloc et la matrice W est nulle pour les caméras qui n'observent que des points dont la mesure de profondeur est disponible.

Néanmoins nous obtenons un système de matrices à structures creuses, ce qui nous permet

d'implémenter l'ajustement de faisceaux en prenant en compte les structures par bloc comme proposé par Triggs et al. (1999). Cette structure doit nous permettre d'obtenir les performances temps réel visées.
Les détails des calculs des différentes dérivées pour la résolution de ce système sont présentés en annexe B.2.

3.3 Conclusion

Dans ce chapitre nous avons présenté une méthode de RGBD-SLAM. Celle-ci repose sur le formalisme d'un SLAM visuel dans lequel l'information de profondeur a été intégrée dans différentes étapes de ce dernier. L'idée majeure dans ces travaux est d'utiliser les profondeurs des primitives comme contraintes additionnelles dans l'ajustement de faisceaux. La nouvelle fonction de coût de l'ajustement de faisceaux prend en compte des contraintes multivues des erreurs de re-projection classiques et des contraintes multivues liées aux profondeurs des points. La solution que nous avons proposée permet de conserver les structures creuses des matrices engendrées par l'ajustement de faisceaux. Ces structures permettent d'avoir un fonctionnement temps réel de notre système.

Le chapitre suivant est consacré à l'évaluation expérimentale de notre méthode. Nous allons évaluer notre solution sur un ensemble de séquences de synthèse et de séquences réelles communes à l'état de l'art. Nous présenterons également une comparaison de notre méthode par rapport aux récentes approches de l'état de l'art, en terme de précision de la localisation de la caméra.

CHAPITRE 4

Évaluation expérimentale du RGBD SLAM

La présente section est consacrée à l'évaluation expérimentale de notre méthode. Elle a pour objectif de démontrer la qualité de la localisation et de la reconstruction obtenues avec RGBD-SLAM.
Les performances du RGBD-SLAM sont présentées, en comparant les résultats de la localisation de la caméra par rapport au SLAM visuel classique. Cette évaluation est réalisée sur des séquences de synthèse et sur des séquences réelles. Une comparaison avec d'autres méthodes de l'état de l'art en terme de précision et de temps de calcul est présentée dans la section 4.3.2.2.

Notons que dans la majeure partie de nos expérimentations, l'évaluation se fait en terme d'erreur de position absolue et/où relative de la caméra. L'erreur de rotation est introduite seulement dans l'évaluation sur les séquences réelles.

4.1 Les métriques d'évaluation

Les données disponibles à la sortie du SLAM sont les trajectoires des caméras et le nuage de points reconstruits (éparse dans notre cas). Dans nos travaux, nous validons nos algorithmes en évaluant les trajectoires obtenues. Cette évaluation est réalisée par le calcul de deux erreurs citées dans les travaux de Sturm et al. (2012).

Erreur de position absolue (Absolute Trajectory Error ATE)

Cette erreur permet d'évaluer la cohérence globale de la trajectoire en mesurant la distance absolue entre la trajectoire estimée et la trajectoire réelle.
Soit la trajectoire estimée $\mathsf{P}_1, ..., \mathsf{P}_n$, et la trajectoire réelle $\mathsf{G}_{t_1}, ..., \mathsf{G}_{t_n}$, avec n poses de caméras. Ces deux trajectoires ne sont pas forcément exprimées dans le même référentiel, elles doivent être alignées. Dans la solution proposée par Sturm et al. (2012), les auteurs estiment la transformation rigide S entre la trajectoire estimée $\mathcal{P}_{1:n}$ et la trajectoire réelle $\mathsf{G}_{t_{1:n}}$, par décomposition en valeurs singulières (SVD). L'erreur de position absolue à l'instant i est donnée par :

$$\mathcal{E}_i = \mathsf{G}_{t_i}^{-1}\mathsf{S}\mathsf{P}_i \tag{4.1}$$

L'erreur quadratique moyenne (Root Mean Square Error RMSE) est calculée à partir de ces erreurs :

$$RMSE(\mathcal{E}_{1:n}) = \left(\frac{1}{n} \sum_{i=1}^{n} \|Vect(\mathcal{E}_i)\|^2 \right)^{\frac{1}{2}}. \tag{4.2}$$

Où $Vect(\mathcal{E}_i)$ contient les composantes en translation du vecteur $\mathcal{E}_i$.
Notons qu'il est possible de calculer la moyenne et l'écart type dans l'évaluation.

Erreur de position relative (Relative Pose Error RPE)

L'erreur de position relative mesure la précision locale de la trajectoire sur un intervalle donné $\triangle$. Cette erreur correspond à la dérive de la trajectoire, qui est particulièrement utile pour évaluer l'odometrie visuelle. Elle est donnée par la formule suivante :

$$\mathcal{E}_i = \left(\mathsf{G}_{t_i}^{-1} \mathsf{G}_{t_{i+\triangle}} \right)^{-1} \left(\mathsf{P}_i^{-1} \mathsf{P}_{i+\triangle} \right) \tag{4.3}$$

Avec :
$\triangle$: le pas considéré pour le calcul de la dérive ; pour un système utilisant des images consécutives le pas est trivialement égal à 1. $\triangle = 30$ par exemple, donne la dérive par seconde pour des images enregistrées à $30Hz$.
Comme pour l'erreur absolue, nous calculons l'erreur quadratique moyenne sur $m = n - \triangle$ poses relatives.

$$RMSE\left(\mathcal{E}_{1:n}, \triangle\right) = \left(\frac{1}{m} \sum_{i=1}^{m} \|Vect(\mathcal{E}_i)\|^2 \right)^{\frac{1}{2}} \tag{4.4}$$

4.2 Méthode D-SLAM

Dans ces travaux, nous voulons comparer la qualité de la localisation obtenue en utilisant une erreur de re-projection dans l'image (exprimée en pixel) et celle obtenue en utilisant une erreur sur les profondeurs des points (exprimée en mètre). Pour cela l'erreur de re-projection dans la fonction de coût que nous proposons est remplacée par une erreur sur les profondeurs des points telle que proposée dans Scherer et al. (2012). Nous rappelons la fonction de coût proposée par Scherer et al. (2012), cette fonction de coût est utilisée par les auteurs dans un RGBD-SLAM reposant sur un algorithme PTAM modifié pour y intégrer l'information de profondeur, comme expliqué dans la section 2.3.2.
Soit un point 3D $\mathcal{Q}_i$ de la scène dont la mesure de profondeur d_i^j est donnée par le capteur à l'instant j. L'erreur considérée est la différence entre cette profondeur d_i^j et la profondeur estimée obtenue en exprimant le point $\mathcal{Q}_i$ dans la caméra j (équation 4.5) .

$$\mathcal{E}_d = \mathrm{d}_i^j - \left[\mathsf{P}_j \tilde{\mathcal{Q}}_i \right]_z \tag{4.5}$$

Cette erreur est intégrée dans le processus d'optimisation, la fonction de coût utilisée dans l'ajustement de faisceaux est la suivante :

$$d\left(\{\mathsf{P}_j\}_{j=1}^{\mathcal{N}_c}, \{\mathcal{Q}_i\}_{i=1}^{\mathcal{N}_p} \right) = \sum_{1 \le i \le \mathcal{N}_p} \sum_{j \in \mathcal{D}_i} \left\| \mathrm{d}_i^j - \left[\mathsf{P}_j \tilde{\mathcal{Q}}_i \right]_z \right\|^2 \tag{4.6}$$

Nous intégrerons cette fonction de coût dans notre RGBD-SLAM afin de pouvoir comparer les résultats obtenus avec celle-ci à ceux de notre méthode. Pour faciliter la compréhension

des résultats expérimentaux, nous notons D-SLAM, le RGBD-SLAM dont la fonction de coût (équation 3.4) est remplacée par l'équation 4.6.

Dans l'ajustement de faisceaux du D-SLAM, deux termes sont combinés :

- un terme lié aux résidus des points n'ayant pas de mesure de profondeur connue ; l'erreur à minimiser dans ce terme est une erreur de re-projection (équation 3.2), exprimée en pixels.
- un terme lié aux résidus des points ayant une information de profondeur associée ; l'erreur à minimiser dans ce terme est une erreur sur les profondeurs exprimée en mètres.

Pour pouvoir combiner ces deux termes de différentes natures, un facteur de pondération doit être utilisé. Ce facteur repose sur la covariances des profondeurs.

Pour modéliser le bruit sur les profondeurs, Scherer et al. (2012), utilisent un capteur *Kinect*V1 pour acquérir un ensemble d'images RGBD d'un mur plan. Ces images sont enregistrées à des distances allant de 0.5m à 3.5m. Le modèle du mur pour chaque image est estimé par l'algorithme LO-RANSAC (Chum et al. (2004)). Ce modèle est ensuite utilisé pour calculer les erreurs sur la profondeur. Les erreurs sont ensuite triées dans des bacs de 10cm de largeur. La mesure de l'écart type σ_d de chaque bac a permis de conclure que l'erreur sur les profondeurs des points est proportionnelle au carré de la profondeur. Des expérimentations similaires ont été réalisées par Khoshelham and Elberink (2012) et les conclusions sont identiques (voir la section 1.3.0.2).
Les auteurs ont modélisé l'écart type par un polynôme de second degrés comme suit :

$$\sigma(d) = \alpha d^2 \tag{4.7}$$

Le paramètre α est estimé en minimisant l'erreur relative du modèle par rapport aux écarts types mesurés.
Ainsi, pour combiner les deux termes de la fonction de coût du D-SLAM, l'erreur sur les profondeurs est pondérée comme suit :

$$\mathcal{E}_d = \frac{\mathrm{d}_i^j - \left[\mathsf{P}_j \tilde{\mathcal{Q}}_i\right]_z}{\alpha \mathrm{d}_i^j} \tag{4.8}$$

Où
α : est le facteur à appliquer aux profondeurs des points comme expliqué précédemment.

La fonction de coût qui intègre l'erreur sur les profondeurs est donnée par :

$$\begin{aligned} d\left(\{\mathsf{P}_j\}_{j=1}^{\mathcal{N}_c}, \{\mathcal{Q}_i\}_{i=1}^{\mathcal{N}_p}\right) &= \sum_{1\leq i\leq \mathcal{N}_p} \sum_{j\in\mathcal{D}_i} \left\| \frac{\mathrm{d}_i^j - \left[\mathsf{P}_j \tilde{\mathcal{Q}}_i\right]_z}{\alpha \mathrm{d}_i^j} \right\|^2 \\ &= \sum_{1\leq i\leq \mathcal{N}_p} \sum_{j\in\mathcal{D}_i} \left\| \mathbf{d}_{i,j}^k \right\|^2 \end{aligned} \tag{4.9}$$

4.3 Séquences utilisées et résultats

Pour évaluer les différents algorithmes, nous avons utilisé des séquences de synthèse et des séquences réelles qui représentent différents environnements. Aussi les séquences choisies sont soit générées par nos propres outils, soit tirées de bases de données de benchmarks, sur

lesquelles nous reviendrons par la suite. Pour l'ensemble de ces séquences nous disposons de la vérité terrain qui correspond aux points 3D et aux poses réelles des caméras.

4.3.1 Séquences de synthèse

4.3.1.1 Séquence "Bureaux"

La première séquence de synthèse utilisée dans l'évaluation est illustrée dans la Figure 4.2. Elle représente une trajectoire de 153m (Figure 4.1.), dans une grande pièce de taille $30\text{m} \times 19.32\text{m} \times 9.37\text{m}$ contenant des bureaux. Les images couleurs et les images de profondeur ont une résolution de (640×480) et ne présentent aucune distorsion.
Cette séquence est riche en géométrie 3D ; les profondeurs sont idéales, mais elle est pauvre en texture.

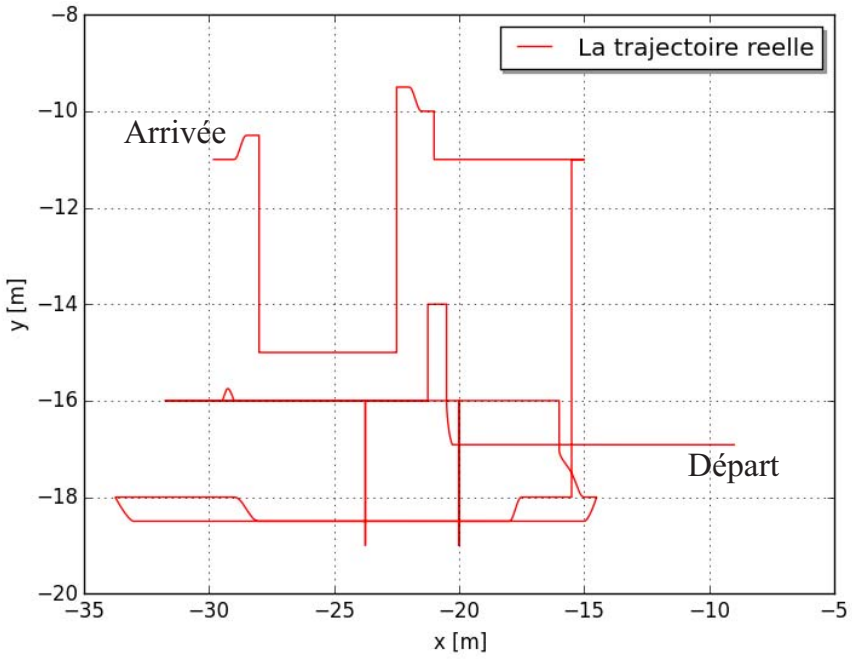


FIGURE 4.1 – La trajectoire réelle de la caméra, projetée sur le plan $(\mathbf{x}, \mathbf{y})$.

Sur cette séquence, trois algorithmes sont testés : le SLAM visuel, le RGBD-SLAM et le D-SLAM. Pour le SLAM visuel, nous avons modifié l'initialisation en utilisant les mesures de profondeur, par conséquent l'initialisation de ce dernier est similaire à celle du RGBD-SLAM (voir section 3.2.1). De cette manière l'échelle est fixée au lancement de l'algorithme.
Pour le D-SLAM la valeur du paramètre α assurant les meilleurs résultats sur cette séquence est égale à 0.003.

Résultats : Les résultats sur la précision de la localisation, obtenus sur cette séquence sont présentés dans la Figure 4.3. L'évaluation quantitative de ceux-ci est reproduite dans le tableau 4.1.

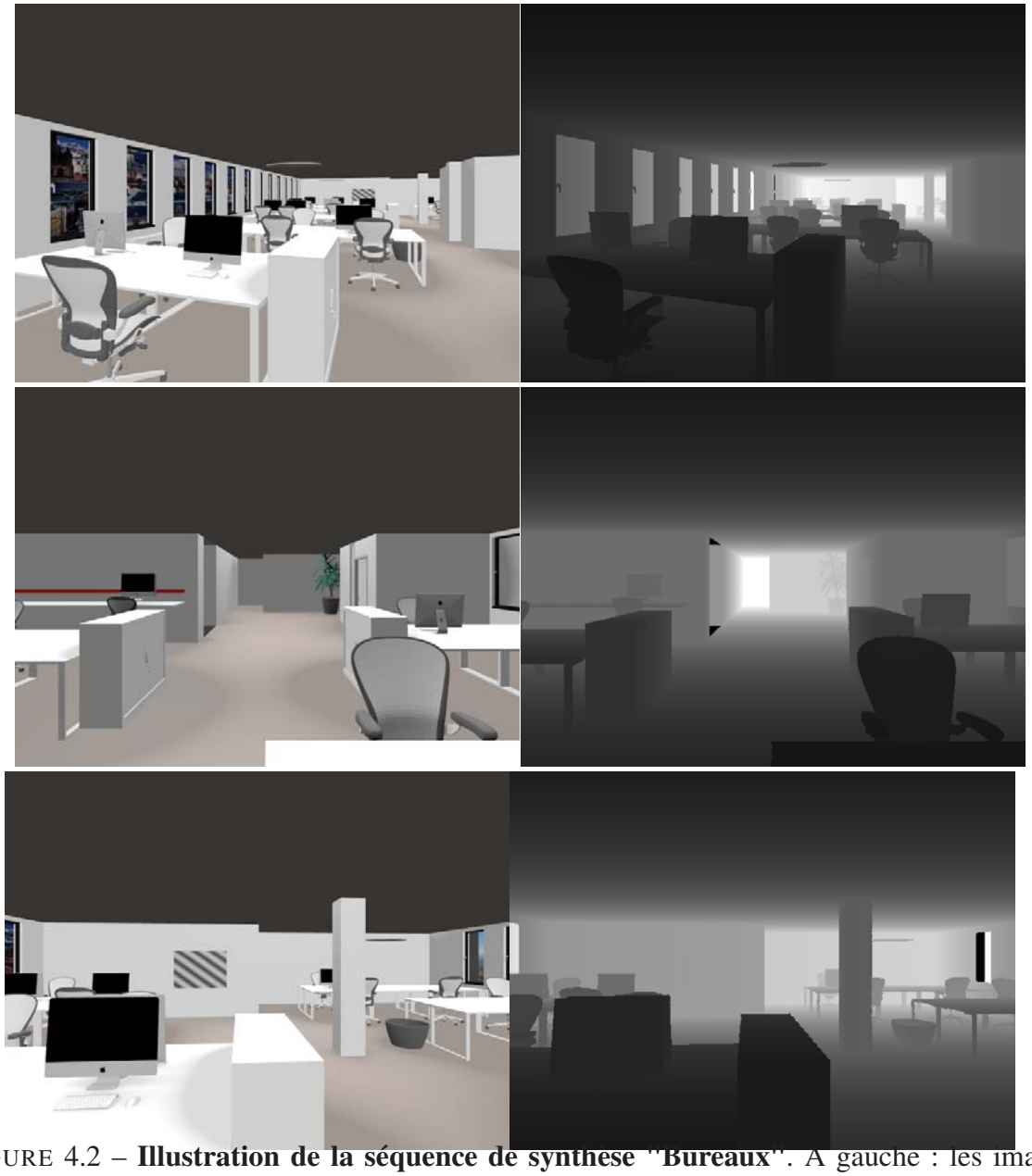


FIGURE 4.2 – **Illustration de la séquence de synthèse "Bureaux"**. A gauche : les images RGB utilisées, à droite : leurs cartes de profondeur associées.

Pour des mesures de profondeur parfaites, l'erreur sur la position des caméras obtenue avec notre algorithme est environ 20 fois inférieure à l'erreur de position obtenue dans le cas du SLAM visuel. Intégrer la mesure de profondeur a permis d'améliorer considérablement la localisation du SLAM visuel.

Erreurs [m] / Methodes	RMSE	STD	MAX
RGBD-SLAM	**0.052**	**0.039**	**0.239**
D-SLAM ($\alpha = 0.003$)	0.072	0.037	0.240
SLAM	0.961	0.575	1.940

TABLE 4.1 – Les erreurs sur la position absolue, obtenues sur la séquence de synthèse "Bureaux".

Une explication de la dérive importante du SLAM dans le cas de cette séquence, est la structure de la scène. Celle-ci contient des surfaces parallèles au mouvement de la caméra. Les positions des primitives visuelles glissent, ce qui conduit à des erreurs dans la détection de ces primitives, donc à des erreurs dans la mise en correspondance de celles-ci et aussi à des erreurs dans la reconstruction 3D des points. Tout cela impacte négativement le calcul de poses et la convergence de l'ajustement de faisceaux du SLAM. Contraindre les points par leur profondeur sur plusieurs vues (RGBD-SLAM), permet d'atténuer ce phénomène.

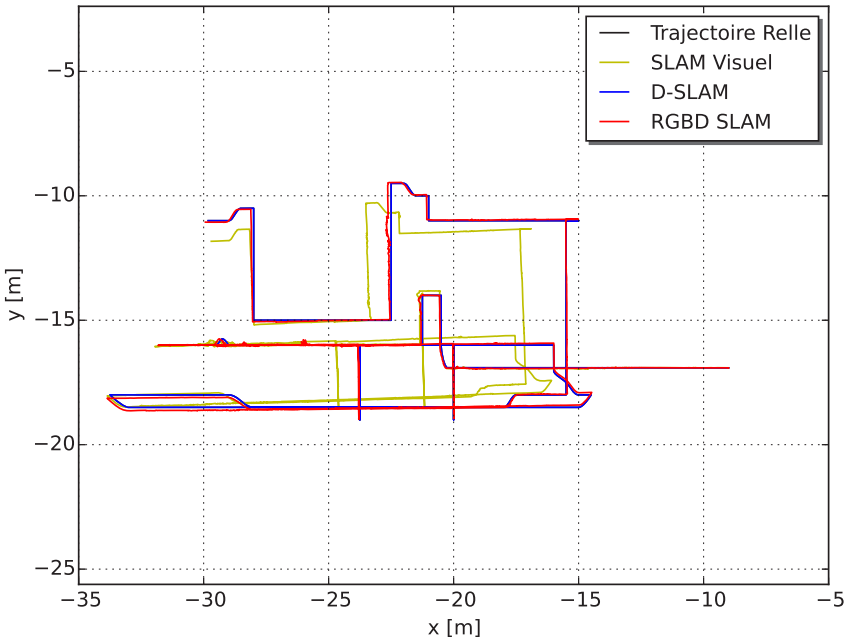
Pour le D-SLAM, nous constatons qu'en choisissant une valeur adéquate du paramètre α, les résultats obtenus sont satisfaisants ; l'erreur sur la position absolue est très faible pour une valeur de $\alpha = 3.\ 10^{-3}$. Néanmoins nous constatons une légère dégradation des résultats par rapport à ceux obtenus avec le SLAM-RGBD.

4.3.1.2 Séquence de synthèse "living room"

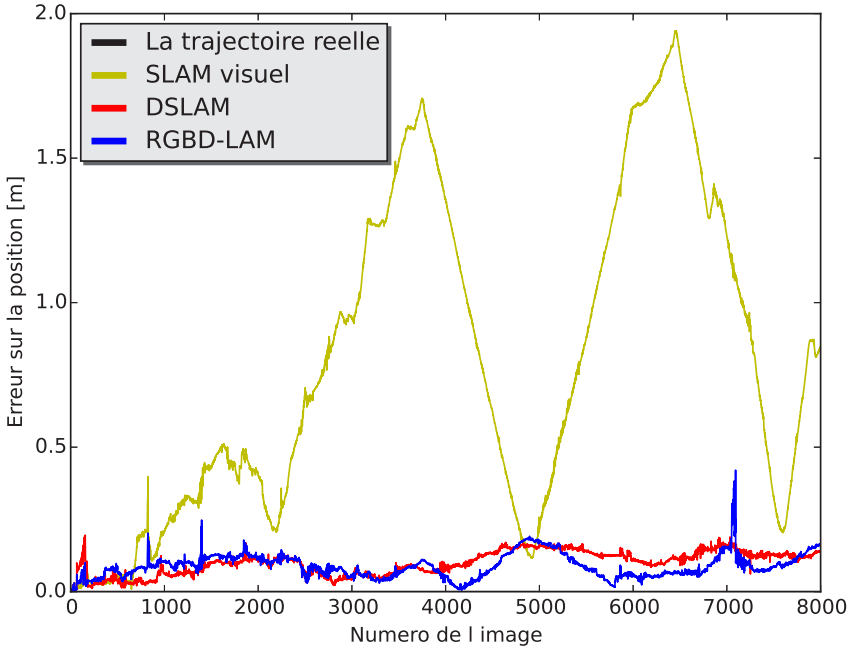
Le deuxième test porte sur des séquences de synthèses illustrées dans la Figure 4.4. Elle sont tirées d'un benchmark réalisé par Handa et al. (2014) et citées plusieurs fois par la communauté scientifique. Ces séquences contiennent des images RGB (640×480), des cartes de profondeur (640×480). Elles sont fournies avec un modèle polygonal complet de la scène, ainsi que les vérités terrain des poses des caméras. Pour modéliser le bruit du capteur *Kinect*V1 et fournir des données se rapprochant davantage de la réalité, les auteurs introduisent un bruit sur les images couleur et sur les données de profondeur.
Pour simuler le bruit du capteur *Kinect*V1 sur les données de profondeur, les auteurs s'inspirent des travaux de Barron and Malik (2013). Une première étape consiste à appliquer un décalage aléatoire sur les positions des pixels et à calculer les nouvelles valeurs par interpolation bilinéaire. Les valeurs de profondeur sont converties en disparité. Par la suite, un bruit gaussien est rajouté pour chaque disparité. La valeur de cette dernière est arrondie à l'entier le plus proche et convertie en mesure de profondeur.
Cette étape est résumée par l'équation suivante :

$$\hat{\mathbf{Z}}(x,y) = \frac{35130}{\lfloor 35130/\mathbf{Z}\left(x+n_x, y+n_y\right) + \mathcal{N}\left(0,\sigma_d^2\right) + 0.5 \rfloor} \tag{4.10}$$

où :
$\mathbf{Z}$ est la valeur de la profondeur initiale, n_x et n_y sont les valeurs des déplacements aléatoires



(a)



(b)

FIGURE 4.3 – **Les résultats du SLAM visuel, du D-SLAM et du RGBD-SLAM sur la séquence "Bureaux".** (a) : les différentes trajectoires obtenues, projetées sur le plan. (b) : l'évolution des erreurs de position absolue (ATE).

appliqués suivant l'axe x et l'axe y respectivement, σ_d^2 est l'écart type du bruit appliqué sur les profondeurs, la valeur 35130 est obtenue à partir de la baseline de la caméra.
Ce modèle garantit que les pixels ayant une petite valeur de disparité (points éloignés) sont plus affectés par le bruit que les pixels dont la valeur de disparité est importante (points proches du capteur).

Pour les images couleur, les auteurs appliquent un bruit suivant le modèle statistique proposé par Handa et al. (2012). Ce bruit est modélisé à partir d'une caméra réelle. La fonction de réponse de la caméra est obtenue en prenant des images d'une scène statique pour différents temps d'expositions.
La Figure 4.6. présente quelques images bruitées de la séquence de Handa et al. (2014).
Pour cette scène plusieurs trajectoires sont générées, nous testons les deux nommées kt1

FIGURE 4.4 – **Illustration de la scène issue de la base de données de Handa et al. (2014)** : différents phénomènes d'éclairages sont représentés dans ces images *e.g.* les ombres, les reflets, la lumières du soleil, la diffusion de la lumière.

et kt2. Ces trajectoires sont présentées dans la Figure 4.5 et leurs caractéristiques sont résumées dans le tableau 4.3.1.2. Notons que la trajectoire kt1 contient plus de mouvements brusques et plus de rotations importantes ce qui la rend plus difficile que la trajectoire kt2. Pour plus d'informations sur ces séquences, le lecteur peut se référer à l'article de Handa et al. (2014)

Caractéristiques / Séquences	Nombre d'images	longueur de la séquence
kt1	967	2.05m
kt2	882	8.43m

TABLE 4.2 – Les caractéristiques des séquences de synthèse "Living room".

Résultats : Comme précédemment nous comparons nos algorithmes : RGBD-SLAM, D-SLAM et le SLAM visuel. Le tableau 4.3 présente les différentes erreurs obtenues sur les deux trajectoires kt1 et kt2 et la Figure 4.7 représente l'évolution du RMSE sur celles-ci.

Sur les deux trajectoires de la scène "Living room", l'erreur de position du RGBD-SLAM est considérablement inférieure à l'erreur de position issue du SLAM-visuel, bien que les mesures

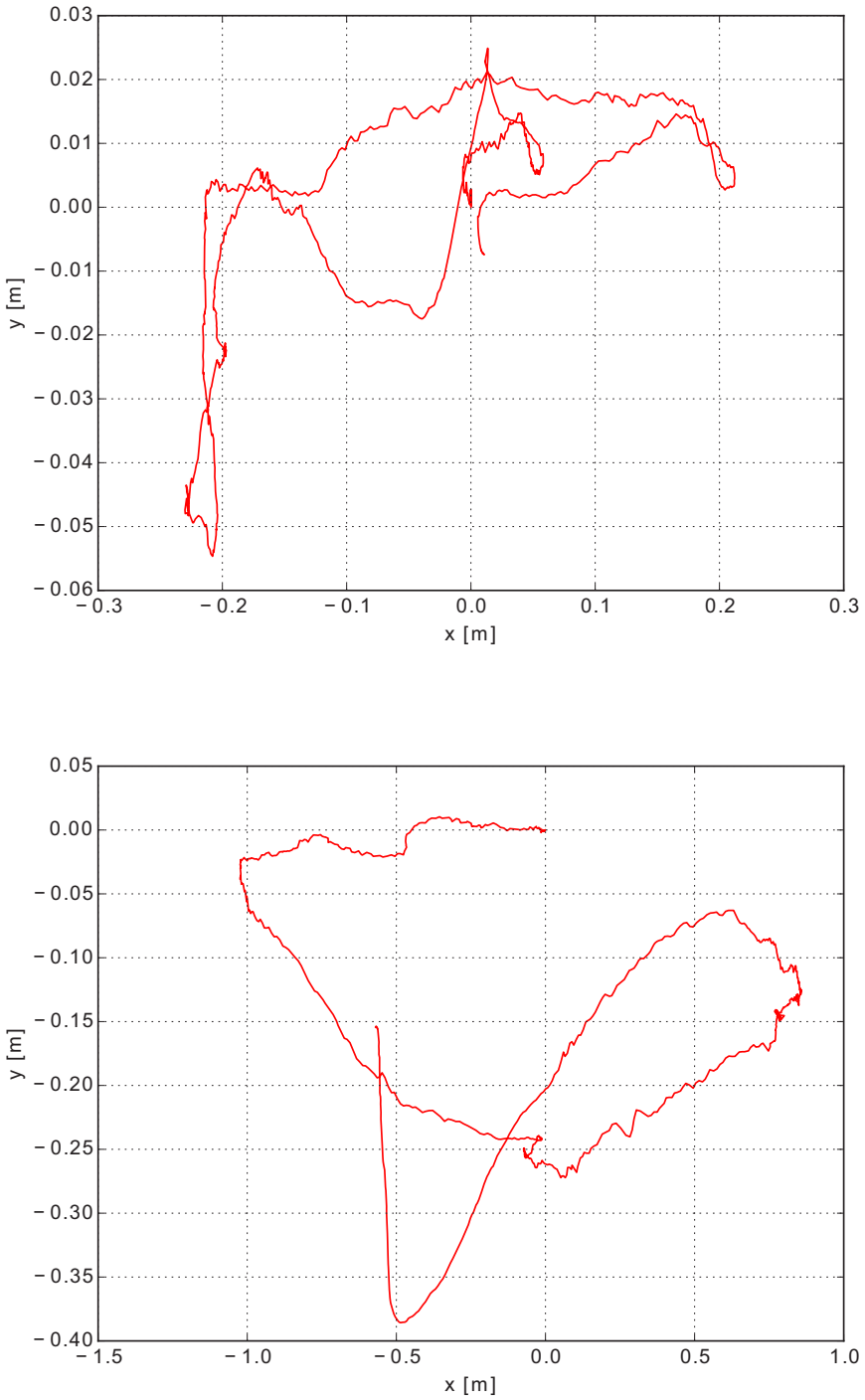


FIGURE 4.5 – Les trajectoires vérité terrain des séquences kt1 et kt2.

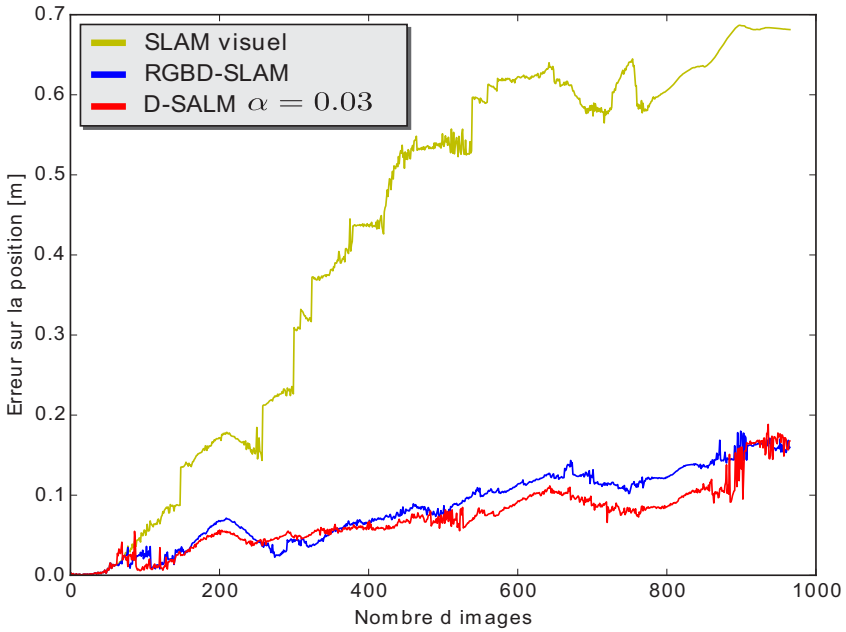
FIGURE 4.6 – **Exemples de données bruitées sur la séquence de Handa et al. (2014)**. En haut les images couleurs et en bas, leurs cartes de profondeur associées.

de profondeur soient bruitées. Comme précédemment, ces résultats montrent que l'intégration de la profondeur dans le SLAM améliore considérablement la précision de la localisation. Pour le D-SLAM, nous constatons que la valeur du facteur de pondération $\alpha = 0.003$ appliquée sur la séquence "Bureaux" n'est pas adaptée sur cette séquence. En effet sur la trajectoire kt2 la RMSE obtenue avec le D-SLAM est le double de celle obtenue avec le RGBD-SLAM et sur la trajectoire kt1 l'algorithm enregistre un échec.
Ces résultats s'expliquent par le fait que les données de profondeur sur les séquences kt1 et kt2 soient bruitées contrairement à la séquence " Bureaux". Comme indiqué dans Scherer et al. (2012), ce facteur de pondération α dépend de l'incertitude que l'on accorde à la profondeur (section 5.3.2). Dans le cas des séquences kt1 et kt2, il doit donc être augmenté. Nous avons appliqué une valeur $\alpha = 0.03$. Les résultats obtenus pour cette nouvelle valeur sont satisfaisants et sont légèrement meilleurs que les résultats atteints par le SLAM-RGBD.

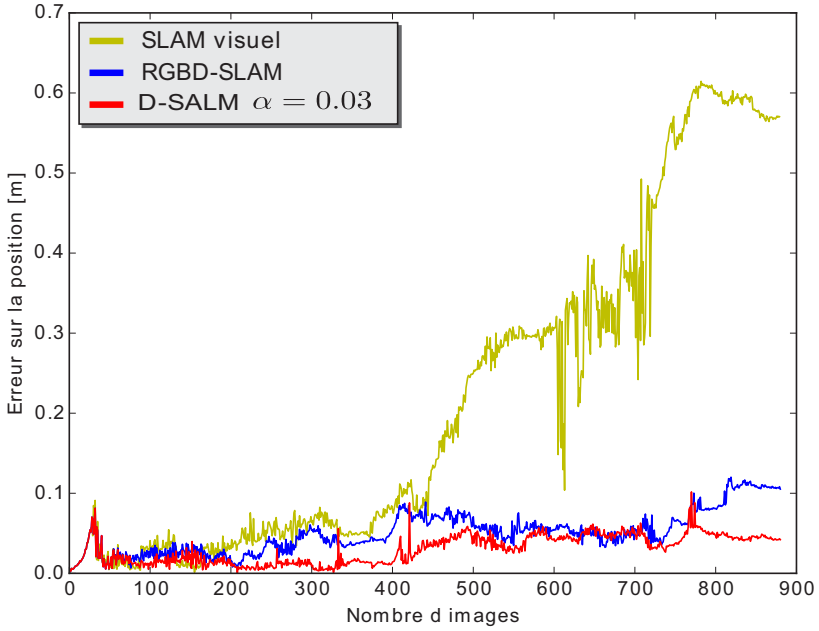
Cependant, si nous analysons les courbes de la Figure 4.7., nous observons, pour les trois algorithmes testés et sur les deux séquences, des pics d'erreurs à certains endroits de la séquence. Ceci est la conséquence des mouvements brusques de la caméra à ces endroits précis où la caméra effectue un mouvement de rotation très important. En effet les mouvements brusques de la caméra augmentent les mauvaises associations 2D/3D, ce qui impacte négativement la précision de la localisation et la convergence de l'ajustement de faisceaux. Néanmoins nous constatons que les algorithmes intégrant la mesure de profondeur (le D-SLAM et le RGBD-SLAM) sont plus robustes aux mouvements brusques par rapport au SLAM visuel. Ce dernier continue de dériver après le premier mouvement difficile jusqu'à atteindre une erreur moyenne d'environ $25cm$ sur les deux trajectoires, tandis que les deux autres méthodes arrivent à se stabiliser autour d'une erreur moyenne ne dépassant pas $4cm$.

Méthodes	Erreurs [m]	kt1	kt2
RGBD-SLAM	RMSE	0.041	0.037
	STD	**0.018**	0.017
D-SLAM	RMSE	**0.038**	**0.022**
$\alpha = 0.03$	STD	0.023	**0.009**
D-SLAM	RMSE	-	0.060
$\alpha = 0.003$	STD	-	0.038
SLAM-Visuel	RMSE	0.251	0.235
	STD	0.085	0.101

TABLE 4.3 – Les différentes erreurs obtenues sur les séquences "Living room".



$kt1$



$kt2$

FIGURE 4.7 – Évolution du RMSE sur la scène de synthèse "Living room" pour les deux trajectoires kt1 et kt2.

4.3.2 Séquences réelles

4.3.2.1 Séquences réelles "Freiburg"

Ces séquences sont issues de la base de données du benchmark de l'université de TUM [1], réalisées par Sturm et al. (2012). Elles contiennent des images couleurs et des images de profondeur ainsi qu'un fichier des poses réelles des caméras. Ces données ont été acquises avec un capteur *Kinect*V1 ou *Asus Xtion*, à 30Hz. Les images ont une résolution de (640×480). Ces séquences ont été enregistrées dans différents environnements (bureaux, hangars) comme le montre la Figure 4.8.

Pour obtenir la trajectoire réelle de la caméra, les auteurs utilisent un système de capture de mouvement externe, constitué de huit caméras de type Raptor-E [2]. Ces caméras fonctionnent à une fréquence de 300Hz et ont une résolution de (1280×1024) pixels. Ce système calcule la pose de la caméra en suivant la position 3D de marqueurs passifs posés sur le capteur *Kinect*.

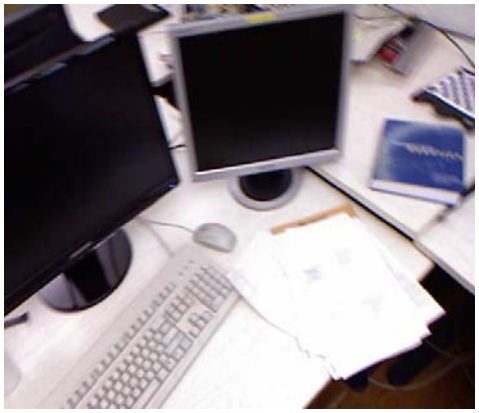
Nous évaluons nos algorithmes en terme de précision de la localisation sur sept séquences. Nous calculons pour chaque séquence, l'erreur de position absolue (ATE) ainsi que l'erreur relative en translation et en rotation (RPE), telles que proposées par Sturm et al. (2012) et rappelées en section 4.1.
Pour les séquences "fr1" et "fr2" de la base de données, les auteurs utilisent le capteur *Kinect*. Les trajectoires pour les séquences "fr1" sont enregistrées dans un environnement de bureaux de taille $(6 \times 6m^2)$. Les séquences "fr2" sont enregistrées dans un grand garage industriel de taille $(10 \times 12m^2)$. Pour les séquences "fr3" les auteurs utilisent le capteur *Asus Xtion*. Dans ce qui suit, nous présentons une brève description de chaque séquence :

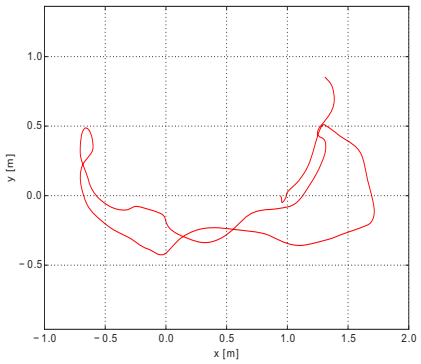
- fr1/desk : séquence de 23 secondes sur 9 mètres de longueur contenant plusieurs passages autour d'un bureau.
- fr1/xyz : séquence de 30 secondes sur 7 mètres. Le capteur regarde un bureau dans un environnement de travail typique et la trajectoire contient principalement des translations sur les axes principaux du capteur.
- fr1/rpy : séquence de 27 secondes sur 1 mètre de longueur où le capteur effectue essentiellement des mouvement de rotation autours des axes principaux.
- fr1/plant : séquence de 41 secondes sur 14 mètres autour d'un pot sur un bureau.
- fr2/desk : séquence de 99 secondes sur 19 mètres autour d'un bureau encombré d'objets dans un grand environnement intérieur.
- fr2/xyz : séquence de 122 secondes sur 7 mètres de longueur autour d'un bureau. Les mouvements enregistrés sont principalement des translations sur les axes principaux.
- fr3/long : séquence de 87 secondes sur une trajectoire de 21 mètres. Le capteur fait un tour complet (360 degrés) autour d'un grand bureau à deux faces dans un grand environnement intérieur.

Résultats : Pour montrer l'efficacité de l'algorithme proposé, plusieurs erreurs sont estimées. Nous présentons d'abord les résultats du SLAM et du RGBD-SLAM en terme d'erreurs absolues dans la tableau 4.4. La Figure 4.9 montre la différence entre la trajectoire réelle et la trajectoire estimée par le RGBD-SLAM, projetée sur le plan. Nous présentons par la suite les erreurs relatives en position dans le tableau 4.5 et les erreurs relatives en rotation dans le tableau 4.6. Enfin, nous évaluons également l'algorithme D-SLAM sur ces séquences pour une valeur de $\alpha = 0.03$. Les résultats sont présentés dans le tableau 4.4.

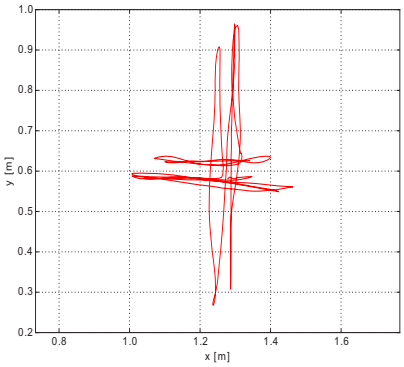
1. http ://vision.in.tum.de/data/datasets/rgbd-dataset
2. http ://www.motionanalysis.com/html/industrial/raptore.html

fr1/desk

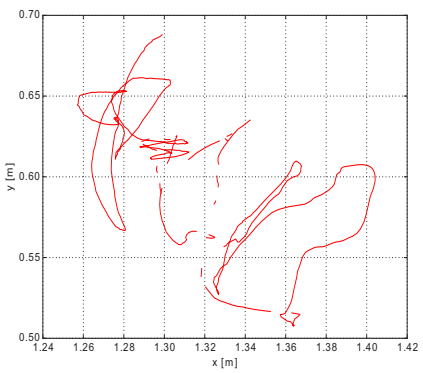




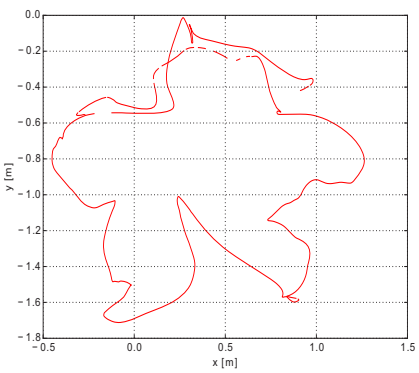
fr1/xyz



fr1/rpy



fr1/plant



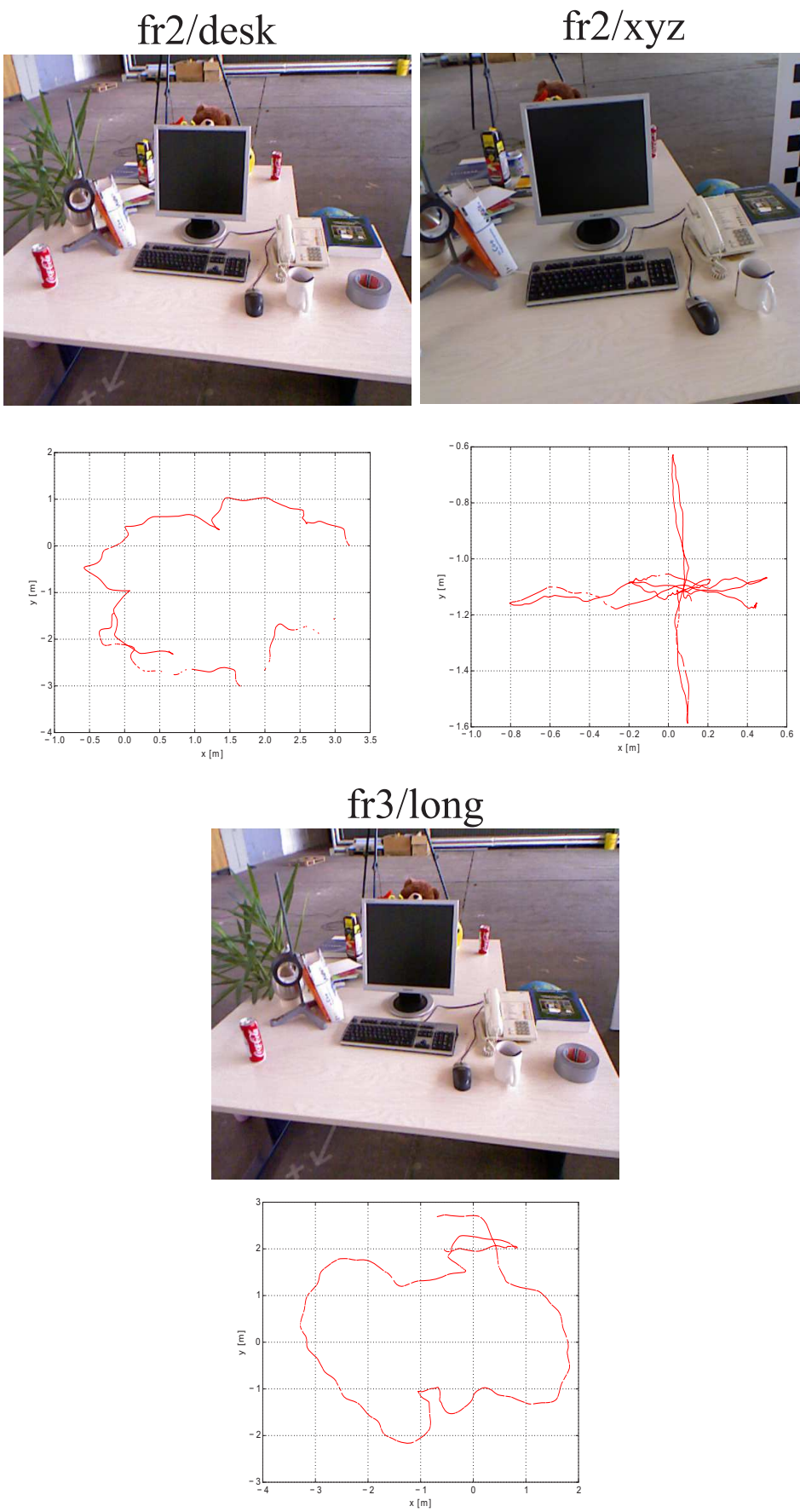


FIGURE 4.8 – **Séquences réelles "Freiburg"**. Pour chacune des séquences, une image couleur et la trajectoire réelle de la caméra, sont présentées.

Sur l'ensemble des trajectoires, les résultats obtenus montrent qu'intégrer la mesure de pro-

Séquences	algorithmes	les erreurs absolues (m)		
		RMSE	STD	MAX
fr1/desk	SLAM	0.631	0.201	1.012
	D-SLAM	**0.058**	**0.019**	**0.082**
	RGBD-SLAM	0.062	0.024	0.156
fr1/xyz	SLAM	0.183	0.061	0.316
	D-SLAM	0.036	**0.008**	**0.097**
	RGBD-SLAM	**0.028**	0.091	0.133
fr1/rpy	SLAM	-	-	-
	D-SLAM	0.068	**0.013**	**0.127**
	RGBD-SLAM	**0.056**	0.025	0.159
fr1/plant	SLAM	-	-	-
	D-SLAM	**0.073**	0.032	0.233
	RGBD-SLAM	0.075	**0.027**	**0.174**
fr2/desk	SLAM	-	-	-
	D-SLAM	-	-	-
	RGBD-SLAM	-	-	-
fr2/xyz	SLAM	0.044	0.029	0.128
	D-SLAM	0.035	0.021	0.159
	RGBD-SLAM	**0.021**	**0.015**	**0.111**
fr3/long	SLAM	0.654	0.302	1.135
	D-SLAM	0.058	0.033	0.176
	RGBD-SLAM	**0.034**	**0.019**	**0.175**

TABLE 4.4 – **Résultats des trois algorithmes SLAM, D-SLAM et RGBD-SLAM sur les séquences réelles "Freiburg".** Les erreurs de translation absolues (ATE).

fondeur dans le SLAM visuel permet d'améliorer la précision de sa localisation : la précision du RGBD-SLAM est 2 à 20 fois meilleure que celle du SLAM visuel classique. D'autre part, notons que pour 3 séquences, le SLAM Visuel a échoué.

Pour des mouvements de translation non brusques comme dans le cas de la séquence fr2/xyz, les différentes erreurs enregistrées par le SLAM visuel sont quasiment le double de celles enregistrées par le RGBD-SLAM et pour des mouvements de translations saccadés (cas de la séquence fr1/xyz), l'erreur produite par le SLAM visuel est jusqu'à dix fois plus importante que l'erreur produite par le RGBD-SLAM, ce qui souligne les apports de la mesure de profondeur pour la robustesse face aux mouvements brusques.

Pour la séquence où le mouvement dominant est la rotation (fr1/rpy), le SLAM visuel a enregistré un échec, alors que le RGBD-SLAM a enregistré une RMSE de 5cm pour l'ATE et une RMSE de 7cm pour la RPE, ce qui prouve que le RGBD-SLAM est plus robuste aux mouvements de rotation. En effet, dans le cas du SLAM visuel les points sont construits par triangulation. Dans le cas de rotations pures l'estimation de ces points par triangulation est erronée (à l'infini). Ces points sont alors rejetés et considérés comme des points aberrants ce qui provoque l'échec de l'algorithme. Reconstruire les points par rétro-projection -grâce à leur mesure de profondeur- permet de palier en partie ce problème. Même si les points reconstruits

Séquences	algorithmes	Les erreurs relatives (m)		
		RMSE	STD	MAX
fr1/desk	SLAM	0.507	0.185	1.109
	RGBD-SLAM	**0.067**	**0.029**	**0.150**
fr1/xyz	SLAM	0.199	0.131	0.500
	RGBD-SLAM	**0.018**	**0.009**	**0.166**
fr1/rpy	SLAM	-	-	-
	RGBD-SLAM	**0.079**	**0.034**	**0.176**
fr1/plant	SLAM	-	-	-
	RGBD-SLAM	**0.077**	**0.034**	**0.201**
fr2/desk	SLAM	-	-	-
	RGBD-SLAM	-	-	-
fr2/xyz	SLAM	0.046	0.024	0.158
	RGBD-SLAM	**0.030**	**0.014**	**0.096**
fr3/long	SLAM	0.087	0.055	0.251
	RGBD-SLAM	**0.028**	**0.016**	**0.181**

TABLE 4.5 – **Évaluation de la dérive du RGBD-SLAM** : Les erreurs de translation relatives (RPE) obtenues sur les différentes séquences "Freiburg".

Séquences	algorithmes	les erreurs relatives (ř)		
		RMSE	STD	MAX
fr1/desk	SLAM	30.622	12.353	51.090
	RGBD-SLAM	**3.41**	**1.670**	**8.203**
fr1/xyz	SLAM	9.369	6.608	26.537
	RGBD-SLAM	**1.765**	**0.873**	**3.653**
fr1/rpy	SLAM	-	-	-
	RGBD-SLAM	**5.429**	**2.289**	**11.039**
fr1/plant	SLAM	-	-	-
	RGBD-SLAM	**1.881**	**0.830**	**5.197**
fr2/desk	SLAM	-	-	-
	RGBD-SLAM	-	-	-
fr2/xyz	SLAM	1.985	1.203	7.692
	RGBD-SLAM	**0.782**	**0.414**	**5.213**
fr3/long	SLAM	3.899	3.175	16.325
	RGBD-SLAM	**1.237**	**0.699**	**7.895**

TABLE 4.6 – **Évaluation de la dérive du RGBD-SLAM** : Les erreurs de rotation relatives (RPE) obtenues sur les différentes séquences "Freiburg".

par rétro-projection sont peu précis, ils seront ensuite optimisés dans l'ajustement de faisceaux que nous proposons.

Sur la séquence fr2/desk, les trois algorithmes ont échoué en raison du saut important présent dans cette séquence (voir Figure 4.9). Notre algorithme repose sur la mise en correspondance des points sur des images successives, pour estimer la pose de la caméra, de ce fait, il doit y avoir un recouvrement suffisant pour assurer le fonctionnement de l'algorithme.

Enfin pour la séquence fr1/plant, le RGBD-SLAM a enregistré la RMSE la plus élevée sur l'ATE et la RPE ; environ $7cm$ sur une trajectoire de 14m. Cette séquence présente beaucoup de flous et de mouvements rapides. Elle met en évidence la sensibilité de notre méthode vis à vis des erreurs ou des manques d'associations de points 2D. La précision de la localisation est corrélée aux mouvements présents sur la séquence. Un mouvement rapide engendre un flou de bougé, augmente l'effet du rolling-shutter du capteur RGB-D et entraine une mauvaise association des points 2D, ce qui impacte négativement la localisation et l'ajustement de faisceaux. Un moyen d'y remédier est une approche pyramidale. Celle ci a été validée au laboratoire sur le SLAM visuel mais n'a pas encore été intégrée dans le RGBD SLAM. C'est la raison pour laquelle nous ne présentons pas de résultats en pyramidal sur le RGBD-SLAM.

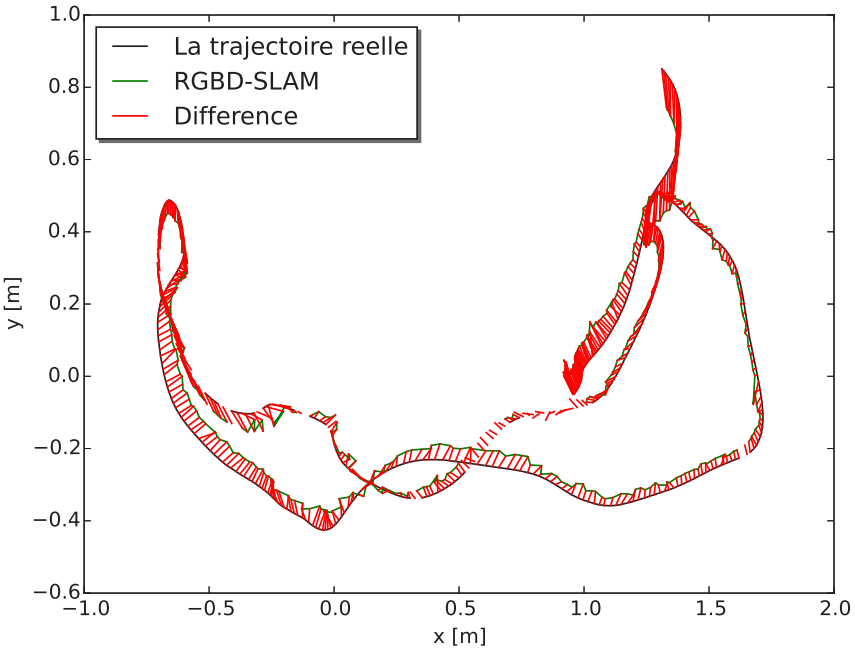
4.3.2.2 Comparaison avec les méthodes de l'état de l'art

Nous comparons notre algorithme avec cinq méthodes RGBD-SLAM de l'état de l'art : DVO (Kerl et al. (2013a)), RGBD (Endres et al. (2012)), Kintinuous (Whelan et al. (2015)) et le Unified keyframe dense SLAM (Meilland and Comport (2013)), dSLAM (Scherer et al. (2012)). Cette étude comparative est réalisée sur les séquences réelles présentées dans la section 4.3.2.
Pour le DVO nous relevons les résultats publiés par (Whelan et al. (2015)), notés par la suite DVO(1) et nous testons le code source publié dans ([3]), notés par la suite DVO(2). Les détails de ces méthodes sont présentés dans le chapitre 2.
Le tableau 4.7 résume les erreurs obtenues par chaque méthode. Notons que les résultats présentés dans ce tableaux, pour les méthodes de l'état de l'art, sont issus des publications de leurs auteurs respectifs.

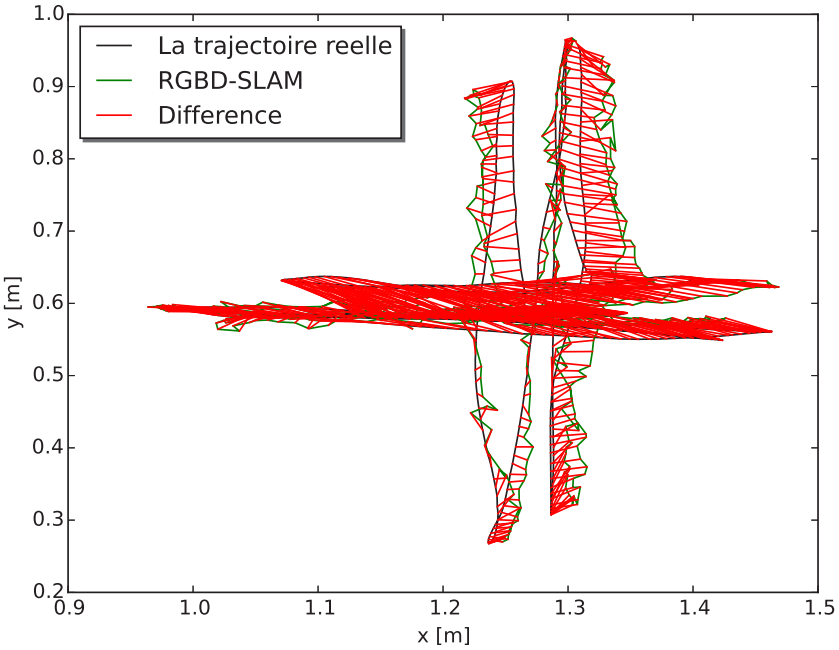
Séquences	Notre méthode	DVO(1)	DVO (2)	RGBD	Kintinuous	Unified	dSLAM
fr1/desk	0.062	0.021	0.229	0.023	0.037	**0.018**	-
fr1/xyz	0.028	**0.011**	0.029	0.014	0.017	-	-
fr1/rpy	0.056	**0.020**	0.072	0.026	0.028	-	-
fr1/plant	0.073	**0.028**	0.097	0.091	0.041	-	-
fr2/desk	-	**0.017**	0.136	0.057	0.034	0.093	-
fr2/xyz	0.021	0.018	0.023	**0.008**	0.029	-	-
fr3/long	0.034	0.035	0.222	**0.017**	0.031	-	0.136

TABLE 4.7 – Comparaison du RMSE de l'ATE, obtenue par chacune des méthodes RGBD-SLAM.

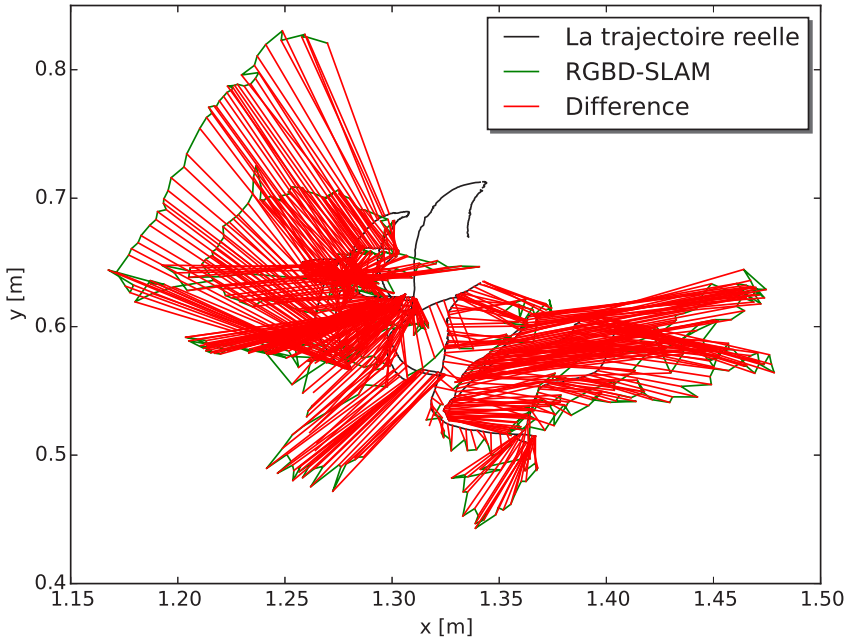
3. https ://vision.in.tum.de/data/software/dvo



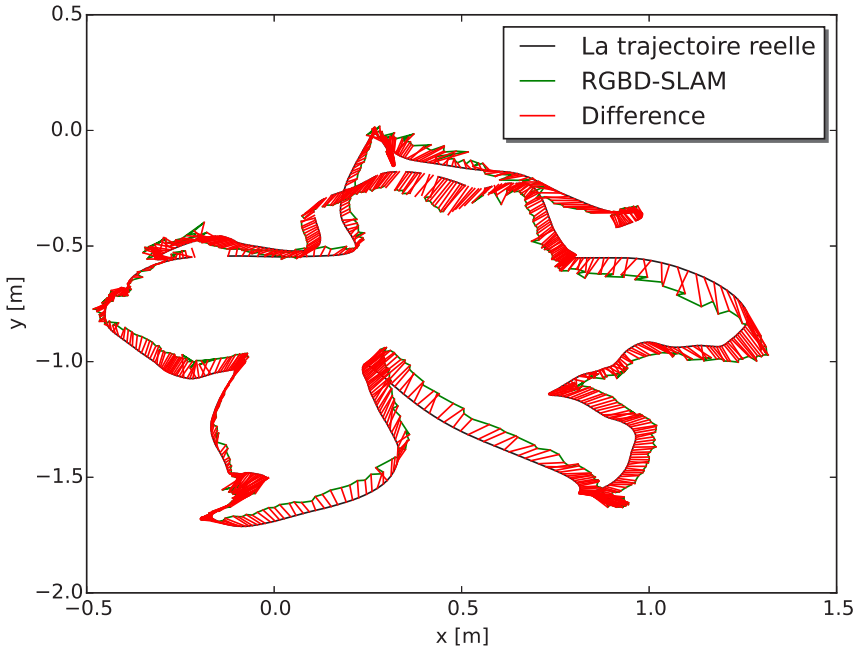
(a) fr1/desk



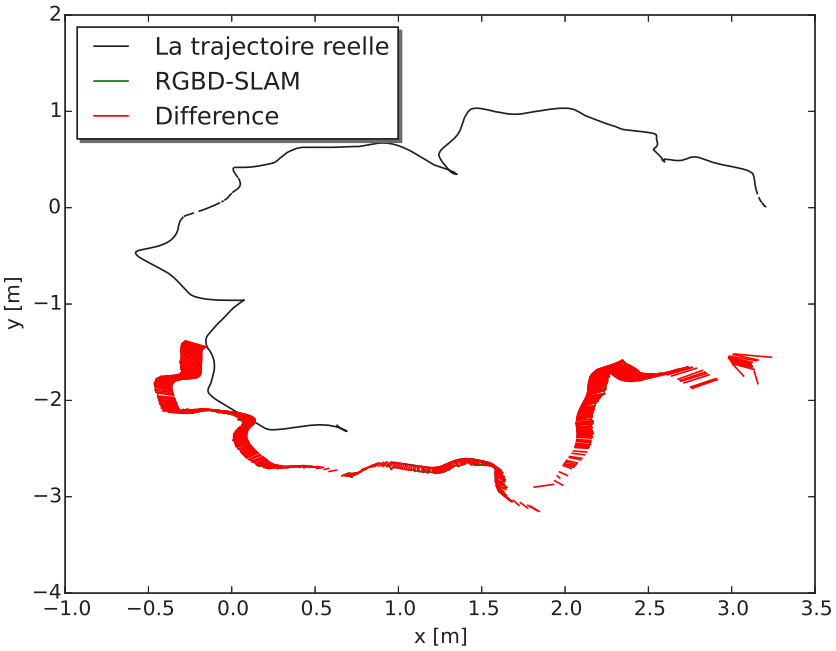
(b) fr1/xyz



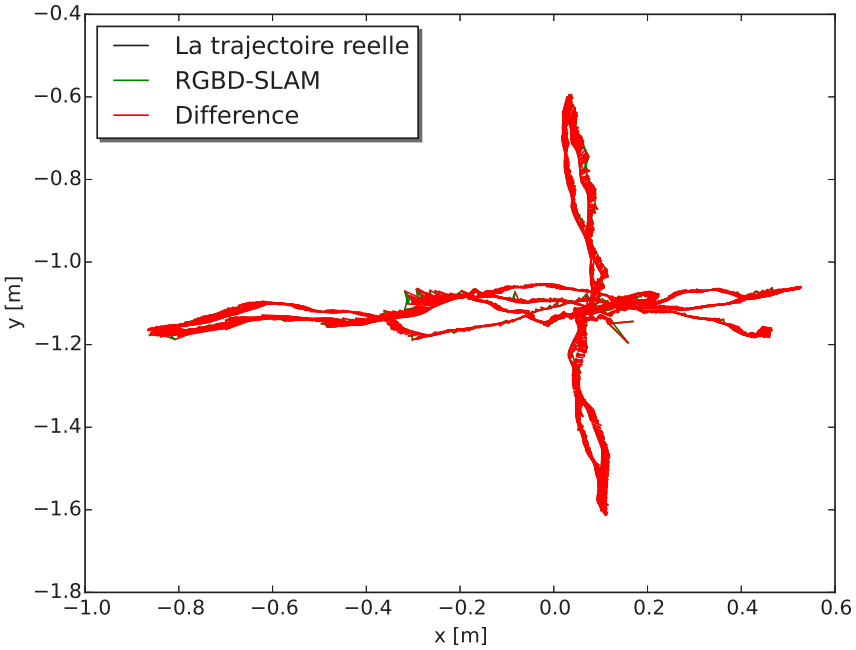
(c) fr1/rpy



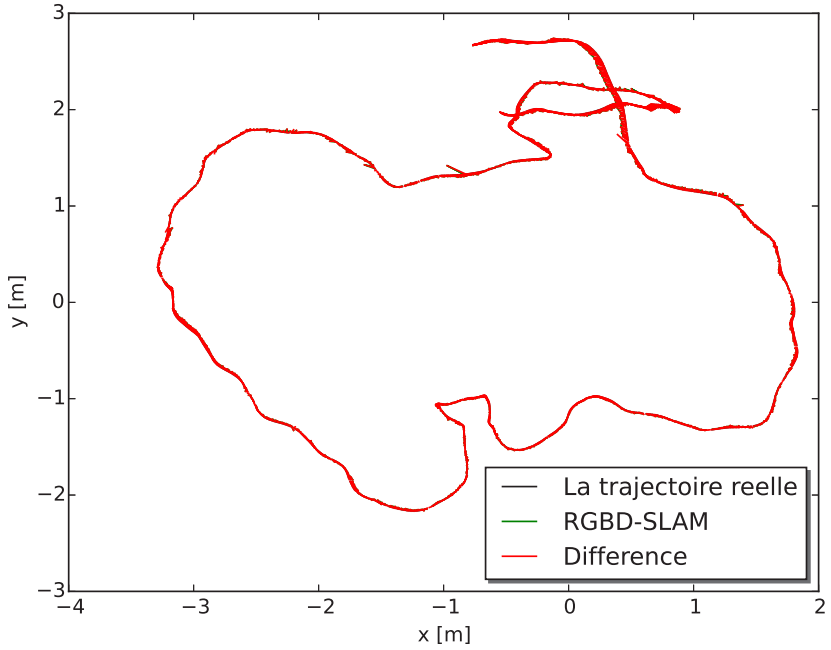
(d) fr1/plant



(e) fr2/desk



(f) fr2/xyz



(g) fr3/long

FIGURE 4.9 – **Séquences réelles "Freiburg".** Comparaison de la trajectoire estimée par rapport à la trajectoire réelle sur la plan (xy).

Nous constatons que notre méthode n'enregistre pas les meilleurs résultats sur toutes les séquences. Néanmoins l'erreur obtenue par notre méthode ne dépasse jamais $8cm$.
Les méthodes DVO(1), RGBD et Kintinous sont des méthodes denses, elles utilisent tous les points de l'image. La méthode Unified, est une approche semi dense qui utilise les points les plus saillants de l'image. L'exploitation du maximum de points contribue à une meilleure précision. Cependant ces méthodes denses ou semi denses sont couteuses en temps de calcul. Elle nécessitent l'exploitation d'un processeur graphique GPU pour atteindre des performances temps réel. Notre objectif est de proposer un algorithme fonctionnant uniquement sur CPU en vue d'une utilisation sur une structure légère à forte autonomie de type (casque, lunette, drone). De ce fait, il est intéressant de comparer notre méthode avec la méthode DVO(2) et la méthode dSLAM, les seules, dans le tableau 4.7 qui fonctionnent sur CPU. Les résultats du tableau 4.7 montrent que notre méthode est plus précise que DVO(2) sur toutes les séquences testées. Elle est également plus précise que dSLAM sur la seule séquence disponible pour la comparaison (fr3/long).

Notons cependant que depuis notre publication (Melbouci et al. (2015)), Mur-Artal and Tardos (2016) ont publié de nouveaux travaux appelés ORB-SLAM2 fonctionnant sur CPU et avec de meilleurs résultats. Les auteurs proposent une approche similaire à la nôtre : un SLAM visuel (ORB-SLAM dans leur cas), revisité dans lequel ils intègrent dans la fonction de coût de l'ajustement de faisceaux, une contrainte supplémentaire liée aux profondeurs des points. Dans ces travaux, cette contrainte est intégrée par des appariements stéréos obtenus à partir de la carte de profondeur, ce qui leur permet d'avoir une fonction de coût générique pouvant fonctionner avec un capteur stéréo ou un capteur RGBD.

4.3.2.3 Séquences réelles "Corbs"

La seconde série de séquences réelles est tirée du benchmark réalisé par Wasenmüller et al. (2016). Cette base de données récemment publiée contient des images couleurs et des images de profondeur de quatre scènes différentes (Figure 4.10.) acquises avec le capteur *KinectV2*. Elle contient la trajectoire réelle des caméras et une reconstruction 3D réelle de la scène. Les trajectoires réelles dans ces séquences ont été générées par un système de capture de mouvement externe, elles sont d'une précision de l'ordre du sous-millimètre. Ces trajectoires sont présentées dans la Figure 4.11.

Résultats : Les résultats obtenus sur les séquences "Corbs" sont présentés dans les Figures 4.12, 4.13. et le tableau 4.8.

Les résultats confirment toujours que rajouter l'information de profondeur comme contrainte

Séquences	algorithmes	les erreurs absolues (m)		
		RMSE	STD	MAX
Human	SLAM visuel	0.583	0.172	1.150
	RGBD-SLAM	**0.114**	**0.055**	**0.468**
Desk	SLAM visuel	0.165	0.041	0.382
	RGBD-SLAM	**0.016**	**0.006**	**0.074**
Electrical	SLAM visuel	-	-	-
	RGBD-SLAM	**0.065**	**0.025**	**0.141**

TABLE 4.8 – Les erreurs de translation absolues (ATE) obtenues sur les séquences "Corbs"

supplémentaire dans l'ajustement de faisceaux du processus de SLAM visuel améliore significativement sa précision. Cependant sur la séquence "Desk", l'erreur de position est réduite d'un facteur 10, mais sur la séquence "Human" l'erreur de position est réduite d'un facteur 5. Ceci peut s'expliquer par le fait que la séquence "Desk" soit plus texturée que la séquence "Human". Notre RGBD-SLAM, par sa caractéristique éparse peut souffrir du manque de texture à certains endroits de la scène.

Si nous analysons l'évolution de l'erreur (ATE) sur la séquence "Human", on constate des pics d'erreurs entre les images (1190 et 1194). A ces endroits précis, le flou de bougé sur les images est important comme le montre l'exemple illustré dans la Figure 4.14. ; ce qui explique la difficulté du RGBD-SLAM à se localiser à ces endroits.

Sur la séquence "Electrical", le SLAM visuel a enregistré un échec à l'image 741. Á cet endroit de la séquence nous avons constaté un flou de bougé combiné à un mouvement de rotation important, par conséquent le nombre de correspondances nécessaires pour le calcul de pose et le nombre de points 3D cohérents n'est pas suffisant, ce provoque l'échec du SLAM visuel. Le RGBD-SLAM pour sa part n'a pas échoué, ce qui prouve sa robustesse face aux mouvements difficiles, par rapport au SLAM visuel. Néanmoins il enregistre plusieurs pics d'erreurs. Le maximum de l'erreur est enregistré à l'image 454 (Figure 4.15.) L'erreur (ATE) est de 0.14m. Á cette endroit le saut entre l'image 453 et l'image 454 est brusque et conséquent. La mise en correspondance des points devient difficile, ce qui affecte négativement le calcul de pose et la convergence de l'ajustement de faisceaux.

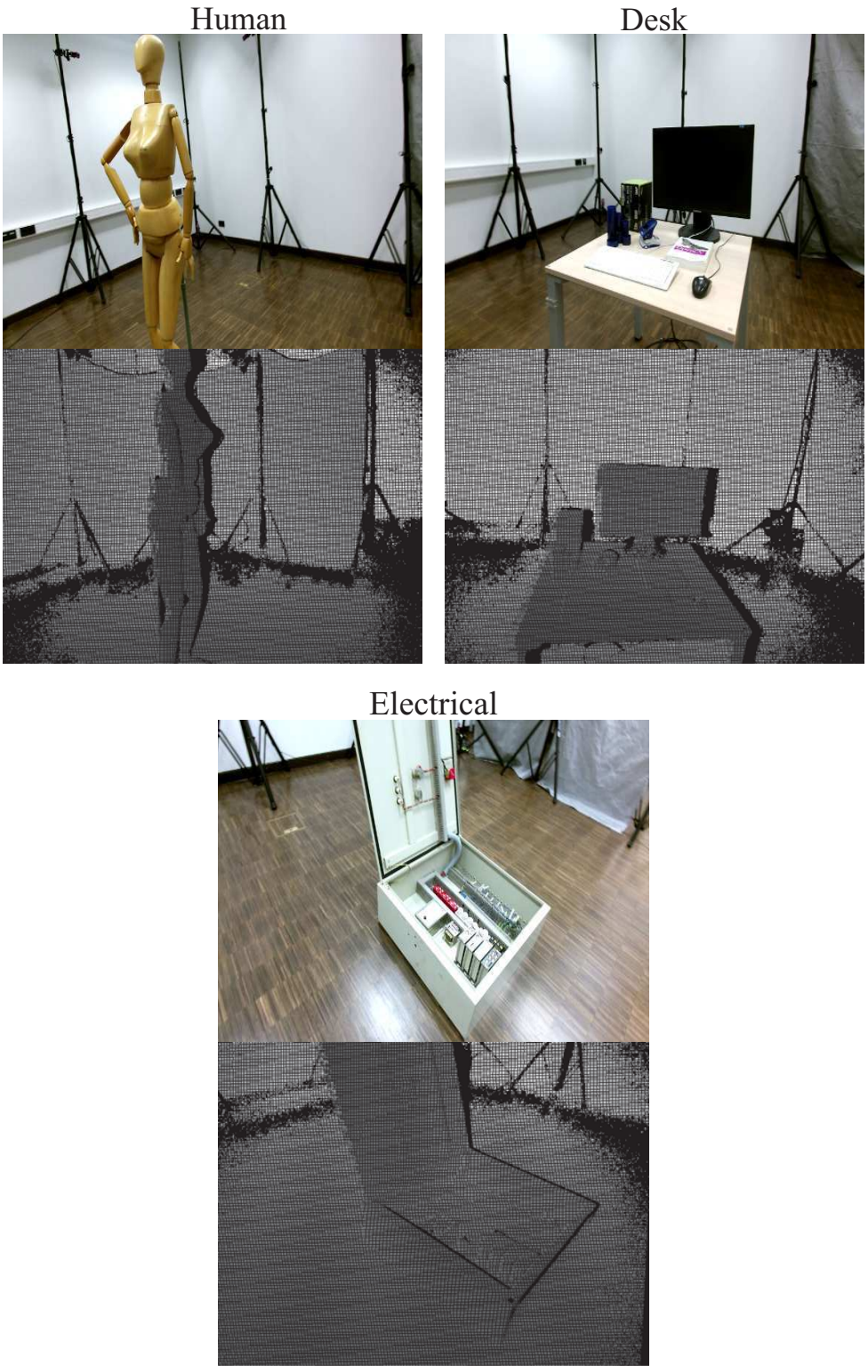
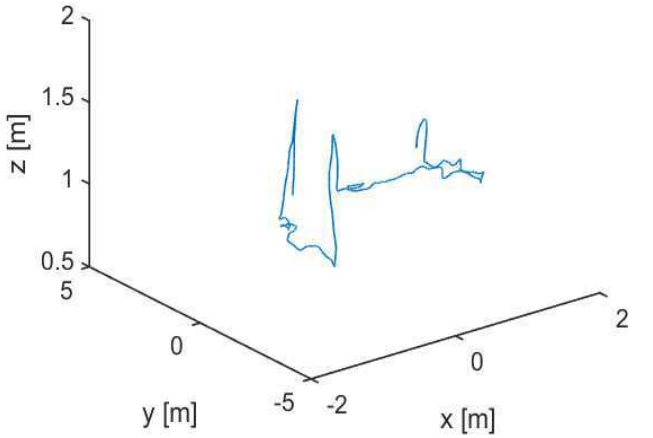
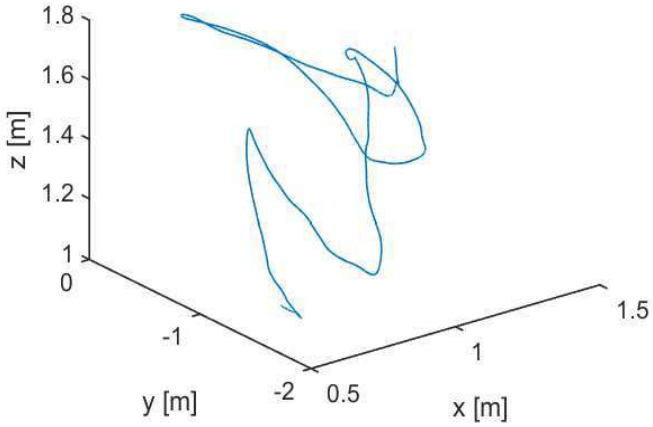


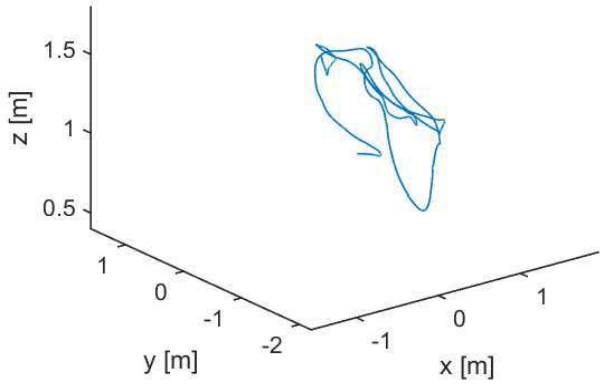
FIGURE 4.10 – **Séquences réelles "Corbs" :** Exemples d'images couleurs des scènes observées et leurs images de profondeur associées.



Human

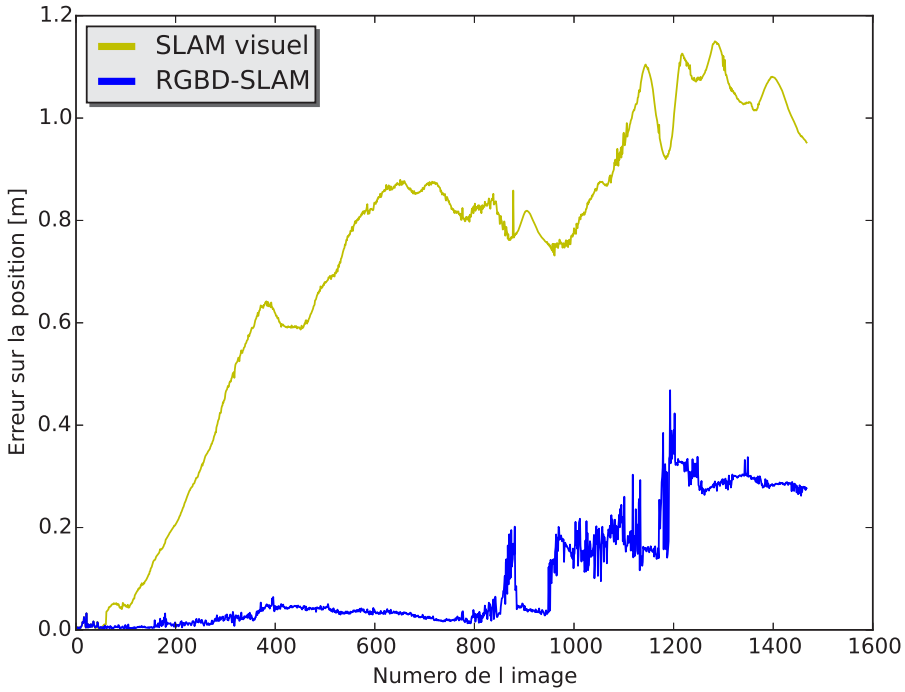


Desk

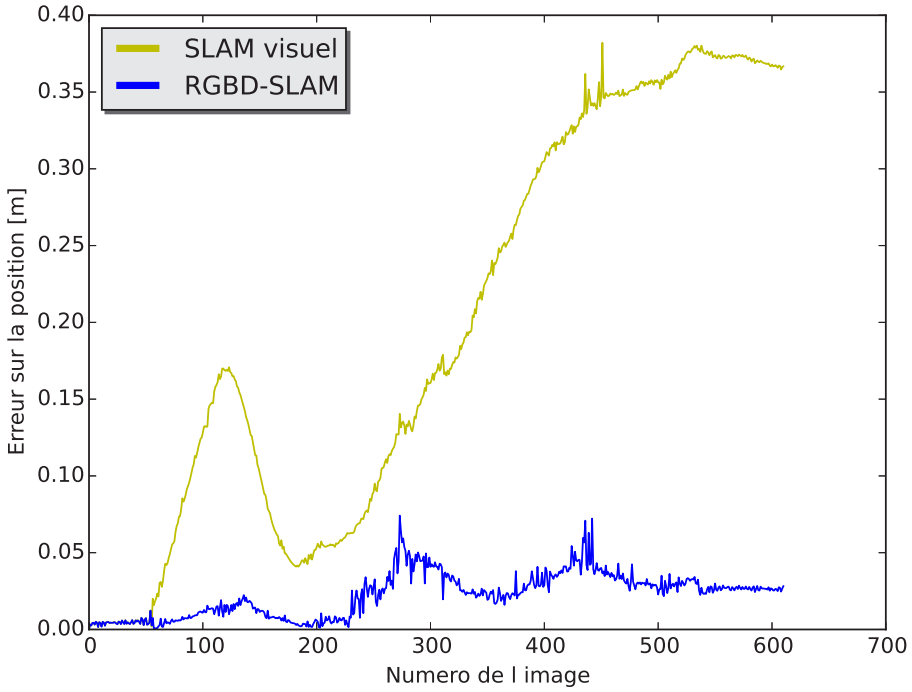


Electrical

FIGURE 4.11 – **Séquences réelles "Corbs".** Les trajectoires réelles des caméras.



(a) Human



(b) Desk

FIGURE 4.12 – Évolution des erreurs (ATE) sur les séquences "Human" et "Desk".

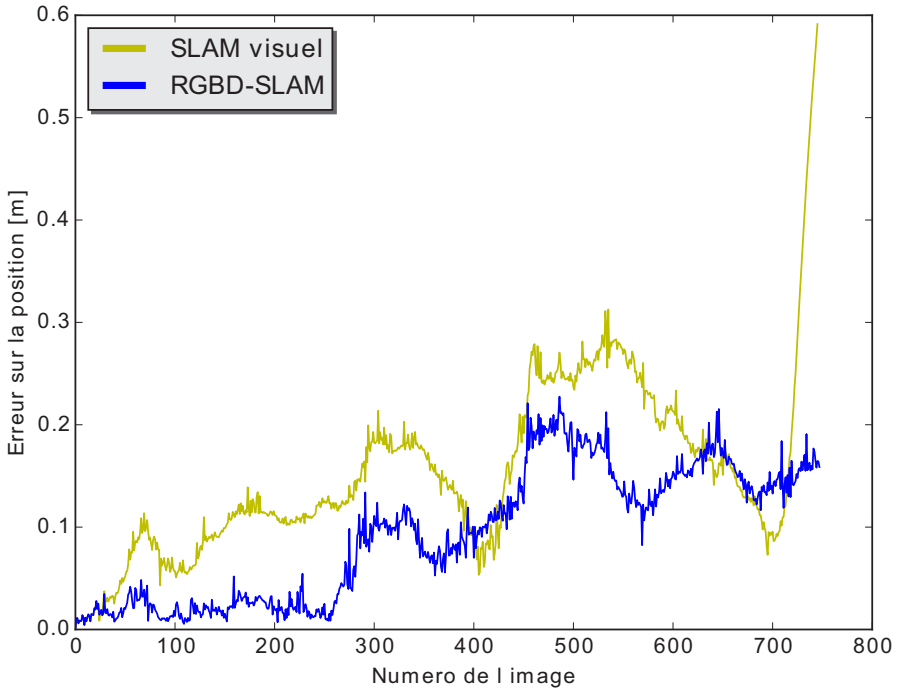


FIGURE 4.13 – Évolution des erreurs (ATE) sur la séquence "Electrical".

FIGURE 4.14 – **Séquence réelle "Human"** : Exemple d'image couleur ou le flou de bougé est important (image 1193).

(a) Image 453

(b) Image 454

FIGURE 4.15 – **Séquence réelle "Electrical"** : Exemple de mouvement brusque entre l'image 453 et l'image 454.

Sur la séquence "Electrical", l'amélioration apportée par le RGBD-SLAM en terme de précision de la localisation n'est pas aussi importante comparée aux améliorations enregistrées sur les deux autres séquences. La raison est que les mouvements de la caméra dans la séquence "Electrical" sont plus difficiles (plus de mouvement brusques, plus de rotations) que les deux autres séquences ("Human" et "Desk") comme le montre la Figure 4.11.

4.3.3 Conclusion

Pour valider la qualité de la localisation du RGBD-SLAM, nous avons procédé à plusieurs expérimentations. Cet algorithme a été évalué sur un ensemble de séquences de synthèses et de séquences réelles. Cette évaluation montre que l'exploitation de l'information de profondeur réduit la dérive en facteur d'échelle du SLAM visuel et améliore sa précision. Cette nouvelle fonction de coût permet de garder la structure creuse des matrices engendrées par l'ajustement de faisceaux et par conséquent d'obtenir des temps de traitement encourageants (25ms en moyenne sur un Intel Xeon W3570 à 3.20 GHz en utilisant 1 coeur).

Cependant, le RGBD-SLAM reste sensible aux mouvements brusques, aux rotations importantes et au manque de texture. Dans le chapitre suivant nos travaux porteront sur l'amélioration de la robustesse de ce RGBD-SLAM dans des zones avec peu de texture dans l'environnement. Notons que la solution proposée ne se limite pas qu'aux caméras RGB-D. Elle peut être utilisée avec d'autres types de capteurs qui fournissent l'information de profondeur comme un laser couplé à une caméra.

CHAPITRE 5

RGBD-SLAM Contraint : Contrainte d'appartenance aux plans de la scène

Dans ce chapitre nous proposons un nouvel ajustement de faisceaux basé modèle, pour améliorer la robustesse et la précision du SLAM-RGBD proposé précédemment. Dans ces travaux nous considérons que ce modèle est préalablement connu, complètement ou partiellement.

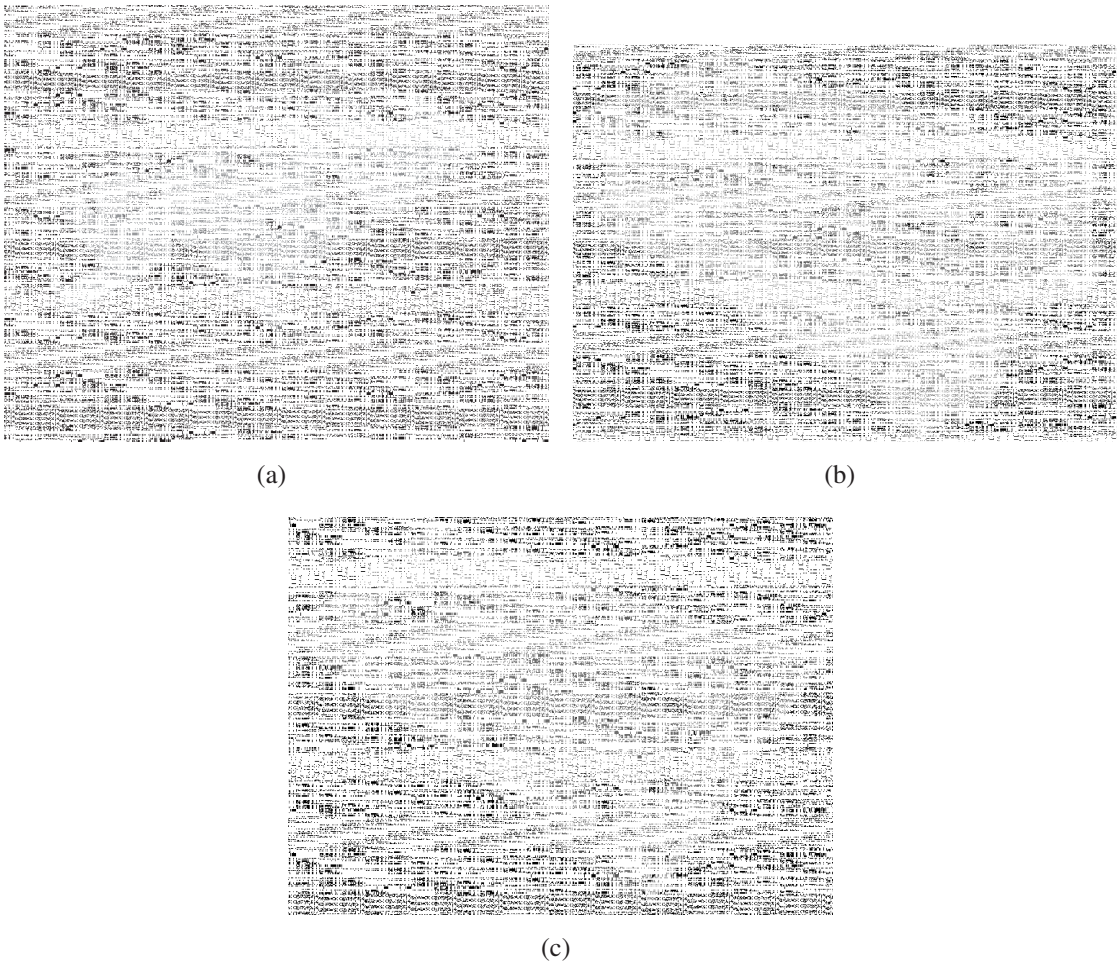
Cette approche est inspirée des travaux de Tamaazousti et al. (2011a). L'idée est de contraindre les points 3D de l'environnement par leur profondeur fournie par un capteur 3D et par les plans de la scène auxquels ils seront associés. Ces deux contraintes sont fusionnées au niveau de l'ajustement de faisceaux pour assurer une meilleure localisation.

Nous commençons ce chapitre par expliquer l'intérêt de combiner, dans l'ajustement de faisceaux, des informations absolues extraites du modèle 3D de la scène avec des informations fournies par un capteur 3D. Nous présentons quelques approches de l'état de l'art, qui utilisent un modèle 3D de la scène pour améliorer la localisation d'un capteur. Nous détaillons par la suite les différentes fonctions de coût qui interviennent dans le processus d'optimisation et leur influence sur la précision et la robustesse de la localisation.
Ces travaux ont fait l'objet d'une publication internationale (Melbouci et al. (2016)).

5.1 Introduction

Intégrer l'information de profondeur dans le SLAM visuel nous a permis d'améliorer la précision de la localisation et par conséquent d'améliorer la reconstruction 3D. Cependant, nous avons vu que le RGBD-SLAM dérive tout de même au cours du temps, notamment sur les longues trajectoires. Ceci est dû aux imprécisions liées au capteur, aux environnements parfois pauvres en structures géométriques ou en textures et au caractère incrémental du SLAM qui a tendance à propager les erreurs.

Pour améliorer la précision, nous proposons d'utiliser une information géométrique absolue issue de la connaissance partielle ou complète du modèle 3D de l'environnement. La Figure 5.1. montre des exemples de modèles 3D, pouvant être exploités par notre algorithme. Cette



(a) (b)

(c)

FIGURE 5.1 – **Exemple de modèles 3D.** (a) : modèle 3D généré par un logiciel de rendu 3D [4], (b) : modèle obtenu par conception assistée par ordinateur (CAO), fourni par le constructeur, (c) : modèle 2.5D reconstruit à partir des plans 2D d'un bâtiment.

information est intégrée comme contrainte additionnelle dans le RGBD-SLAM présenté dans le chapitre 3. Pour que l'impact de cette contrainte soit efficace, sans dégrader les performances calculatoires du SLAM-RGBD, nous l'intégrons directement dans le processus d'optimisation comme le montre la Figure 5.2. Nous commençons ce chapitre par un rapide état de l'art des méthodes de localisation par SLAM RGBD, exploitant un modèle 3D a priori de la scène. Nous détaillerons par la suite notre méthode que nous appellerons *RGBD SLAM Contraint*. Nous terminerons ce chapitre par discuter des apports de cette méthode en vue d'améliorer la localisation du SLAM-RGBD sur les grandes trajectoires et dans des environnements difficiles.

5.2 État de l'art des méthodes utilisant la connaissance d'un modèle 3D

Comme présenté dans le chapitre précédent, de nombreux algorithmes de localisation et de reconstruction 3D au moyen d'un capteur RGBD sont aujourd'hui proposés (Endres et al.

4. https ://www.blender.org/

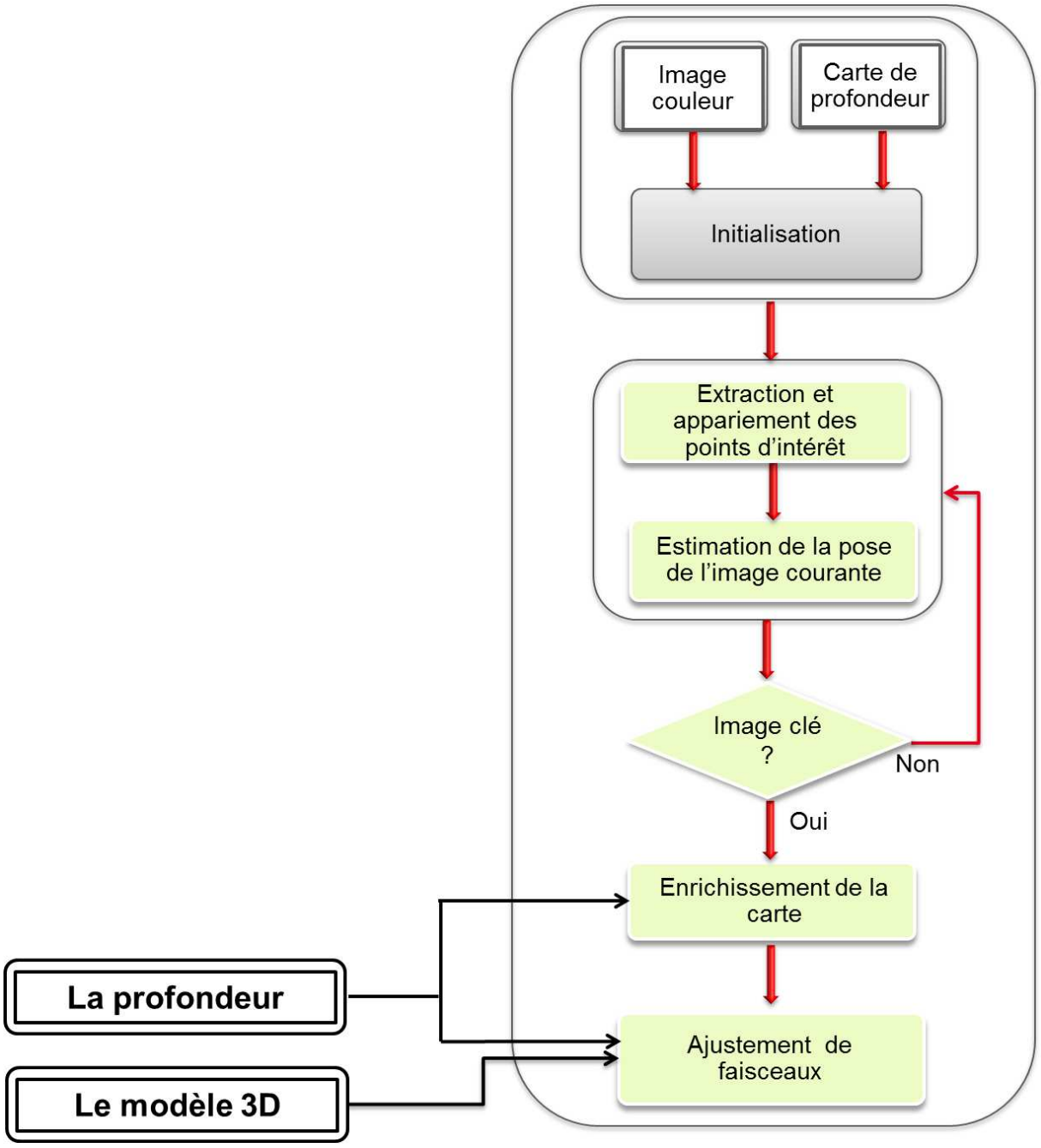


FIGURE 5.2 – **RGBD-SLAM Contraint.** Le SLAM visuel augmenté par la mesure de profondeur et contraint par les plans de l'environnement.

(2012), Kerl et al. (2013a), Whelan et al. (2015), Melbouci et al. (2015)). Ces méthodes reconstruisent une carte d'un environnement inconnu. Cette carte est utile pour la planification de la trajectoire de la caméra et permet de limiter les erreurs cumulées lors de l'estimation de cette trajectoire.
Cependant ces solutions ne sont pas précises sur de longues distances car elles dérivent au cours du temps. Pour limiter la dérive, certaines méthodes proposent des techniques de fermeture de boucles mais elles nécessitent le passage du capteur par un endroit déjà visité, ce qui limite leur utilisation. D'autre part, ces méthodes ne fournissent pas une localisation géo-référencée

par rapport à un modèle (localisation absolue) mais une localisation relative par rapport à la position initiale du capteur. De plus, les capteurs visuels sont sujets aux bruits et sensibles aux conditions d'éclairage.

Une solution possible consiste à localiser la caméra par rapport à un modèle 3D connu. Ces modèles sont exempts de dérive, et présentent peu de données aberrantes. Ils sont indépendants des conditions d'illumination et sont de plus en plus disponibles. Dans certaines applications la carte de l'environnent est préalablement connue. Nous pouvons citer l'exemple d'un robot évoluant dans un bâtiment où la carte est construite manuellement à partir des plans 2D du bâtiment. Ces derniers étant fournis par le constructeur (Winterhalter et al. (2015)).

De nombreux travaux proposent d'exploiter le modèle de l'environnement dans un processus d'ajustement de faisceaux (Larnaout et al. (2013), Geva et al. (2015), Ozog and Eustice (2016)). Mais depuis l'apparition des capteurs RGBD, d'autres méthodes proposent d'exploiter la connaissance complète ou partielle des modèles 3D de l'environnement dans des algorithmes statistiques (Monte Carlo Localisation (MCL), filtre à particules basées vision, Dellaert et al. (1999), Thrun et al. (2001), Oishi et al. (2013), Fang and Scherer (2015)).

5.2.1 Méthodes basées sur l'ajustement de faisceaux assisté par un modèle

L'ajustement de faisceaux continue à être l'état de l'art des techniques pour résoudre le problème de la localisation et de la cartographie simultanée par vision (Mouragnon et al. (2006a), Klein and Murray (2007), Strasdat et al. (2010), Huang et al. (2011)). Comme expliqué précédemment, ces techniques nécessitent des informations supplémentaires pour résoudre le problème d'échelle. De plus les capteurs visuels sont sujets au bruit et sensibles aux conditions d'illumination. Les chercheurs ont déjà proposé de modifier l'ajustement de faisceaux pour prendre en compte des informations absolues. Ces méthodes sont parfois appelées ajustement de faisceaux assisté par un modèle ou ajustement de faisceaux contraint (Nilsson et al. (2013), Larnaout et al. (2013)).

Une partie de ces méthodes, propose de contraindre la trajectoire de la caméra, dans l'ajustement de faisceaux, en utilisant des informations absolues issues de systèmes GPS ou issues de la connaissance de la structure de la scène, ou une combinaison des deux (Ellum (2006), Lhuillier (2011), Nilsson et al. (2013), Larnaout et al. (2013)). D'autres méthodes proposent d'utiliser le modèle numérique de terrain (Lerner et al. (2006)). Geva et al. (2015) ont proposé une méthode dans laquelle un drone parcourt une zone reculée. Ils ont utilisé des modèles numériques de terrain (MNT) pour optimiser les positions des points 3D observés par une caméra montée sur le drone. Ils intègrent des informations sur le MNT dans la fonction de coût de l'ajustement de faisceaux en plaçant les points 3D sur la surface du terrain à laquelle ils sont associés. L'association point/surface se fait par lancer de rayons et l'erreur à minimiser dans l'ajustement de faisceaux est une distance euclidienne entre la position du point 3D et sa surface associée.

D'autres méthodes proposent d'utiliser les modèles 3D obtenus par conception assistée par ordinateur (modèles CAO). Ozog and Eustice (2016) proposent de classifier les points 3D selon leur appartenance à un modèle (CAO) connu au préalable, en utilisant l'algorithme d'espérance maximisation (Santos (2015)). Pour l'étape d'association point/plan, les auteurs utilisent une fonction de distance signée, où une valeur 1 est attribuée aux points appartenant à la partie connue de l'environnement et une valeur 0 est attribuée aux points appartenant à la partie absente du modèle. Les poses des caméras et les paramètres des points 3D sont optimisés par maximisation d'une fonction de vraisemblance. Tamaazousti et al. (2011a) présentent une méthode

pour estimer la position de la caméra par rapport à un modèle 3D connu, avec une contrainte d'appartenance à des plans du modèle CAO. L'association point/plan se fait par lancer de rayons dans un processus de minimisation d'une erreur de re-projection. Les points appartenant à un plan du modèle sont contraints à se déplacer uniquement sur ce plan. Cette contrainte est intégrée dans l'ajustement de faisceaux d'un SLAM visuel, permettant ainsi de combiner des points appartenant à la partie connue de l'environnement et les points appartenant à la partie inconnue de l'environnement dans une même fonction de coût.

5.2.2 Méthodes basées sur le RGBD-MCL assisté par un modèle

Ces techniques exploitent les modèles de l'environnement comme information géométrique absolue, dans des algorithmes statistiques se basant sur les filtres à particules combinés à des données de profondeur et des données visuelles. La résolution se fait sur deux étapes : une étape de propagation consistant à calculer le déplacement du capteur par une technique classique d'odométrie visuelle ou RGBD-SLAM et une étape de minimisation d'une fonction de vraisemblance portant sur la cohérence entre la carte de profondeur fournie par le capteur et le plan 3D tel que supposé être observé à la pose estimée.

Fallon et al. (2012) ont proposé une méthode pour estimer les six degrés de liberté de la pose en environnements intérieurs en utilisant un algorithme de Monte-Carlo avec une caméra RGBD. Leur technique nécessite une phase de cartographie préliminaire, obtenue par le passage préalable d'un capteur de type lidar pour créer le modèle 3D, puis une phase d'extraction des plans principaux. Se basant sur cette carte comme donnée connue, un algorithme d'odométrie visuelle est utilisé pour la propagation des particules, et une fonction de vraisemblance est calculée pour chaque particule en comparant les données actuelles avec les données prédites grâce à la connaissance de la géométrie de la scène. Cette fonction de vraisemblance repose sur la comparaison entre la carte de profondeur courante fournie par le capteur et l'image de profondeur prédite à partir du modèle 3D et de la pose prédite pour cette particule.
Fang and Scherer (2015) proposent une méthode de localisation des six degrés de liberté de la caméra, basée sur les filtres à particules pour estimer la pose de la caméra par rapport à un modèle 3D global connu. Ils proposent également une méthode de type MCL dans laquelle la fonction de vraisemblance repose sur la cohérence entre la carte de profondeur fournie par le capteur et le plan 3D tel que supposé être observé à la pose estimée. La fonction de propagation des particules, comme dans Fallon et al. (2012), repose sur le calcul du déplacement entre les images mais par une technique d'odométrie RGBD. Les auteurs proposent d'estimer le déplacement inter-images par une technique rapide à partir des cartes de profondeurs. En cas d'échec, cette dernière est remplacée par une technique d'odométrie visuelle.
Winterhalter et al. (2015) présente également une approche de MCL pour les capteurs RGBD. L'odométrie est calculée en combinant des données RGBD et des données issues d'une centrale inertielle. Le modèle de vraisemblance pour le filtre à particules, attribue une probabilité à la mesure de profondeur en utilisant une fonction de distance entre cette mesure et le plan du sol donné par le modèle de l'environnement.
Les principaux inconvénients de ces méthodes sont les suivants :

- Elles sont généralement lentes et couteuses en temps de calcul à cause de la complexité de l'algorithme, la diversité des particules et les difficultés de paramétrage. De ce fait, elles nécessitent des unités de calcul de type GPU (Fallon et al. (2012)) ou des calculs déportés sur le cloud (Winterhalter et al. (2015)). Notons cependant que Fang dans (Fang and Scherer (2015)) arrive à obtenir des performances quasi temps réel sur CPU mais en

utilisant des a priori sur la position du plan du sol par rapport au capteur, ce qui limite l'application.
- L'étape de minimisation de la fonction de vraisemblance, n'utilise que l'information de profondeur fournie par le capteur. Cependant, si seule l'information de profondeur est utilisée, le calcul de pose peut souffrir de cas dégénérés dans des environnements peu riches en géométrie. Ces cas peuvent être fréquents : par exemple dans un couloir où le calcul de la position longitudinale est ambiguë. D'autres part, l'information de profondeur peut être erronée voir absente pour certaines surfaces telles que les surfaces vitrées.
- Elles ne permettent pas de fournir une reconstruction de l'environnement.

5.3 RGBD-SLAM Contraint

Le problème que nous traitons dans cette section repose sur le principe qu'une reconstruction d'une scène par l'algorithme RGBD-SLAM devrait s'aligner avec le modèle 3D préalablement connu de cette même scène. Il est possible d'évaluer le respect de cette contrainte en mesurant l'écart entre les positions des points 3D reconstruits et ce modèle 3D en terme de distance euclidienne. Dans ce cas, une distance point/plan peut être calculée. Dans les travaux de Tamaazousti et al. (2011a), les auteurs ont proposé une autre adaptation de ce principe. L'idée est de contraindre un point 3D du nuage de points reconstruit à se déplacer seulement sur le plan auquel il est associé. Ainsi les degrés de liberté de ce point sont réduits à deux degrés de liberté. En intégrant cette contrainte sous forme d'une erreur de re-projection 2D dans l'ajustement de faisceaux, les auteurs ont démontré une amélioration remarquable sur la localisation et la reconstruction 3D issue du SLAM.

C'est cette hypothèse que nous allons exploiter dans nos travaux, en y intégrant l'information de profondeur. En effet avec la disponibilité d'une mesure de profondeur grâce aux capteurs 3D, il est possible d'évaluer l'erreur entre la profondeur du point 3D reconstruit (qui est fournie par le capteur) et la profondeur estimée du plan auquel il est associé (Figure 5.3). Cette distance est intégrée dans l'ajustement de faisceaux sous forme d'une erreur 1D sur les profondeurs, pour permettre une meilleure localisation de la caméra.

Dans la réalité, les modèles 3D disponibles peuvent être incomplets (voir Figure 5.4.), les points 3D reconstruits par le SLAM n'appartiennent pas forcément à un plan connu du modèle. Une étape d'association est donc nécessaire pour attribuer un plan ou non à chaque primitive 3D. De plus, pour mieux contraindre le nuage de points issu du SLAM-RGBD et garantir une bonne convergence de l'ajustement de faisceaux, un maximum de points doit être utilisé. Par conséquent les autres points n'appartenant à aucun plan du modèle seront tout de même intégrés dans le processus d'optimisation. Les détails sur la gestion de ces points seront présentés dans la section 5.3.1.

5.3.1 Ajustement de faisceaux Contraint par les profondeurs et par le modèle 3D

Dans nos travaux nous souhaitons améliorer la localisation du SLAM-RGBD par l'ajout d'une information absolue sans toutefois dégrader ses performances calculatoires. Pour cette raison, nous intégrons les informations sur les plans de la scène dans l'ajustement de faisceaux seulement, cela va nous permettre d'avoir une contrainte supplémentaire dans le processus

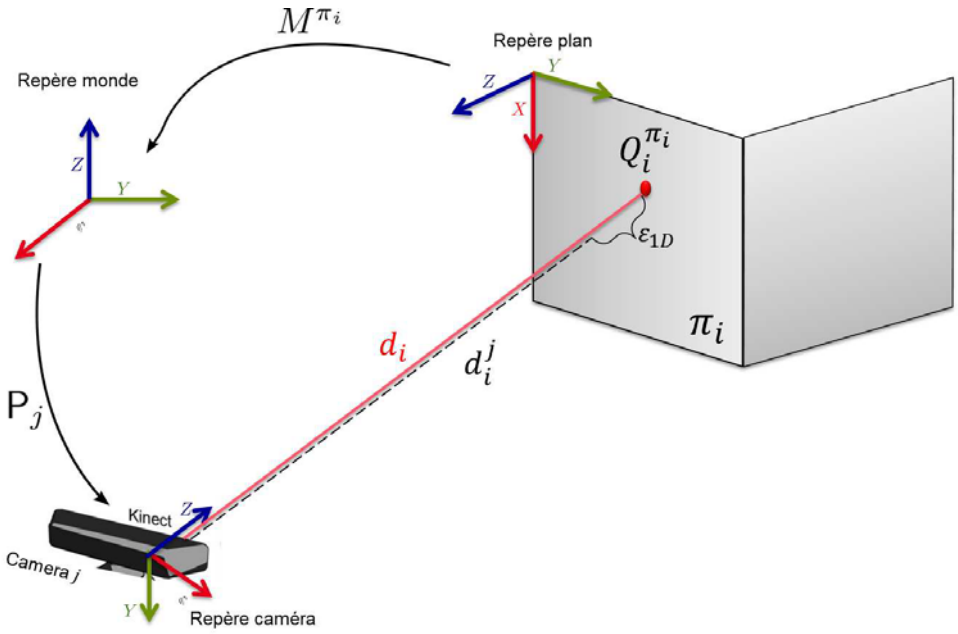


FIGURE 5.3 – Un point 3D $\mathcal{Q}_i^{\pi_i}$ appartenant au plan $\boldsymbol{\pi}_i$ dont la mesure de profondeur est disponible à l'image j, est transformé dans le repère monde grâce à la matrice de passage M^{π_i}, puis dans le repère de la caméra j grâce à la matrice de pose P_j. L'erreur à minimiser est la différence entre la profondeur estimée de ce point 3D, exprimée dans la caméra j, et la profondeur réelle mesurée dans cette même caméra.

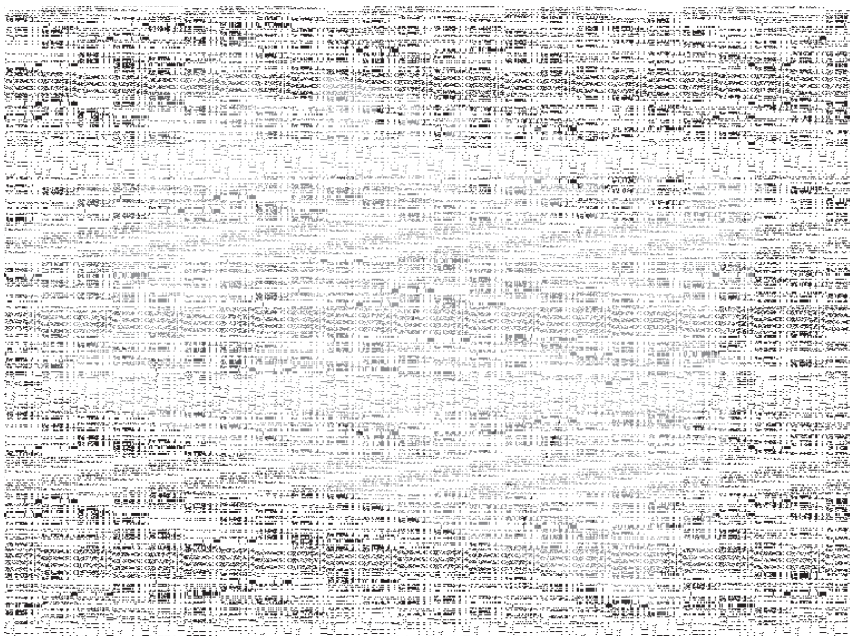


FIGURE 5.4 – **Modèle partiel de la scène.** Le modèle 3D d'un bureau où les fenêtres n'ont pas été modélisées.

d'optimisation.

Nous rappelons que l'ajustement de faisceaux du SLAM optimise les poses des $\mathcal{N}_c$ dernières caméras clefs et les paramètres des $\mathcal{N}_p$ points 3D, observés dans un ensemble $\mathcal{A}_i$ de caméras clefs. Nous rappelons ci-après les deux termes de la fonction de coût du RGBD-SLAM :

- L'erreur de re-projection des points visuels :

$$\begin{aligned} f(\{\mathsf{P}_j\}_{j=1}^{\mathcal{N}_c}, \{\mathcal{Q}_i\}_{i\in\mathcal{N}_p}) &= \sum_{i\in\mathcal{N}_p}\sum_{j\in\mathcal{A}_i} \left\| \mathbf{q}_{i,j} - \pi\left(\mathsf{K}\mathsf{P}_j\tilde{\mathcal{Q}}_i\right)\right\|^2 \\ &= \sum_{i\in\mathcal{N}_p}\sum_{j\in\mathcal{A}_i} \left\|\mathbf{f}_{i,j}\right\|^2 \end{aligned} \tag{5.1}$$

- L'erreur de re-projection des points dont la profondeur est connue :

$$\begin{aligned} g\left(\{\mathsf{P}_j\}_{j=1}^{N_c}\right) &= \sum_{i\in\mathcal{N}_p}\sum_{j\in\mathcal{A}_i}\sum_{k\in\mathcal{A}_i}^{k\neq j} \left\| \mathbf{q}_{i,j} - \pi\Big(\mathsf{K}\mathsf{P}_j\mathsf{P}_k^{-1}\pi^{-1}\left(q_{i,k},\ \mathrm{d}_i^k\right)\Big)\right\|^2 \\ &= \sum_{i\in\mathcal{N}_p}\sum_{j\in\mathcal{A}_i}\sum_{k\in\mathcal{A}_i}^{k\neq j} \left\|\mathbf{g}_{i,j}^k\right\|^2 \end{aligned} \tag{5.2}$$

Pour améliorer la précision de la localisation du RGBD-SLAM, nous allons introduire dans la fonction de coût une erreur additionnelle qui prend en compte la connaissance a priori d'un modèle 3D de la scène. Ce modèle consiste en un ensemble de plans $\{\boldsymbol{\pi}_i\}$.
Le principe est inspiré des travaux de Tamaazousti et al. (2011b) réalisé dans le cadre d'un SLAM visuel. Notre contribution est de généraliser ce principe dans le cadre d'un RGBD-SLAM.

La méthode proposée par Tamaazousti et al. (2011b), nommée *SLAM Contraint*, consiste à contraindre dans l'ajustement de faisceaux certains points grâce à leur plan associé. L'idée est qu'un point 3D $\mathcal{Q}_i^{\boldsymbol{\pi}_i}$ appartenant à un plan $\boldsymbol{\pi}_i$ ne possède que deux degrés de liberté. Sa coordonnée $\mathbf{Z}^{\boldsymbol{\pi}_i}$ le long de la normal du plan $\boldsymbol{\pi}_i$ est supposée nulle. La nouvelle paramétrisation du point $\mathcal{Q}_i^{\boldsymbol{\pi}_i}$ est la suivante :

$$\mathcal{Q}_i^{\boldsymbol{\pi}_i} = \begin{pmatrix} \mathbf{X}^{\boldsymbol{\pi}_i} \\ \mathbf{Y}^{\boldsymbol{\pi}_i} \\ 0 \end{pmatrix} \tag{5.3}$$

où : $\mathbf{X}^{\boldsymbol{\pi}_i}$ et $\mathbf{Y}^{\boldsymbol{\pi}_i}$ sont les coordonnées du point $\mathcal{Q}_i$ exprimées dans le repère associé au plan $\boldsymbol{\pi}_i$.

Ce point se déplace uniquement sur le plan $\boldsymbol{\pi}_i$ auquel il est associé. Les poses des caméras et les paramètres des points 3D sont alors optimisés en minimisant une erreur de re-projection définie par :

$$\varepsilon_{2D} = \mathbf{q}_{i,j} - \pi\left(\mathsf{K}\mathsf{P}_j\mathsf{M}^{\boldsymbol{\pi}_i}\tilde{\mathcal{Q}}_i^{\boldsymbol{\pi}_i}\right) \tag{5.4}$$

où : $\mathsf{M}^{\boldsymbol{\pi}_i}$ est la matrice de passage du repère associé au plan $\boldsymbol{\pi}_i$ vers le repère monde.
Sur l'ensemble des images $\mathcal{A}_i$, observant le point 3D $\mathcal{Q}_i$, la fonction de coût utilisée est :

$$\begin{aligned} k\left(\{\mathsf{P}_j\}_{j=0}^{\mathcal{N}_c}, \{\mathcal{Q}_i^{\boldsymbol{\pi}_i}\}_{i\in\mathcal{N}_{\boldsymbol{\pi}}}\right) &= \sum_{i\in\mathcal{N}_{\boldsymbol{\pi}}}\sum_{j\in\mathcal{A}_i} \left\| \mathbf{q}_{i,j} - \pi\left(\mathsf{K}\mathsf{P}_j\mathsf{M}^{\boldsymbol{\pi}_i}\tilde{\mathcal{Q}}_i^{\boldsymbol{\pi}_i}\right)\right\|^2 \\ &= \sum_{i\in\mathcal{N}_{\boldsymbol{\pi}}}\sum_{j\in\mathcal{A}_i} \left\|\mathbf{k}_{i,j}\right\|^2 \end{aligned} \tag{5.5}$$

où : $\mathcal{N}_{\boldsymbol{\pi}}$ est le nombre de points 3D à optimiser.

Nous proposons de généraliser cette contrainte sur des données RGBD. Ainsi, chaque point 3D $\mathcal{Q}_i$ associé au plan $\boldsymbol{\pi}_i$ [5] dont la mesure de profondeur d_i est connue, va contribuer dans le processus d'optimisation avec une erreur sur la profondeur, comme le montre la Figure 5.3. Cette erreur est définie comme suit :

$$\varepsilon_{1D} = \mathrm{d}_i^j - [\mathsf{P}_j \mathsf{M}^{\boldsymbol{\pi}_i} \tilde{\mathcal{Q}}_i^{\pi_i}]_z \tag{5.6}$$

où : $[\mathsf{P}_j \mathsf{M}^{\boldsymbol{\pi}_i} \tilde{\mathcal{Q}}_i^{\pi_i}]_z$ est la $3^{ième}$ coordonnée du point 3D exprimé dans le repère de la caméra j.

Cette erreur est calculée sur l'ensemble des points 3D $\mathcal{Q}_i^{\pi_i} \in \mathcal{N}_{\boldsymbol{\pi}}$ et sur l'ensemble des images $\mathcal{A}_i$ l'observant.
La fonction de coût associée est définie par :

$$l\left(\{\mathsf{P}_j\}_{j=0}^{\mathcal{N}_c}, \{\mathcal{Q}_i^{\boldsymbol{\pi}_i}\}_{i\in\mathcal{N}_{\boldsymbol{\pi}}}\right) = \sum_{i\in\mathcal{N}_{\boldsymbol{\pi}}} \sum_{j\in\mathcal{A}_i} \left\| \mathrm{d}_i^j - [\mathsf{P}_j \mathsf{M}^{\boldsymbol{\pi}_i} \tilde{\mathcal{Q}}_i^{\pi_i}]_z \right\|^2. \tag{5.7}$$

5.3.2 Gestion des différentes erreurs dans la fonction de coût

L'ajustement de faisceaux que nous proposons repose sur une fonction de coût à plusieurs termes :

1. Un terme lié à l'erreur de re-projection des points 3D de l'environnement, qui n'appartiennent pas au modèle 3D et n'ayant pas d'information de profondeur (équation 5.1).
2. Un terme lié aux résidus des points ayant une mesure de profondeur mais qui n'appartiennent pas au modèle 3D (équation 5.2).
3. Un terme lié aux résidus des points associés à un plan du modèle 3D mais dont la profondeur n'est pas connue (équation 5.5).
4. Un terme lié aux résidus des points ayant une mesure de profondeur et associés à un plan du modèle (équation 5.7).

Pour pouvoir combiner ces différentes erreurs dans une même fonction de coût, deux considérations doivent être prises en compte. La première concerne la manière de combiner une erreur en mètre (l'erreur sur les profondeurs des points dans l'équation 5.7.), et les autres erreurs exprimées en pixel (équation 5.5, 5.2 et 5.1). La seconde concerne la manière de pondérer ces différentes erreurs.

Combinaison de termes de natures différentes

Pour pouvoir combiner une erreur en pixel et une erreur en mètre, nous choisissons d'appliquer un facteur de pondération sur les profondeurs, tel que proposé par Scherer et al. (2012) et expliqué dans la section 4.3.

En intégrant ce facteur de pondération dans l'équation 5.7, nous obtenons :

$$\begin{aligned} l\left(\{\mathsf{P}_j\}_{j=0}^{\mathcal{N}_c}, \{\mathcal{Q}_i^{\boldsymbol{\pi}_i}\}_{i\in\mathcal{N}_{\boldsymbol{\pi}}}\right) &= \sum_{i\in\mathcal{N}_{\boldsymbol{\pi}}} \sum_{j\in\mathcal{B}_i} \left\| \frac{\mathrm{d}_i^j - [\mathsf{P}_j \mathsf{M}^{\boldsymbol{\pi}_i} \tilde{\mathcal{Q}}_i^{\pi_i}]_z}{\alpha \mathrm{d}_i^{j\,2}} \right\|^2 \\ &= \sum_{i\in\mathcal{N}_{\boldsymbol{\pi}}} \sum_{j\in\mathcal{A}_i} \|\mathbf{l}_i\|^2 \end{aligned}. \tag{5.8}$$

5. Une étape d'association point/plan est effectuée. Cette étape est identique pour les points ayant une mesure de profondeur associée et les points sans mesure de profondeur connue.

Pondération des différentes erreurs dans la fonction de coût

Dans une fonction de coût combinant plusieurs termes (de même nature ou de natures différentes), un des problèmes à gérer est la manière de pondérer ces différents termes pour contrôler leur influence d'une part, et d'autre part, la manière de déterminer les seuillages permettant de rejeter les points aberrants, ces derniers pouvant nuire à l'optimisation. Ces points aberrants peuvent provenir de plusieurs causes (par exemple, mauvaises associations entre les points et les plans du modèle, erreurs dans la mise en correspondance des primitives 2D, bruits sur les mesures visuelles et sur les mesures de profondeur).

Dans nos travaux nous choisissons d'utiliser une fonction de coût robuste, ainsi l'influence de chaque terme est contrôlée par le seuil de rejet d'un estimateur robuste. De cette manière les résidus sont normalisés et l'influence des points aberrants est diminuée.

De manière similaire aux travaux de Tamaazousti et al. (2011a), l'estimateur robuste utilisé dans nos travaux est l'estimateur de Geman-McClure. Ainsi la norme $\mathbf{L}_2$ dans les équations 5.1, 5.2, 5.7, 5.8, est remplacée par une norme robuste rappelée dans la section 3.2.2.

5.3.3 Fonction de coût résultante

La fonction de coût du RGBD-SLAM Contraint, qui combine les différents termes présentés précédemment est donnée par :

$$\begin{aligned} l\left(\{\mathcal{P}_j\}_{j=0}^{N_c}, \{\mathcal{Q}_i\}_{i\in\mathcal{N}_p}, \{\mathcal{Q}_i^{\pi_i}\}_{i\in\mathcal{N}_\pi}\right) = & \sum_{i\in\mathcal{N}_p}\sum_{j\in\mathcal{A}_i} \rho\left(\|\mathbf{f}_{i,j}\|, c_f\right) + \sum_{i\in\mathcal{N}_p}\sum_{j\in\mathcal{A}_i}\sum_{k\in\mathcal{A}_i}^{k\neq j} \rho\left(\left\|\mathbf{g}_{i,j}^k\right\|, c_g\right) \\ & + \sum_{i\in\mathcal{N}_\pi}\sum_{j\in\mathcal{A}_i} \rho\left(\|\mathbf{k}_{i,j}\|, c_h\right) + \sum_{i\in\mathcal{N}_\pi}\sum_{j\in\mathcal{A}_i} \rho\left(\|\mathbf{l}_i\|, c_i\right) \end{aligned} \tag{5.9}$$

où c_f, c_g, c_h, c_i sont les seuils de rejet de l'estimateur. Ces seuils sont calculés séparément sur chaque terme par l'écart médian absolu (Absolute Median Deviation, MAD).

Adopter ce schéma permet de bénéficier de plusieurs avantages :

- Traiter différents types d'environnements pouvant être rencontrés durant la navigation : environnements avec peu de structures 3D, environnements peu texturés.
- Exploiter différentes informations liées au nuage de points 3D : profondeurs des points, texture, leurs plans associés, tous dans un même processus d'optimisation.
- Garantir des performances temps réel, notamment en préservant les structures creuses de la matrice Hessienne et Jacobienne.

5.3.4 Étapes complémentaires

5.3.4.1 Initialisation du SLAM-RGBD Contraint au modèle

Cette étape consiste à calculer la position et l'orientation initiale de la caméra par rapport à un repère fixé sur la scène. Dans la littérature, plusieurs méthodes ont été proposées pour réaliser l'initialisation. Le lecteur est invité à se référer aux travaux de Tamaazousti (2013) pour plus de détails sur ces méthodes. Dans nos expérimentations nous utilisons une méthode, exploitant une cible codée.

Initialisation par une cible codée localisée par rapport au modèle de la scène

Il est possible d'effectuer le recalage initial en positionnant une cible codée géolocalisée par rapport au modèle de la scène. La cible codée est facilement détectable dans les images. Cette cible comprend des points particuliers dont les coordonnées 3D sont connues. Un calcul de pose (par mise en correspondance de points 3D-2D) avec l'algorithme de Dementhon and Davis (1995) est réalisé pour estimer la première pose de la caméra. La Figure 5.5. présente un exemple d'une cible codée utilisée pour l'initialisation de nos algorithmes. Cette cible est positionnée à une distance connue d'un repère fixe lié au modèle 3D. Les coordonnées des points particuliers sont calculées manuellement.

FIGURE 5.5 – Exemple de cible codée utilisée pour calculer la pose de la caméra par rapport à un repère fixé sur le modèle de la scène.

5.3.4.2 L'association point-plan

Dans cette section nous rappelons la méthode d'association point/plan proposée par Tamaazousti (2013) qui est directement utilisée dans notre solution.
Pour pouvoir utiliser les contraintes liées au modèle 3D de la scène, une étape d'association entre les points 3D et les plans du modèle est nécessaire. Cette étape est effectuée par un lancer de rayons en faisant correspondre chaque point $\mathcal{Q}_i$ reconstruit à partir de chacune de ses observations sur différentes images $\{\mathbf{q}_{i,j}\}_{j\in\mathcal{B}_i}$, au plan le plus proche en terme de distance euclidienne. $Card(\mathcal{A}_i)$ votes sont alors obtenus pour les différents plans et le choix majoritaire est conservé. Une fois la classification effectuée pour chaque point 3D associé à un plan $\boldsymbol{\pi}_i$ du modèle, une projection des points sur le plan est réalisée. Le barycentre des intersections entre les lancers de rayons et le plan $\boldsymbol{\pi}_i$ est sélectionné comme étant la position initiale du point 3D $\mathcal{Q}_i$ lors de la minimisation de l'erreur dans l'équation 5.4.

Ces points 3D sont exprimés dans le repère lié au plan auquel ils sont associés. C'est cette nouvelle paramétrisation que nous allons intégrer dans notre solution.

L'étape d'association point/plan utilise la pose estimée de la caméra, si cette dernière est mal estimée, les associations point/plan vont être erronées ce qui peut conduire à une mauvaise convergence de l'ajustement de faisceaux. La solution proposée par Tamaazousti (2013) consiste à appliquer un processus itératif lors de la minimisation de l'erreur dans l'équation 5.4. Ce processus est répété jusqu'à atteindre le critère d'arrêt. Ce dernier correspond à une décroissance du MAD jusqu'à un seuil fixé, calculé à partir des erreurs de re-projection des points associés aux plans. Le processus d'optimisation proposé par Tamaazousti (2013) est résumé dans l'algorithme 1.

```
nouveauMAD = 0;
ancienMAD = 0;
repeat
    foreach {Q^i}_{i∈A_i} do
        Associer le point Q^i au plan le plus proche π_i ;
        Projeter orthogonalement le point Q^i sur le plan π_i ;
    end
    ancienMAD = MAD calculé sur les erreurs de re-projection des points {Q^i}_{i∈A_i} ;
    Calculer le seuil de rejet du M-estimateur c;
    Minimiser la fonction de coût considérée en utilisant l'algorithme de Levenberg
     Marquardt ;
    nouveauMAD = MAD calculé sur les erreurs de re-projection de points {Q^i}_{i∈A_i} ;
    Triangulation des points {Q^i}_{i∈M} en prenant en compte les nouvelles poses de la
     caméra ;
until (nouveauMAD − ancienMAD) < 0.1;
```

Algorithme 1 : Le processus itératif pour l'association point/plan proposé par Tamaazousti et al. (2011b)

5.3.5 Conclusion

Dans ce chapitre, nous avons présenté une méthode qui combine des contraintes sur les profondeurs des points et des contraintes planaires dans un algorithme de localisation et de cartographie simultanée. Ces informations sont intégrées dans le processus d'ajustement de faisceaux. La principale contribution de cette méthode est de proposer une autre formulation de la fonction de coût de l'ajustement de faisceaux. Cette dernière considère plusieurs caractéristiques du point 3D ; sa mesure de profondeur, son appartenance à un environnement connu a priori ainsi que son appartenance à un environnement inconnu. Cette méthode sera évaluée dans le chapitre suivant, sur un ensemble de séquences de synthèse et de séquences réelles tirées de bases de données de l'état de l'art. Nous présenterons aussi une évaluation qualitative sur notre propre séquence réelle acquise avec un capteur *Kinect*V1 dans un couloir d'un bâtiment.

CHAPITRE 6

Évaluation expérimentale du RGBD-SLAM Contraint

Ce chapitre présente les expérimentations réalisées pour évaluer le RGBD-SLAM Contraint. Celles-ci sont menées sur des séquences de synthèse et sur des séquences réelles tirées de bases de données de l'état de l'art, ainsi que sur nos propres séquences réelles réalisées dans un couloir d'un bâtiment.
Pour évaluer notre algorithme, nous utilisons les mêmes erreurs statistiques que celles présentées dans le chapitre 4 (voir la section 4.1). Pour montrer l'apport de la solution proposée nous comparons les quatre algorithmes suivants :

- *Le SLAM visuel.*
- *Le RGBD-SLAM proposé dans le chapitre 3.*
- *Le SLAM Contraint visuel de Tamaazousti et al. (2011a).*
- *Le RGBD-SLAM Contraint que nous proposons, qui intègre la fonction de coût définie par l'équation 5.9.*

Dans toutes nos expérimentations, le facteur empirique α est fixé à 0.03.

6.1 Séquences de synthèse

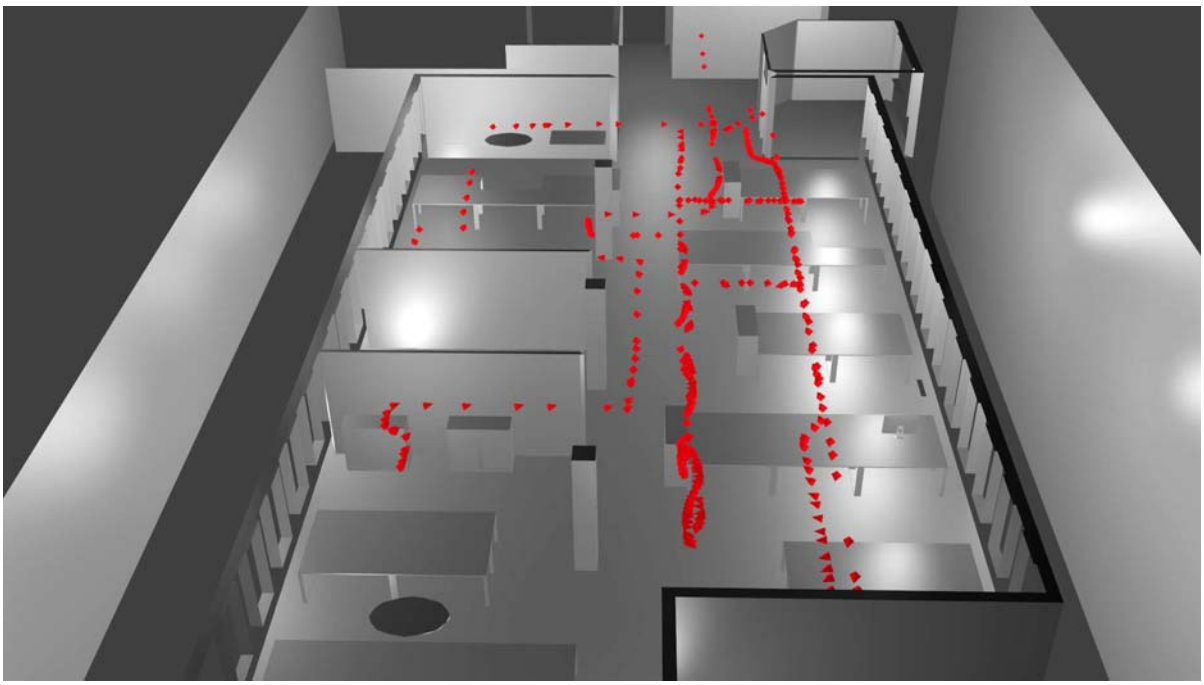
6.1.1 Séquence de synthèse "Bureaux"

Nous utilisons les séquences de synthèse identiques à celles présentées dans le chapitre 4. Nous rappelons dans la Figure 6.1. la séquence de synthèse "Bureaux". Les caractéristiques de cette séquence sont présentées dans la section 4.3.1. La Figure 6.1. montre la trajectoire réelle effectuée par la caméra ainsi que le modèle 3D de la scène.

Résultats : Les résultats obtenus sont présentés dans la Figure 6.2. et le tableau 6.1. Ces résultats montrent que les meilleures performances en terme de précision de la localisation, sont celles obtenues avec le RGBD-SLAM et le RGBD-SLAM Contraint.
Dans cette séquence les données de profondeur sont idéales, ce qui explique une faible différence entre les résultats délivrés par le RGBD-SLAM et ceux délivrés par le RGBD-SLAM Contraint.

(a)

(b)



(c)

FIGURE 6.1 – **Séquence de synthèse "Bureaux".** (a) : image couleur, (b) : image de profondeur associée, (c) : représente le modèle 3D de l'environnement et la trajectoire réelle de la caméra en rouge.

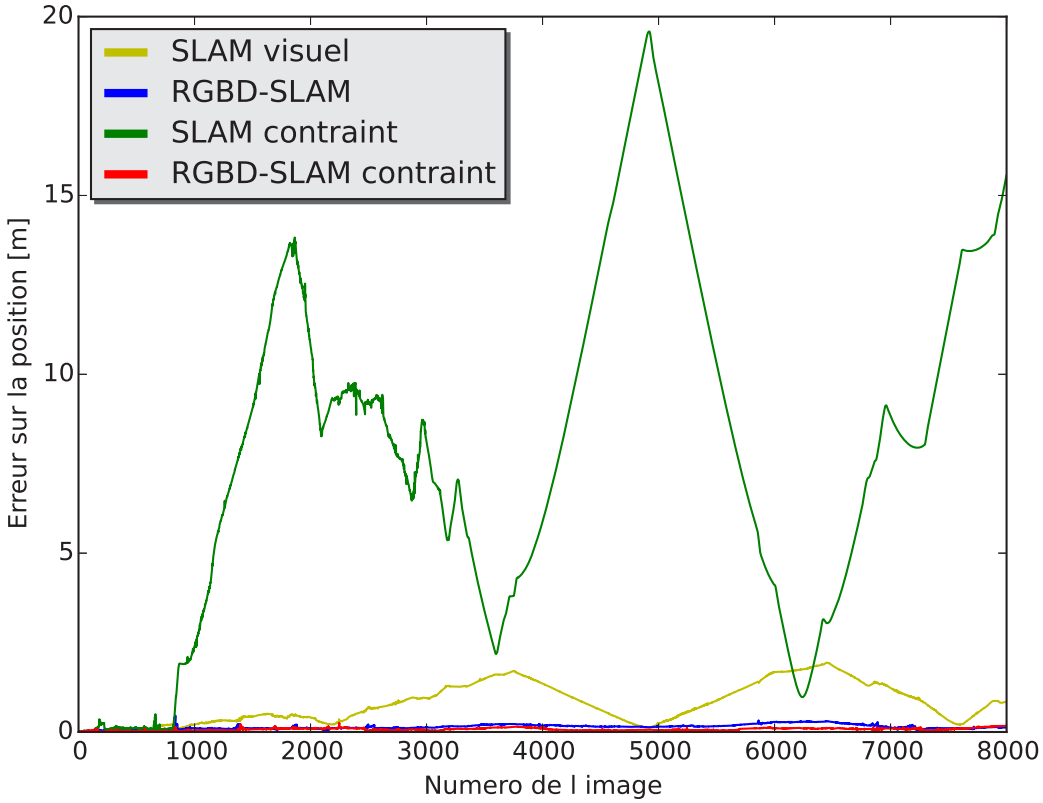


FIGURE 6.2 – Courbes d'évolution de l'erreur de position absolue (Absolute Trajectory Error, ATE) sur les quatre algorithmes (SLAM visuel, RGBD-SLAM, SLAM Contraint visuel et RGBD-SLAM Contraint), pour la séquence "Bureaux".

Erreurs [m] / Methodes	RMSE	STD	Max
SLAM visuel	0.961	0.575	1.940
RGBD-SLAM	0.052	0.039	0.239
SLAM Contraint	4.342	2.217	19.585
RGBD-SLAM Contraint	0.036	0.026	0.217

TABLE 6.1 – Statistiques des erreurs du SLAM visuel, du RGBD-SLAM, du SLAM Contraint visuel et du RGBD-SLAM Contraint sur la séquence de synthèse "Bureaux".

Dans le cas du SLAM Contraint visuel, nous notons une dérive importante. L'erreur sur la position atteint un RMSE de 4.342m et un maximum de 19.5m qui représente 14.5% de la longueur de la trajectoire totale effectuée par la caméra. Entre l'image numéro 811 et l'image numéro 871 (Figure 6.3.), l'erreur de position du SLAM Contraint visuel passe de 0.1m à 1.9m. Sur cette partie de la trajectoire, la scène est peu texturée comme le montre la Figure 6.4., ce qui démontre la sensibilité du SLAM Contraint visuel aux environnements peu texturés. De plus les erreurs de dérive conduisent à de mauvaises associations points/plans, qui à leurs tours, conduisent à une mauvaise convergence de l'ajustement de faisceaux. Ceci explique pourquoi la dérive du SLAM Contraint visuel n'est jamais rattrapée. Intégrer l'information de profondeur en combinaison avec la contrainte par plans (RGBD-SLAM Contraint) permet de robustifier l'algorithme. Bien que le peu de texture dans l'environnement, mette en difficulté notre méthode (des pics d'erreurs sont enregistrés à ces endroits (Figure 6.3.)), cette erreur ne dépasse jamais 0.15% de la longueur de la trajectoire totale. D'autre part la dérive est rattrapée au cours du temps et l'erreur de position se stabilise autours de 0.026m (tableau 6.1).

6.1.2 Séquence de synthèse " Living room"

La deuxième séquence de synthèse utilisée est présentée dans la Figure 4.4. Elle est tirée de la base de données réalisée par Handa et al. (2014). Rappelons que sur cette séquence, les images couleurs et les images de profondeur ont été artificiellement bruitées pour simuler le bruit du capteur *Kinect*. Les détails sur ces bruits sont présentés dans la section 4.3.1.2. De plus le modèle 3D réel de la scène est fourni (Figure 6.5).

Résultats : Le tableaux 6.2, présente les erreurs de position obtenues avec les quatre algorithmes : le SLAM visuel, le RGBD-SLAM, le SLAM Contraint visuel et le RGBD-SLAM Contraint. Sur ces deux trajectoires, le RGBD-SLAM Contraint que nous proposons réduit d'un

Méthodes	Erreurs (ATE) [m]	kt1	kt2
SLAM visuel	RMSE	0.251	0.235
	STD	0.085	0.101
	MAX	0.442	0.446
RGBD-SLAM	RMSE	0.041	0.037
	STD	0.018	0.017
	MAX	0.0942	0.089
SLAM Contraint	RMSE	-	0.067
	STD	-	0.041
	MAX	-	0.241
RGBD-SLAM Contraint	RMSE	**0,025**	**0.023**
	STD	**0.015**	**0.011**
	MAX	**0.087**	**0.093**

TABLE 6.2 – Statistiques des erreurs (ATE) obtenues sur la séquence "Living room" avec le SLAM visuel, le RGBD-SLAM, le SLAM Contraint visuel et le RGBD-SLAM Contraint.

facteur 2 l'erreur de position obtenue avec RGBD-SLAM. L'écart type obtenu est également

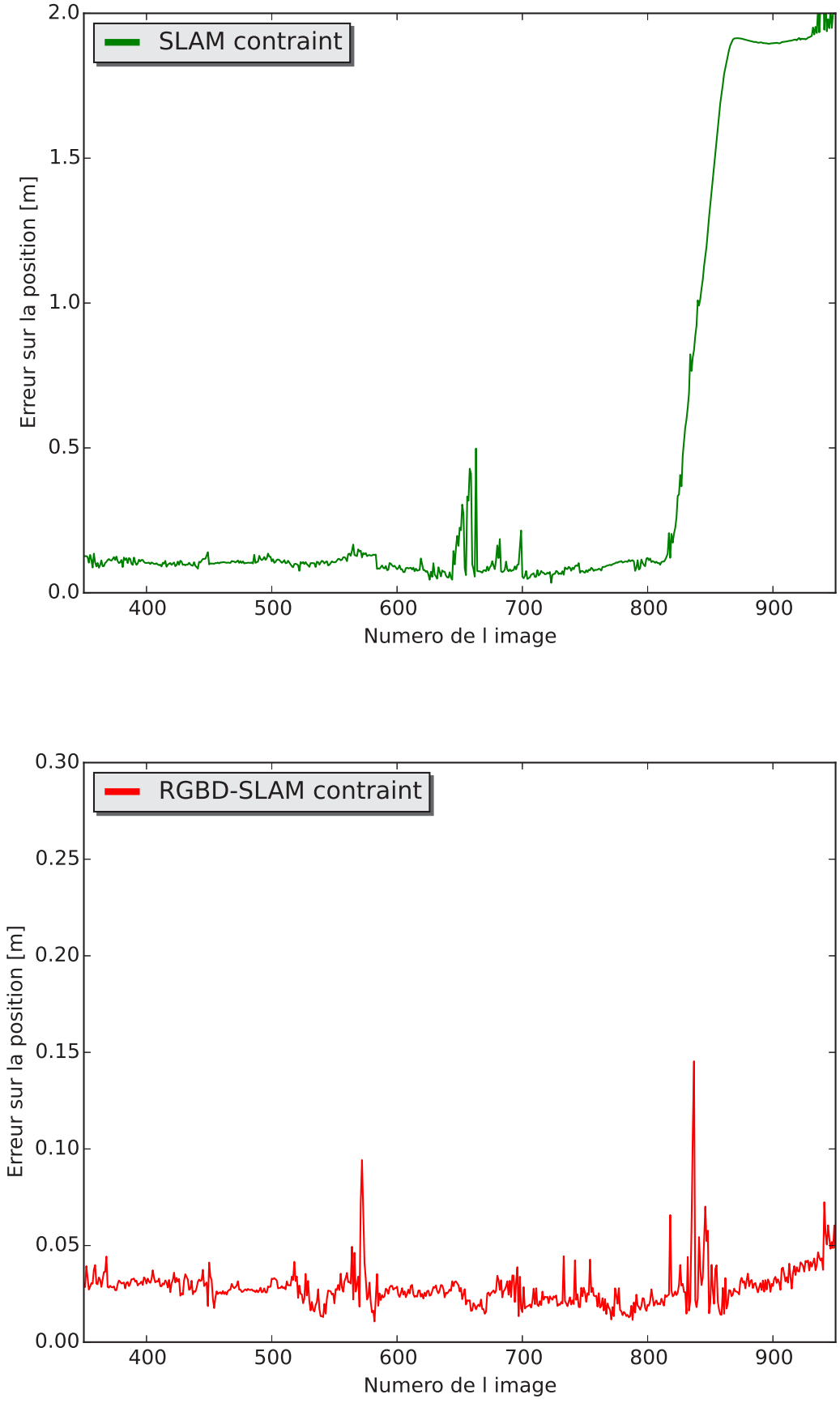
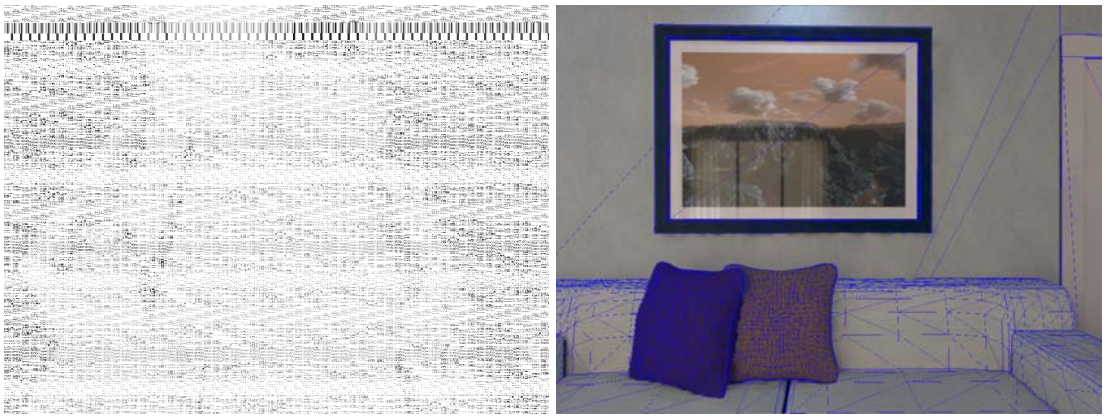


FIGURE 6.3 – **Séquence de synthèse "Bureaux"**. Erreur de position du SLAM Contraint visuel et du RGBD-SLAM Contraint entre l'image 350 et l'image 950.

(a) Image 811 (b) Image 830

FIGURE 6.4 – **Illustrations d'une portion de la séquence "Bureaux".** La scène apparente, peu texturée, tend à faire dériver, d'une manière importante le RGBD-SLAM Contraint.



(a) (b)

FIGURE 6.5 – **Séquence de synthèse "Living room"**. (a) : le modèle 3D réel de la scène, (b) : la projection du modèle (en bleu) sur la première image de la séquence.

plus faible. Ces résultats mettent en évidence l'apport de combiner la contrainte par plans et la mesure de profondeur. En effet même si des erreurs d'associations point/plan peuvent survenir, l'ajustement de faisceaux permet de les corriger en minimisant une erreur sur les profondeurs des points. Pareillement, même si les profondeurs sont bruitées, la contrainte par plans permet de contraindre mieux les points 3D dans l'ajustement de faisceaux.
Notons également la mise en difficulté du SLAM Contraint visuel, liée au fait que la scène soit peu texturée. L'information visuelle seule, n'est pas suffisante dans ce cas.

6.2 Séquences Réelles

6.2.1 Séquences réelles "Corbs"

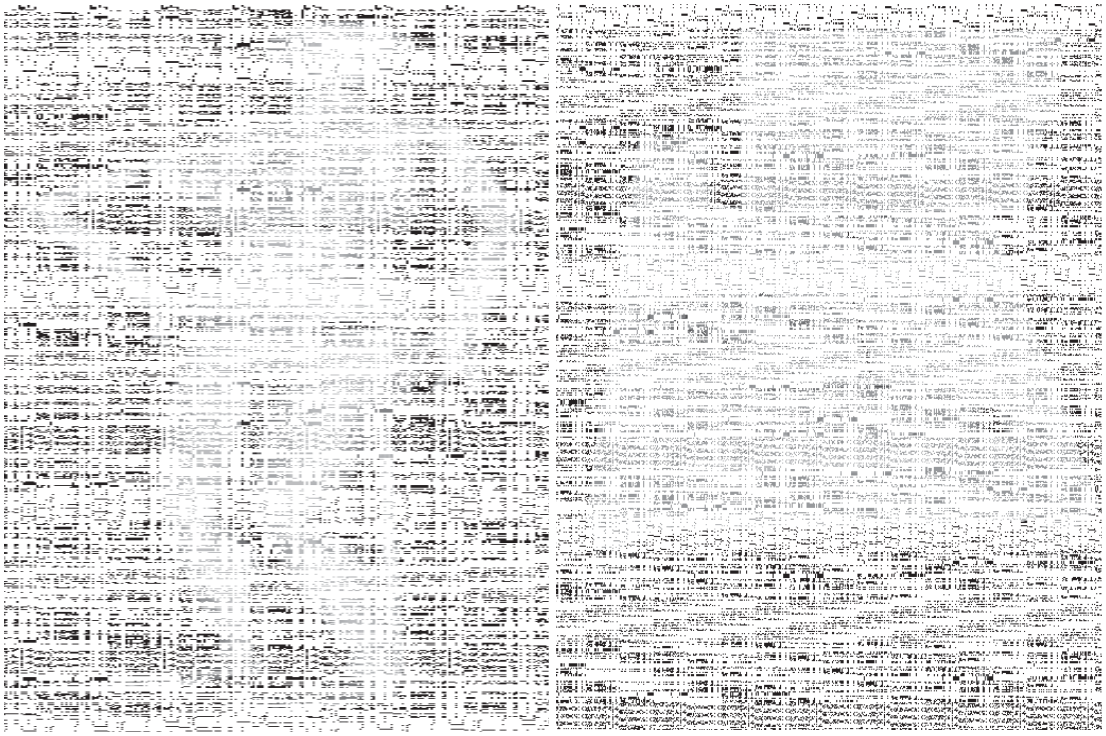
Nous testons les différents algorithmes sur les séquences réelles "Corbs", présentées dans la section 4.3.2.3. Rappelons que ces séquences ont été acquises avec le capteur *KinectV2*. Elles sont fournies avec les trajectoires réelles des caméras (Figure 4.11.) et le modèle 3D partiel de la scène. Il a été généré en utilisant un scanner 3D externe. Ce modèle a une précision similaire à un modèle CAD et il est aligné avec les images de profondeur. (Figure 6.6).

Résultats : Le tableau 6.3 et les Figures (6.7. et 6.8.) résument les résultats obtenus sur les séquences "Corbs". Les erreurs obtenues par le RGBD-SLAM Contraint sont les plus faibles, comparées à celles obtenues avec les autres algorithmes, pour les trois séquences.

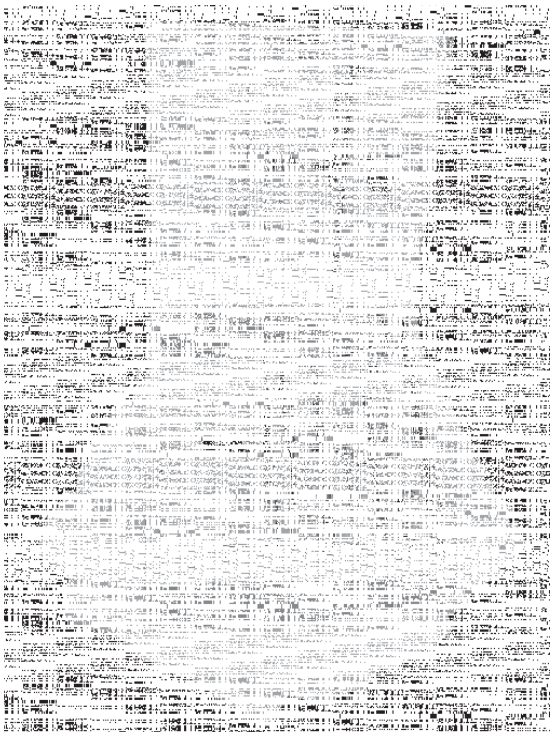
Séquences	algorithmes	les erreurs absolues (m)		
		RMSE	STD	MAX
Human	SLAM visuel	0.583	0.172	1.150
	RGBD-SLAM	0.114	0.055	0.468
	SLAM Contraint visuel	0.283	0.164	0.983
	RGBD-SLAM Contraint	**0.036**	**0.017**	**0.208**
Desk	SLAM visuel	0.165	0.041	0.382
	RGBD-SLAM	0.016	0.006	0.074
	SLAM Contraint visuel	0.022	0.008	0.084
	RGBD-SLAM Contraint	**0.013**	**0.005**	**0.073**
Electrical	SLAM visuel	-	-	-
	RGBD-SLAM	0.065	0.020	0.141
	SLAM Contraint visuel	0.057	0.031	0.148
	RGBD-SLAM Contraint	**0.040**	**0.017**	**0.109**

TABLE 6.3 – Statistiques des erreurs (ATE) obtenues sur les séquences "Corbs" avec le SLAM visuel, le RGBD-SLAM, le SLAM Contraint visuel et le RGBD-SLAM Contraint.

Si nous comparons les résultats obtenus par le RGBD-SLAM (sans la contrainte par plans) et ceux obtenus par RGBD-SLAM Contraint (avec la contrainte par plans), sur la séquence "Human" (Figure 6.9), nous constatons que les deux méthodes enregistrent des pics d'erreurs à plusieurs endroits de la séquence. Néanmoins le RGBD-SLAM Contraint arrive à rattraper la dérive après l'étape d'ajustement de faisceaux. L'exemple présenté dans la Figure 6.10. montre

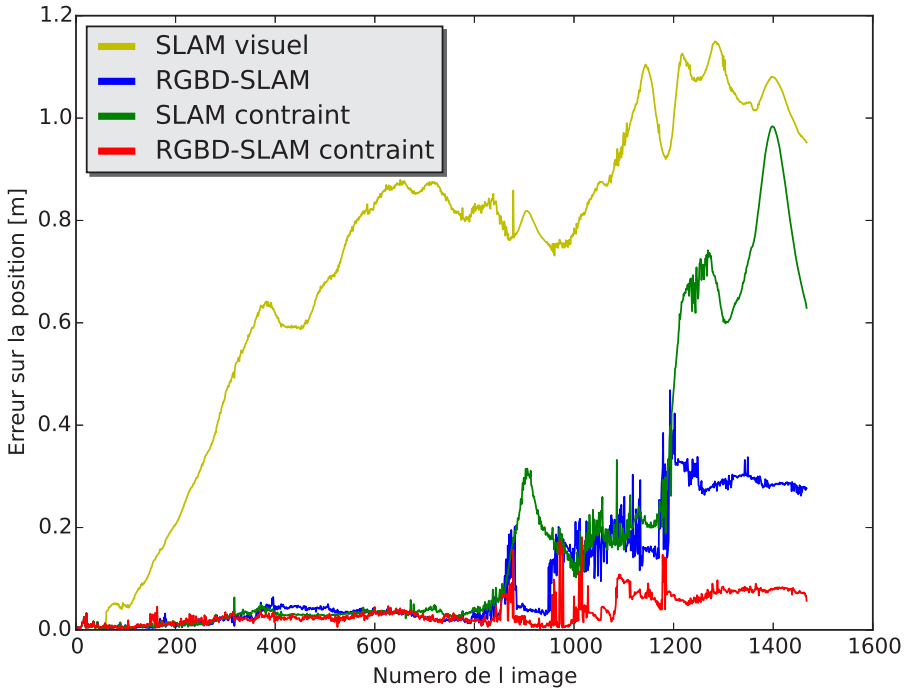


(a) Human (b) Desk

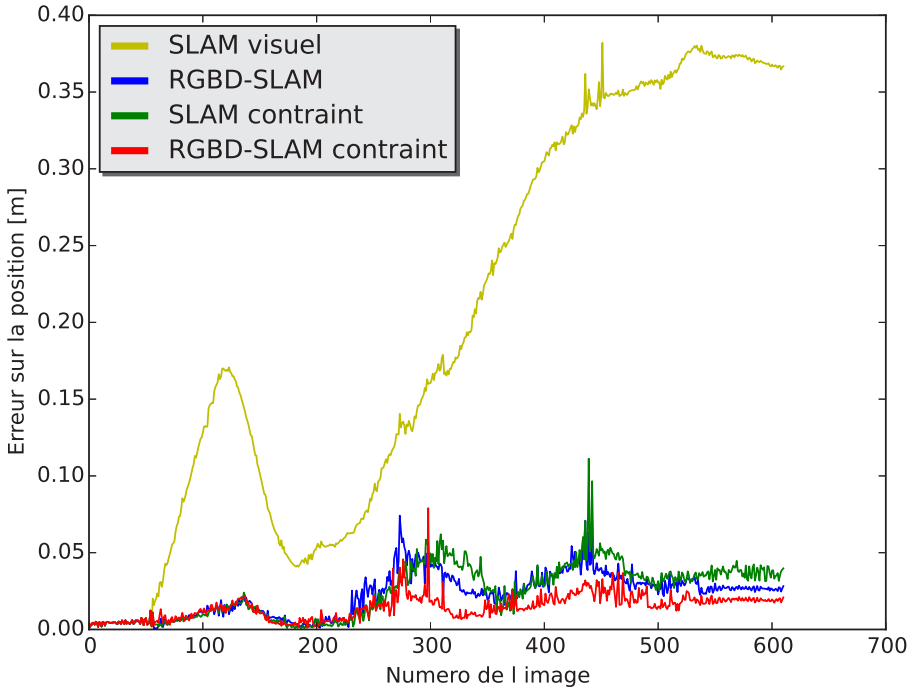


(c) Electrical

FIGURE 6.6 – **Séquences réelles "Corbs".** Les reconstruction 3D réelles des scènes évaluées.

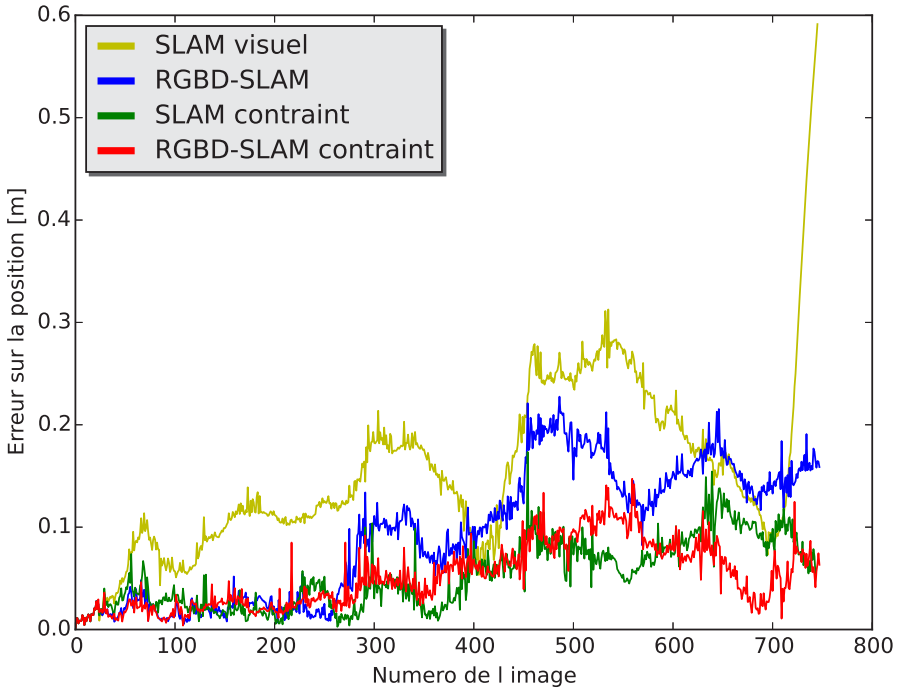


(a) Human



(b) Desk

FIGURE 6.7 – Évolution des erreurs (ATE) sur les séquences "Human" et "Desk".



(a) Electrical

FIGURE 6.8 – Évolution des erreurs (ATE) sur la séquence "Electrical".

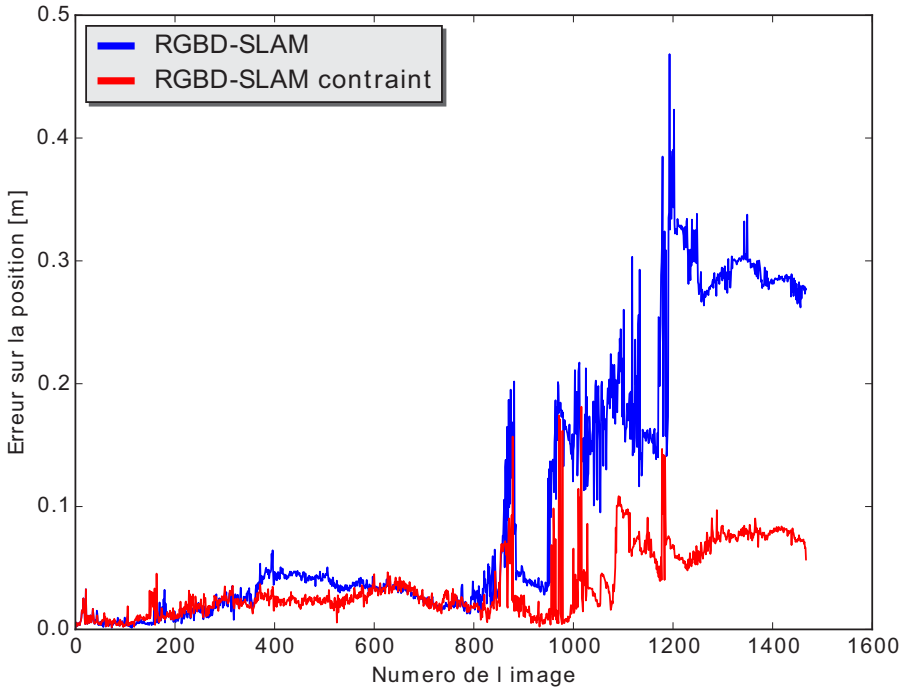
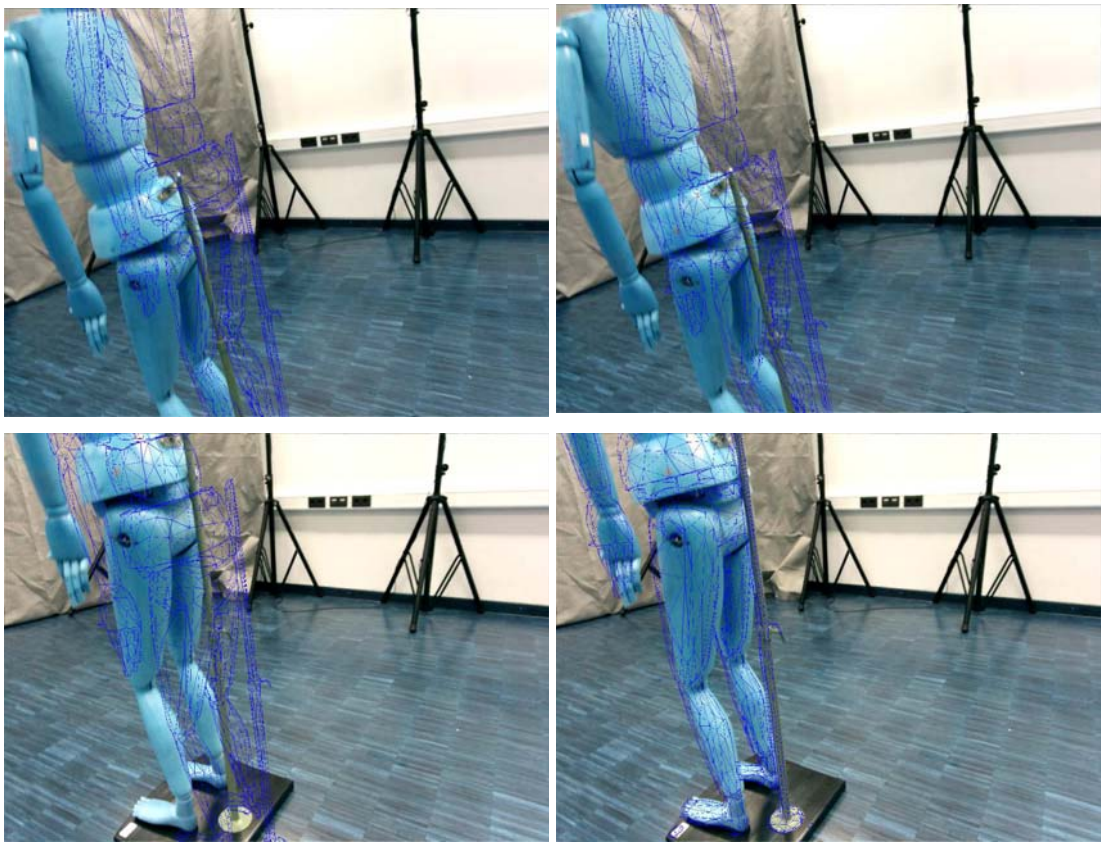


FIGURE 6.9 – **L'apport de la contrainte par plans sur le RGBD-SLAM.** L'évolution de l'erreur ATE sur la séquence "Human".

la dérive enregistrée par les deux algorithmes à l'image 971. Cette dérive est relativement importante (l'erreur de position est de ∼0.2m pour les deux algorithmes). Grâce à l'ajustement de faisceaux proposé dans le RGBD-SLAM Contraint, l'erreur de position diminue rapidement (image 1012) pour se stabiliser autour de 0.03m de moyenne, tandis que l'erreur du RGBD-SLAM diminue à 0.1m seulement. En dépit des mauvaises associations points/plans qui peuvent se produire à cause du cumul d'erreurs, combiner différents termes dans la fonction de coût de l'ajustement de faisceaux du RGBD-SLAM Contraint (comme présenté dans la section 5.3.2) apporte plus de robustesse et de précision.



(a) (b)

FIGURE 6.10 – **Illustration des erreurs de re-projection du modèle, issues du RGBD-SLAM et du RGBD-SLAM Contraint sur la séquence "Human", aux images 971 (première ligne) et 1012 (deuxième ligne).** (a) : Correction de la dérive par le RGBD-SLAM, (b) : Correction de la dérive par le RGBD-SLAM Contraint.

Sur les séquences "Desk" et "Electrical", la précision du RGBD-SLAM Contraint et du SLAM Contraint visuel est pratiquement similaire. Ces séquences assez texturées, justifient les performances similaires obtenues par les deux algorithmes. Par contre sur la séquence "Human", moins texturée que les deux précédentes, la précision du RGBD-SLAM Contraint est ∼10 fois supérieure à celle du SLAM Contraint visuel (Figure 6.11).

Si nous prenons l'exemple de l'image 1178, l'erreur de position absolue du SLAM Contraint visuel est de 0.2m et l'erreur de position absolue du RGBD-SLAM Contraint est de 0.14m. Ces erreurs élevées sont dues au mouvement saccadé de la caméra entre l'image 1176 et l'image 1178 (Figure 6.12.) qui provoque un flou dans les images et entraine des mauvaises associations

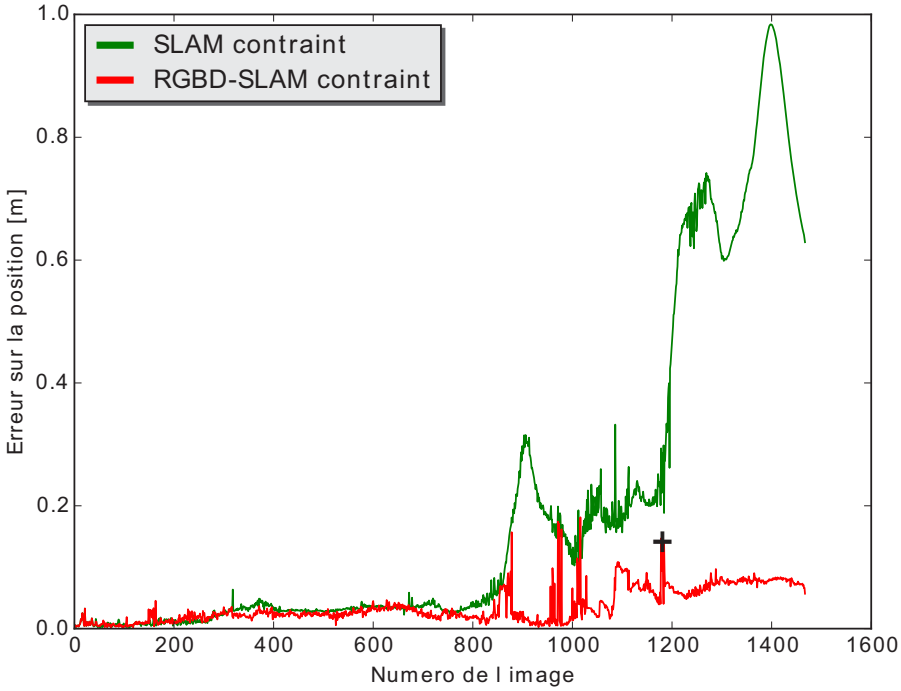


FIGURE 6.11 – Comparaison des erreurs de position absolues (ATE) du SLAM Contraint visuel et du RGBD-SLAM Contraint sur la séquence "Human".

2D/3D qui dégradent le calcul de poses. Néanmoins l'ajustement de faisceaux du RGBD-SLAM Contraint arrive à baisser cette erreur (l'erreur diminue à 0.07m à l'image 1190) tandis que le SLAM Contraint visuel ne corrige jamais cette erreur et continue à dériver. Ceci permet de démontrer la robustesse du RGBD-SLAM Contraint face aux mouvements rapides et brusques, particulièrement dans des scènes peu texturées.

(a) Image 1176 (b) Image 1178

FIGURE 6.12 – Exemple de mouvements saccadés de la caméra entre l'image 1176 et l'image 1178 de la séquence " Human".

6.2.2 Séquence réelle "couloir"

Pour confirmer l'apport de notre méthode sur la précision de la localisation, nous la testons sur notre propre séquence réelle. Cette séquence est enregistrée en utilisant le capteur *Kinect*V1, sur une trajectoire de 50m dans un couloir peu texturé. La largeur du couloir est de 2m. La vérité terrain n'est pas disponible pour cette séquence, néanmoins pendant l'acquisition, la caméra est maintenue au milieu du couloir pour suivre la trajectoire illustrée en Figure 6.13.

L'environnement de cette séquence est complexe, la scène est peu texturée, les conditions de luminosité sont très variables (zones sombres, zones très lumineuses, présence d'ombres, présence de spécularité...). De plus la caméra observe plusieurs endroits où l'information de profondeur n'est pas disponible (vitres, surfaces métalliques). Aussi, la portée du capteur est limitée à environ 5m. La Figure 6.14. présente quelques images couleurs de la scène observée et leur cartes de profondeur associées.

Cette séquence est enregistrée dans un bâtiment, par conséquent le modèle 3D nécessaire pour tester la contrainte par plans est construit à partir des plans 2D fournis par le constructeur. La Figure 6.15. illustre le modèle 3D obtenu.

Résultats : Sur cette séquence réelle, la vérité terrain n'est pas disponible. Les résultats seront donc qualitatifs. Nous présentons la trajectoire obtenue projetée sur le plan (x,y) et nous la comparons visuellement à la trajectoire suivie par la caméra présentée dans la Figure 6.13.

Les résultats obtenus sur cette séquence mettent en évidence le manque de robustesse du RGBD-SLAM qui utilise la fonction de coût avec la contrainte de profondeur seulement. La trajectoire obtenue avec cette méthode dérive de manière considérable. Après le premier virage, la localisation de la caméra est erronée et la dérive n'est jamais corrigée au cours de la trajectoire. Rajouter la contrainte par plans a permis de corriger cette dérive. En effet forcer les points à se déplacer seulement sur les plans auxquels ils sont associés, permet de pallier les problèmes liés aux mouvements brusques et au manque de profondeur à certains endroits.

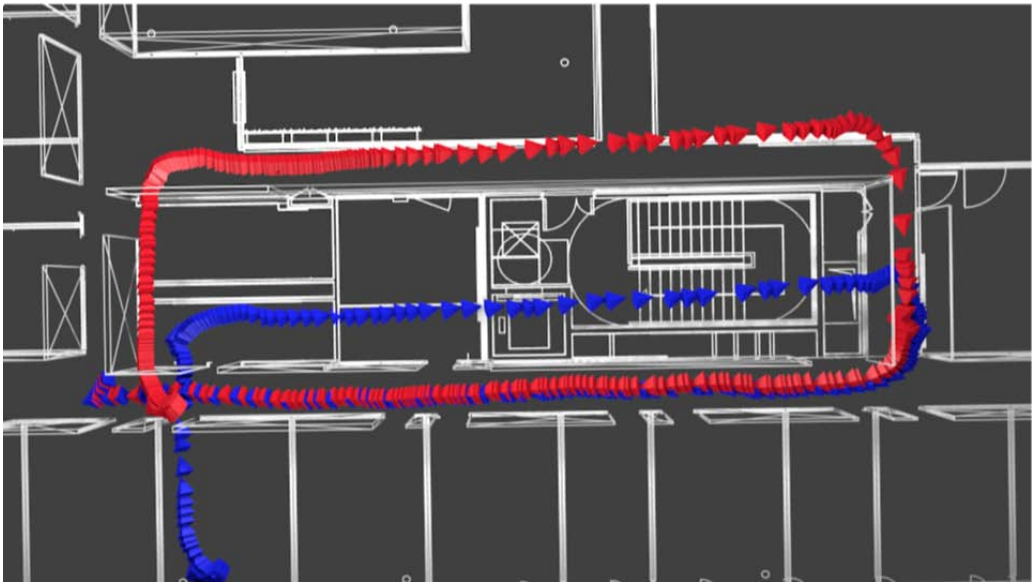
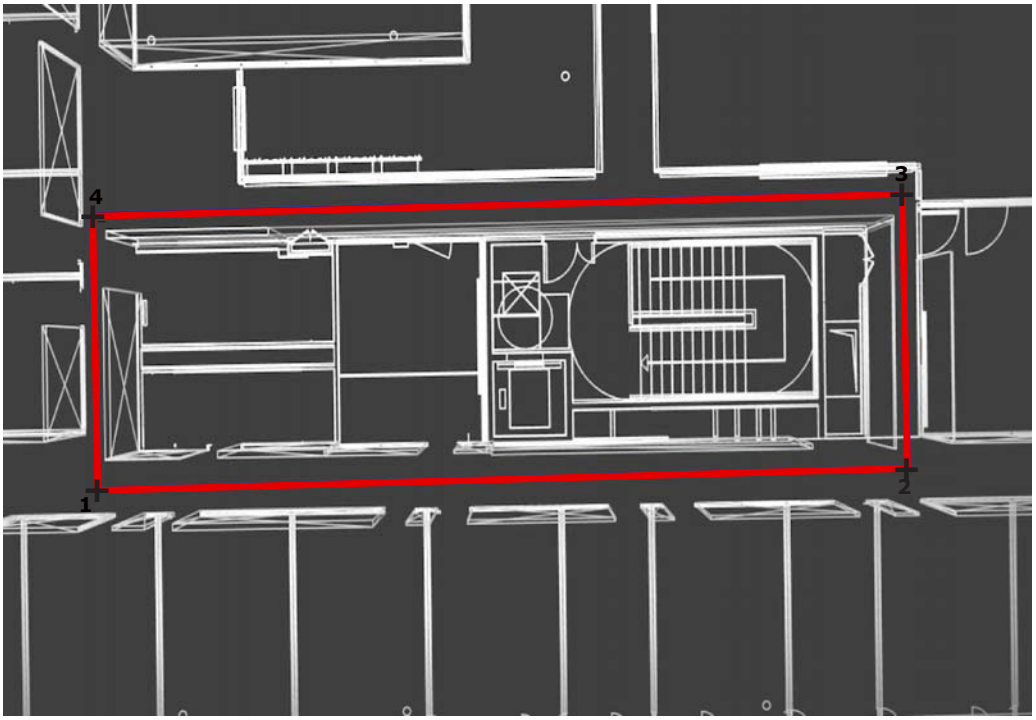
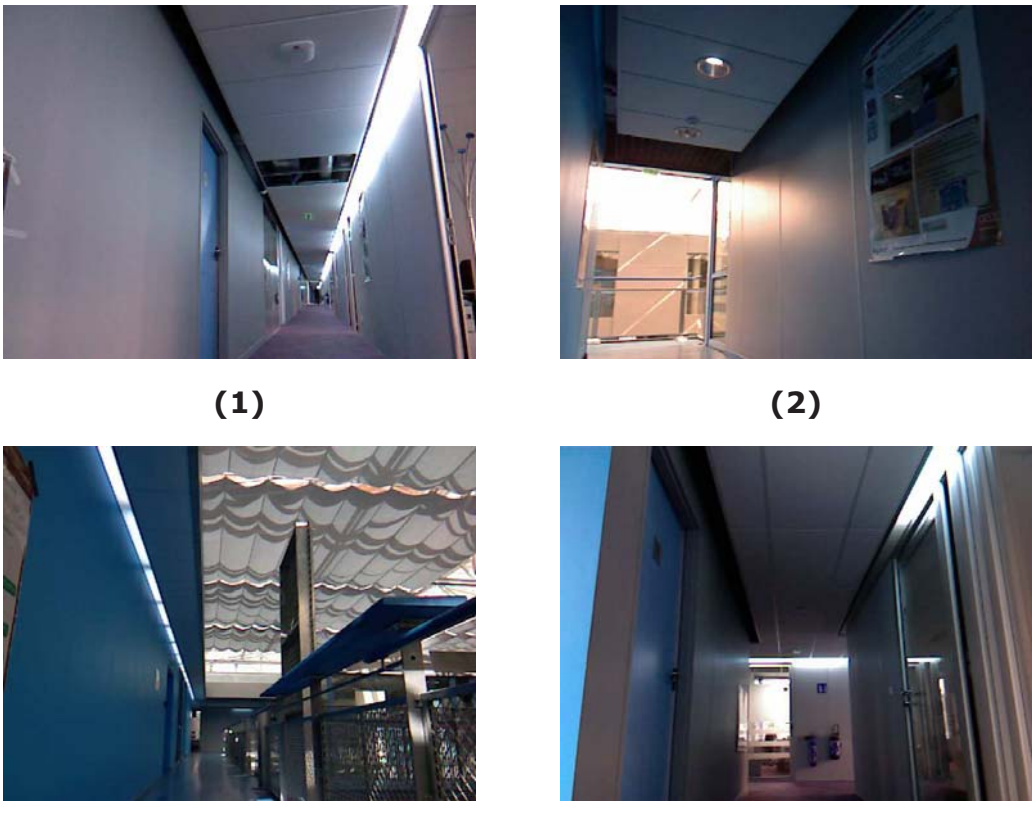


FIGURE 6.16 – **La trajectoire de la caméra obtenue sur la séquence "Couloir"**. La trajectoire en bleu est celle obtenue avec le RGBD-SLAM, la trajectoire en rouge est celle obtenue avec le RGBD-SLAM Contraint.



(a)



(b)

FIGURE 6.13 – **Séquence réelle "Couloir".** (a) : Illustration de la trajectoire (1-2-3-4), suivie par la caméra sur la séquence "Couloir". (b) : Les images RGB correspondant aux coins numérotés.

FIGURE 6.14 – **Séquence réelle "Couloir".** Illustration de la séquence réelle utilisée ; à gauche les images couleur et à droites les cartes de profondeur correspondantes.

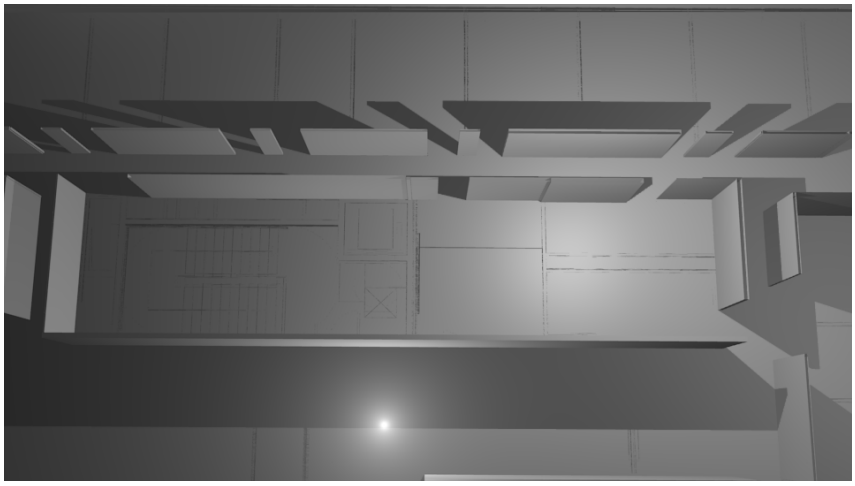


FIGURE 6.15 – Modèle 3D reconstruit à partir des plans 2D de la scène réelle "Couloir".

6.3 Conclusion

Dans ce chapitres, nous avons présenté les différentes expérimentations effectuées pour évaluer notre solution de RGBD-SLAM Contraint. Ces expérimentations ont été réalisées sur des séquences de synthèse et sur des séquences réelles tirées de bases de données de l'état de l'art. Elles démontrent les améliorations apportées par notre solution, notamment dans les environnements peu texturés. Cette technique a montré une meilleure robustesse au flou de bougé et aux mouvements brusques. Des évaluations sur notre propre séquence réelle, confirment l'intérêt de combiner une contrainte par plans et une contrainte sur les profondeurs des points dans l'ajustement de faisceaux, afin d'améliorer la localisation de la caméra.

Conclusion

Travaux réalisés

Les travaux présentés dans cette thèse concernent le problème de localisation par vision embarquée en milieu intérieur. L'objectif principal était d'exploiter un capteur RGBD plutôt qu'une simple caméra pour améliorer les performances d'un SLAM visuel.

De nombreux travaux traitant la problématique du SLAM en utilisant des capteurs RGBD ont été proposés ces dernières années. La majorité d'entre eux se focalisent sur la reconstruction d'une carte dense et nécessitent généralement des puissances de calcul GPU.

Dans l'objectif de proposer une méthode adaptée aux applications embarquées sur des structures légères telles des drones, des lunettes de réalité augmentée ou des petits robots, qui disposent de ressources limitées en capacité calculatoire et en capacités de stockage, nous nous sommes orientés vers une solution de type RGBD-SLAM éparse.

La première contribution de cette thèse concerne le développement d'un RGBD-SLAM reposant sur la modification d'un SLAM monoculaire pour y intégrer des informations de profondeur issues d'un capteur RGBD.

Cette première contribution présentée dans le chapitre 3, a consisté à modifier trois étapes d'un algorithme SLAM visuel monoculaire proposé par Mouragnon et al. (2006b), pour prendre en compte l'information de profondeur. Le principal apport porte sur l'intégration de la mesure de profondeur comme contrainte additionnelle dans le processus d'optimisation par ajustement de faisceaux. Nous avons montré comment combiner dans la même fonction de coût une erreur de re-projection des primitives en fonction de leur profondeur, et une erreur de re-projection des primitives sans information de profondeur associée. Cette solution que nous avons appelée RGBD-SLAM, a été évaluée sur différentes séquences de synthèses et réelles et a été comparée aux récentes méthodes RGBD de l'état de l'art. Les différents résultats ont montré que notre solution a permis de réduire considérablement les erreurs de dérive du SLAM visuel, tout en préservant les structures creuses des matrices engendrées par l'ajustement de faisceaux.

Notons également que l'ajout de cette contrainte supplémentaire, en aucun cas, ne détériore les performances calculatoires de l'algorithme initial ni son fonctionnement en mode monoculaire en cas d'absence d'information de profondeur. Cette solution permet de prendre en compte les points à longues distances en mode SLAM visuel et les points à courte distance, donc disposant d'une profondeur, en mode RGBD-SLAM.

La deuxième contribution de cette thèse est d'exploiter, en complément des informations de profondeur, une information a priori sur la structure de la scène, à savoir un modèle 3D partiel de la scène, par exemple un plan 2,5D obtenu à partir du plan 2D d'un bâtiment. Ces travaux sont inspirés de ceux de Tamaazousti et al. (2011a). Dans ces travaux, les auteurs proposent un SLAM Contraint par un modèle 3D de l'objet. L'idée a été de généraliser ce principe au cas d'un RGBD SLAM. Le concept consiste à rajouter au sein du même ajustement de faisceaux que précédemment, une contrainte additionnelle forçant les points 3D à appartenir aux plans 3D connus de la scène. Nous avons utilisé ce principe en y intégrant une contrainte sur les profondeurs des points. Cette contrainte est rajoutée dans l'ajustement de faisceaux sous forme d'une erreur sur les profondeurs des primitives. Dans le processus d'optimisation plusieurs hypothèses sur les caractéristiques d'un point 3D sont testées : son appartenance ou non à un environnement connu de la scène, la disponibilité ou l'absence de sa mesure de profondeur. Selon la caractéristique du point, ce dernier participe au processus d'optimisation dans une ou plusieurs erreurs spécifiques. Ces différentes erreurs sont combinées dans la même fonction de coût.

Une des difficultés de cette solution fut dans la manière de combiner des erreurs exprimées en mètres (l'erreur sur la profondeur) et des erreurs exprimées en pixels (erreurs de re-projection 2D). La solution adoptée est d'appliquer un facteur de pondération pour les erreurs sur les profondeurs. Ce facteur est lié aux incertitudes sur les profondeurs. Ajouter à cela, un estimateur robuste a été appliqué sur chaque erreur indépendamment, pour gérer les valeurs aberrantes.

Nous avons fourni une évaluation exhaustive, tant sur le plan quantitatif que qualitatif, sur des bases de données communes à l'état de l'art et sur nos propres séquences, démontrant la capacité du système à produire des trajectoires précises même dans les milieux difficiles.

Perspectives d'amélioration

Les différentes solutions proposées dans le cadre de cette thèse ont permis d'améliorer significativement la localisation du SLAM. Les résultats obtenus en intégrant les contraintes sur les profondeurs des points et les contraintes sur les plans de la scène, sont encourageants. Les études et expérimentations réalisées nous ont également permis de mettre en évidence certaines pistes d'amélioration pour notre algorithme.

Localisation dans les zones très peu texturées

L'absence de texture dans l'environnement est un challenge pour notre algorithme. Si le nombre de primitives détectées dans l'image est inférieur à un certain seuil, le calcul de poses échoue ou peut être très erroné. Afin de résoudre ce problème nous avons pensé à mettre en place un autre calcul de poses qui serait une alternative dans le cas d'échec du calcul de poses existant. Ce calcul de pose sera basé sur l'estimation de la déformation d'images, appelé aussi warping d'images (Newcombe et al. (2011b), Kerl et al. (2013a)). Elle est basée sur l'hypothèse qu'un même point observé par deux caméras, produit la même intensité lumineuse sur les deux images. Ainsi, tous les points de la première image sont rétro-projetés en 3D grâce à leur mesure de profondeur fournie par un capteur RGB-D, la fonction de warping estime le mouvement de la caméra qui transforme les points dans la deuxième image en minimisant une erreur photométrique, après la projection de ces points sur cette dernière.

Améliorer la localisation en utilisant une fermeture de boucles

L'utilisation d'une approche incrémentale basée sur un SLAM visuel ajoute inévitablement une petite erreur de localisation à chaque étape. Nous avons optimisé ce SLAM en rajoutant plusieurs contraintes dans l'ajustement de faisceaux afin de réduire cette erreur au maximum. Cependant, cette erreur s'accumule et peut devenir observable sur les longues trajectoires. La résolution de ce problème peut se faire par l'utilisation d'un algorithme de fermeture de boucles à la fin de l'ajustement de faisceaux. Lorsque la caméra repasse par un endroit déjà visité, la dérive cumulée peut être estimée et corrigée.
Ce problème de fermeture de boucles a largement été étudié dans les algorithmes SLAM et plusieurs travaux de l'état de l'art proposent de le résoudre (Henry et al. (2013), Kummerle et al. (2011), Whelan et al. (2013a)).

Généralisation à d'autres capteurs

La solution proposée, RGBD-SLAM ou RGBD-SLAM Contraint, a été testée avec des capteurs RGBD (kinect V1 et V2) ce qui restreint son application à la localisation en milieu intérieur. Cependant, elle peut être utilisée avec un autre capteur 3D (*e.g.* lidar) couplé à une camera RGB. D'autre part, le principe proposé peut directement se transposer au cas d'un capteur stéréo dans la mesure où ce dernier fournit une carte de profondeur. L'utilisation de ces capteurs permettrait d'étendre la solution à la localisation en milieu extérieur.

Reconstruction dense de l'environnent

Cette thèse n'a pas traité le problème de la reconstruction dense. Dans la mesure où les poses du capteur estimées par nos algorithmes, sont précises, une simple solution d'agrégation de cartes de profondeur mériterait d'être testée.

ANNEXE A

Notions de bases supplémentaires

A.1 Géométrie épipolaire

La *géométrie épipolaire* décrit les contraintes reliant les observations d'une même scène observée par deux caméras, notées $\mathcal{C}_1$ et $\mathcal{C}_2$ (figure A.1). Ces contraintes sont directement liées au déplacement relatif (également appelé positionnement relatif) entre les deux caméras mais sont totalement indépendantes de la structure de la scène. Toutefois, il est important de rappeler que dans le cas du déplacement d'une caméra (*i.e.* dans le cas de la configuration temporelle), la géométrie épipolaire est uniquement vérifiée si la scène observée est rigide.

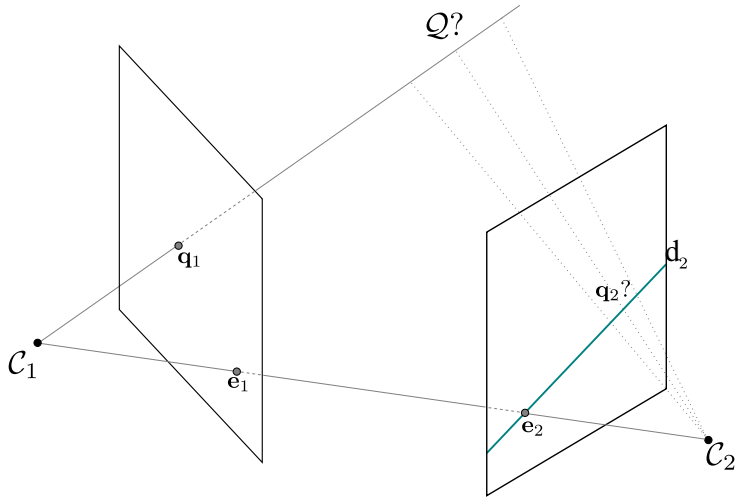


FIGURE A.1 – **Géométrie épipolaire.** La géométrie épipolaire définit des contraintes géométriques entre les différentes observations d'un même point de l'espace.

Matrice fondamentale La *matrice fondamentale* exprime la relation épipolaire dans le cas où les paramètres internes des caméras sont inconnus. Ainsi, pour un point $\mathbf{q}_1$ de l'image de

la première caméra, il est possible de calculer la droite $\mathbf{d}_2$ sur laquelle se situe l'observation correspondante dans la deuxième caméra (figure A.1) :

$$\mathbf{d}_2 \sim \mathsf{F}\tilde{\mathbf{q}}_1. \tag{A.1}$$

La droite $\mathbf{d}_2$ est appelée *droite épipolaire* associée à $\mathbf{q}_1$. De plus, si deux observations $\mathbf{q}_1$ et $\mathbf{q}_2$ correspondent au même point de l'espace, elles vérifient :

$$\tilde{\mathbf{q}}_2{}^{\mathsf{T}}\mathsf{F}\tilde{\mathbf{q}}_1 = 0. \tag{A.2}$$

Cette relation permet d'estimer la matrice fondamentale à partir d'associations 2D entre deux images. En pratique, F peut se calculer à l'aide de 8 points (Hartley (1997)) ou à partir de 7 points sous certaines hypothèses (Torr and Murray (1997)).

Dans chacune des images, un point joue un rôle particulier. Il s'agit des deux épipôles $\mathbf{e}_1$ et $\mathbf{e}_2$. Ils correspondent à la projection dans l'image du centre optique de l'autre caméra. Les épipôles présentent deux caractéristiques intéressantes. Tout d'abord, elles définissent le noyau de F : $\mathsf{F}\mathbf{e}_i = 0, \forall i \in \{1, 2\}$. De plus, les épipôles correspondent aux points d'intersection de toutes les droites épipolaires de chacune des images.

Matrice essentielle La *matrice essentielle* E peut être vue comme le cas particulier de la matrice fondamentale dans le cas où le calibrage des caméras (K_1 et K_2) est connu, ce qui est le cas qui nous intéresse en particulier. La relation entre matrice essentielle et matrice fondamentale est la suivante :

$$\mathsf{E} \sim \mathsf{K}_2{}^{\mathsf{T}}\mathsf{F}\mathsf{K}_1. \tag{A.3}$$

L'équation A.2 devient dans ce cas :

$$\tilde{\mathbf{q}}_2{}^{\mathsf{T}} \left(\mathsf{K}_2{}^{-\mathsf{T}}\mathsf{E}\mathsf{K}_1{}^{-1}\right) \tilde{\mathbf{q}}_1 = 0, \tag{A.4}$$

où $\mathsf{K}_2{}^{-\mathsf{T}}$ est la transposée inverse de K_2. Pour estimer la matrice essentielle, Nistér (2004) a proposé un algorithme efficace appelé *algorithme des 5 points*.

Relation entre matrice essentielle et déplacement relatif En fonction des cas d'étude, la matrice essentielle peut avoir différentes utilisations. En effet, il existe une relation qui lie la matrice essentielle du couple de caméras $(\mathcal{C}_1, \mathcal{C}_2)$ au déplacement relatif entre ces caméras. Le déplacement relatif est défini par le couple $(\mathcal{R}_{1\to2}, \mathbf{t}_{1\to2})$. La relation entre E, $\mathcal{R}_{1\to2}$ et $\mathbf{t}_{1\to2}$ s'écrit :

$$\mathsf{E} = [\mathbf{t}_{1\to2}]_\times \mathcal{R}_{1\to2}, \tag{A.5}$$

où $[\mathbf{t}]_\times$ est la matrice antisymétrique construite à partir du vecteur $\mathbf{t}$, à savoir :

$$[\mathbf{t}]_\times = \begin{pmatrix} 0 & -\mathbf{t}_z & \mathbf{t}_y \\ \mathbf{t}_z & 0 & -\mathbf{t}_x \\ -\mathbf{t}_y & \mathbf{t}_x & 0 \end{pmatrix}. \tag{A.6}$$

Dès lors, deux cas de figure sont possibles. Si le déplacement entre les caméras est connu, la matrice essentielle permet de réduire la recherche du point d'intérêt correspondant à 1 dimension (le long de la droite épipolaire). Dans le cas contraire, une estimation de la matrice essentielle (grâce à l'appariement d'au moins 5 points) permet de retrouver le déplacement relatif entre les caméras.

ANNEXE B

Ajustement de faisceaux

Nous présentons dans cette annexe les calculs des dérivées analytiques engendrées par le processus d'optimisation. Dans un premier lieu nous détaillerons le calcul des dérivées associées à la fonction de coût du SLAM classique par la suite nous exposerons la nouvelle fonction de coût du SLAM RGBD et nous détaillerons les changements liés à l'intégration de l'information de profondeur dans le processus d'optimisation.

B.1 Ajustement de faisceaux basé sur la fonction de coût standard

Comme nous l'avons détaillé dans la section 1.2.4, la fonction de coût utilisée dans l'ajustement de faisceaux consiste à minimiser l'erreur de re-projection standard. Rappelons que la projection du point 3D $\tilde{\mathcal{Q}} = (X, Y, Z, 1)^T$ en un point 2D de coordonnées homogènes $\tilde{p} = (p_1, p_2, p_3)^T$ par la caméra $\mathcal{C}$, définie par sa rotation $\mathcal{R}$, sa translation $\mathbf{t}$ et sa matrice de paramètres intrinsèques K, s'écrit :

$$\begin{pmatrix} p_1/p_3 \\ p_2/p_3 \end{pmatrix} = \pi(\mathsf{K}\mathcal{R}^T \left[\mathsf{I}_{3\times 3}\right| - \mathbf{t}] \, \tilde{\mathcal{Q}}). \tag{B.1}$$

Ainsi l'erreur de re-projection pour l'observation 2D $\mathbf{q} = (q_1, q_2)^T$ associé au point 3D $\mathcal{Q}$ est définie par :

$$\mathbf{f} = \begin{pmatrix} q_1 - p_1/p_3 \\ q_2 - p_2/p_3 \end{pmatrix} = \mathbf{q} - \pi(\mathsf{K}\mathcal{R}^T \left[\mathsf{I}_{3\times 3}\right| - \mathbf{t}] \, \tilde{\mathcal{Q}}). \tag{B.2}$$

La dérivée de $\mathbf{f}$ par rapport aux paramètres de caméra et les paramètres du point 3D est donnée par

$$d\left(\mathbf{f}\right) = \left(\frac{\partial \mathbf{f}}{\partial \mathcal{R}} | \frac{\partial \mathbf{f}}{\partial \mathbf{t}} | \frac{\partial \mathbf{f}}{\partial \mathcal{Q}} \right) = d\left(\pi\right) \times d\left(\mathsf{K}\mathcal{R}^T \left[\mathsf{I}_{3\times 3}\right| - \mathbf{t}] \, \tilde{\mathcal{Q}}) \right). \tag{B.3}$$

π représente la matrice qui permet de passer des coordonnées homogènes aux coordonnées non homogènes d'une observation 2D. Sa dérivée est donnée par :

$$d(\pi) = \begin{pmatrix} \frac{1}{p_3} & 0 & \frac{-p_1}{p_3^2} \\ 0 & \frac{1}{p_3} & \frac{-p_2}{p_3^2} \end{pmatrix}. \tag{B.4}$$

Il reste maintenant à calculer les dérivées de $\mathsf{K}\mathcal{R}^T \left[\mathsf{I}_{3\times3}| - \mathbf{t}\right] \tilde{\mathcal{Q}}$ par rapport aux paramètres de la caméra $(\mathcal{R}|\mathbf{t})$ et les paramètres du point 3D $\mathcal{Q}$.

Dérivées par rapport aux paramètres de la caméra.

- Dérivées par rapport aux paramètres de la rotation. Ces dernières sont calculées en utilisant un repère relatif au voisinage de la rotation courante $\mathcal{R}_0$, telle que $\mathcal{R}_0(\omega) = \mathcal{R}(\omega)\mathcal{R}_0$, avec $\omega = (\omega_1, \omega_2, \omega_3)^T$ contient les faibles angles d'Euler autour de $\mathcal{R}_0$ et :

$$\mathcal{R}(\omega) = \begin{pmatrix} cos(\omega_1) & -sin(\omega_1) & 0 \\ sin(\omega_1) & cos(\omega_1) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos(\omega_2) & 0 & sin(\omega_2) \\ 0 & 1 & 0 \\ -sin(\omega_2) & 0 & cos(\omega_2) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\omega_3) & -sin(\omega_3) \\ 0 & sin(\omega_3) & cos(\omega_3) \end{pmatrix}. \tag{B.5}$$

Au voisinage de $\mathcal{R}_0$, $\mathcal{R}(\omega)$ peut s'écrire sous la forme suivante :

$$\mathcal{R}(\omega) = \mathsf{I}_{3\times3} + [\omega]_\times + o(\omega) \approx \begin{pmatrix} 1 & -\omega_3 & \omega_2 \\ \omega_3 & 1 & -\omega_1 \\ -\omega_2 & \omega_1 & 1 \end{pmatrix}. \tag{B.6}$$

Ainsi, à chaque itération de l'ajustement de faisceaux, les dérivées par rapport aux angles Euler α, β et γ correspondant à la matrice de rotation $\mathcal{R}$ dans le repère global sont remplacées par les dérivées par rapport aux angles ω_1, ω_2 et ω_3 dans le repère local lié à la matrice de rotation $\mathcal{R}_0$. Par conséquent :

$$\begin{aligned} \frac{\partial \mathbf{f}}{\partial \omega} &= d(\pi) \times \frac{\partial}{\partial \omega} \left(\mathsf{K}\mathcal{R}_0^T(\omega) \left[\mathsf{I}_{3\times3}| - \mathbf{t}\right] \tilde{\mathcal{Q}} \right) \\ &= d(\pi) \times \mathsf{K} \times \frac{\partial}{\partial \omega} \left(\left(\mathcal{R}(\omega)\mathcal{R}_0\right)^T \right) \left[\mathsf{I}_{3\times3}| - \mathbf{t}\right] \tilde{\mathcal{Q}} \\ &= d(\pi) \times \mathsf{K}\mathcal{R}_0^T \times -\frac{\partial}{\partial \omega} \left(\mathcal{R}^T(\omega) \right) \left[\mathsf{I}_{3\times3}| - \mathbf{t}\right] \tilde{\mathcal{Q}} \end{aligned} \tag{B.7}$$

Avec l'hypothèse des faibles angles, chaque dérivée peut s'écrire comme suit :

$$\frac{\partial \mathbf{f}}{\partial \omega} = d(\pi) \times \mathsf{K}\mathcal{R}_0^T \times -\frac{\partial \left[\omega\right]_\times}{\partial \omega_j} \left[\mathsf{I}_{3\times3}| - \mathbf{t}\right] \tilde{\mathcal{Q}} \tag{B.8}$$

où les $\frac{\partial [\omega]_\times}{\partial \omega_j}$, $j \in \{1, 2, 3\}$ sont :

$$\frac{\partial \left[\omega\right]_\times}{\partial \omega_1} \approx \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \tag{B.9}$$

$$\frac{\partial \left[\omega\right]_\times}{\partial \omega_2} \approx \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \tag{B.10}$$

$$\frac{\partial \left[\omega\right]_{\times}}{\partial \omega_3} \approx \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{B.11}$$

- Dérivées par rapport aux paramètres de la translation. Elles sont données par :

$$\begin{aligned} \frac{\partial \mathbf{f}}{\partial \mathbf{t}} &= d(\pi) \times \frac{\partial}{\partial \mathbf{t}} \left(\mathsf{K}\mathcal{R}^T \left[\mathsf{I}_{3\times 3} | - \mathbf{t} \right] \tilde{\mathcal{Q}} \right) \\ &= d(\pi) \times \mathsf{K}\mathcal{R}^T \times \frac{\partial}{\partial \mathbf{t}} \left(\left[\mathsf{0}_{3\times 3} | - \mathbf{t} \right] \mathcal{Q} \right) \\ &= -1 \times d(\pi) \times \mathsf{K}\mathcal{R}^T. \end{aligned} \tag{B.12}$$

Dérivées par rapport aux paramètres des points 3D. Ces dernières s'obtiennent immédiatement en dérivant par rapport à $\mathcal{Q}$:

$$\begin{aligned} \frac{\partial \mathbf{f}}{\partial \mathcal{Q}} &= d(\pi) \times \frac{\partial}{\partial \mathcal{Q}} \left(\mathsf{K}\mathcal{R}^T \left[\mathsf{I}_{3\times 3} | - \mathbf{t} \right] \mathcal{Q} \right) \\ &= d(\pi) \times \mathsf{K}\mathcal{R}^T \left[\mathsf{I}_{3\times 3} | - \mathbf{t} \right] \end{aligned} \tag{B.13}$$

Construction du système creux. La fonction de coût associée à l'erreur de re-projection standard est calculée comme suit :

$$f\left(\{\mathcal{R}_j, \mathbf{t}_j\}_{j=1}^{\mathcal{N}_c}, \left\{ \mathcal{Q}^i \right\}_{i=1}^{\mathcal{N}_p} \right) = \sum_{i=1}^{i=N_p} \sum_{j \in \mathcal{A}_i} \|\mathbf{f}_{i,j}\|^2, \tag{B.14}$$

Avec $\mathbf{f}_{i,j}$ est l'erreur de re-projection correspondante au $i^{\text{ème}}$ point 3D observé par la $j^{\text{ème}}$ caméra.

Dans ce qui suit, nous souhaitons optimiser cette fonction de coût en utilisant l'algorithme de Levenberg Marquardt exploitant la nature creuse de la Jacobienne et la Hessienne.

Le calcul des dérivées de chaque $\mathbf{f}_{i,j}$ permet de construire la matrice Jacobienne J et ainsi de construire le système des équations normales. Cependant, l'évaluation complète de la matrice Jacobienne n'est pas indispensable pour la résolution du système. Il est possible de calculer directement la Hessienne $\mathsf{H} \approx \mathsf{J}^T\mathsf{J}$ ainsi que le vecteur gradient $\mathbf{g} = \mathsf{J}^T\mathbf{r} = (\mathbf{g}_{caméra}^T, \mathbf{g}_{point}^T)$, avec $\mathbf{r}$ est le vecteur des résidus concaténant les erreurs de re-projection $\mathbf{f}_{i,j}$. Le système à résoudre est représenté dans la figure B.1. Les matrices U et V sont diagonales par blocs où chaque bloc a une dimension de 6×6 et 3×3 respectivement. Quant à la matrice W, elle contient des blocs de dimension 6×3. Ces différentes matrices sont construites d'une façon incrémentale. Au départ, chacun des blocs $\mathsf{U}_j, \mathsf{V}_i$ et W_{ij} est initialisé à zéro. Ensuite, ils sont mis à jour pour chaque observation (i, j) :

- $$\mathsf{U}_j = \mathsf{U}_j + \left(\frac{\partial \mathbf{f}_{i,j}}{\partial \mathcal{R}_j} | \frac{\partial \mathbf{f}_{i,j}}{\partial \mathbf{t}_j} \right)^T \left(\frac{\partial \mathbf{f}_{i,j}}{\partial \mathcal{R}_j} | \frac{\partial \mathbf{f}_{i,j}}{\partial \mathbf{t}_j} \right). \tag{B.15}$$

- $$\mathsf{V}_i = \mathsf{V}_i + \left(\frac{\partial \mathbf{f}_{i,j}}{\partial \mathcal{Q}^i} \right)^T \left(\frac{\partial \mathbf{f}_{i,j}}{\partial \mathcal{Q}^i} \right). \tag{B.16}$$

- $$\mathsf{W}_{i,j} = \left(\frac{\partial \mathbf{f}_{i,j}}{\partial \mathcal{R}_j} | \frac{\partial \mathbf{f}_{i,j}}{\partial \mathbf{t}_j} \right)^T \left(\frac{\partial \mathbf{f}_{i,j}}{\partial \mathcal{Q}^i} \right). \tag{B.17}$$

- $$\mathbf{g}_{caméra}[j] = \mathbf{g}_{caméra}[j] + \left(\frac{\partial \mathbf{f}_{i,j}}{\partial \mathcal{R}_j} | \frac{\partial \mathbf{f}_{i,j}}{\partial \mathbf{t}_j}\right)^T \times \mathbf{f}_{i,j}. \tag{B.18}$$
- $$\mathbf{g}_{point}[i] = \mathbf{g}_{point}[i] + \left(\frac{\partial \mathbf{f}_{i,j}}{\partial \mathcal{Q}^i}\right)^T \times \mathbf{f}_{i,j}. \tag{B.19}$$

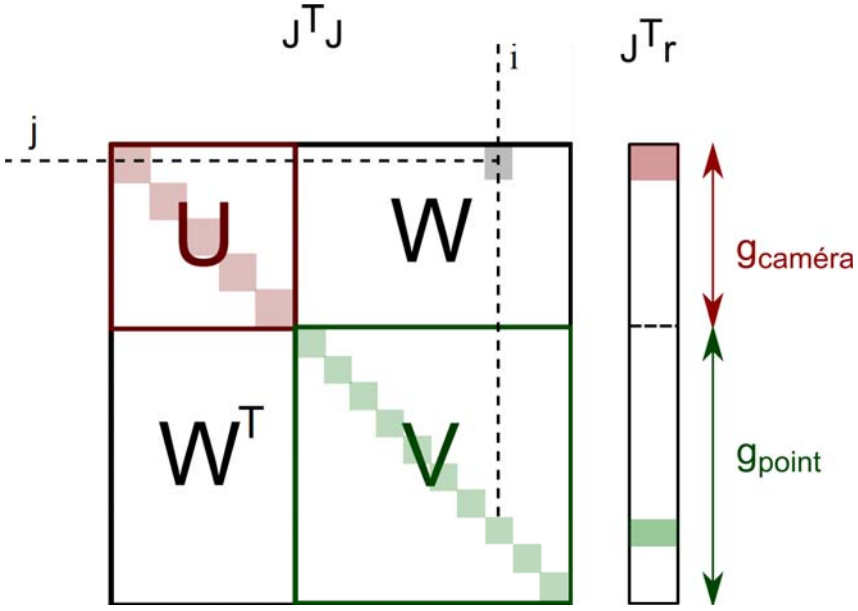


FIGURE B.1 – **Structure de la matrice Hessienne de l'ajustement de faisceaux.**

Le système obtenu a ainsi la forme suivante :

$$\begin{gathered} \mathsf{H}\delta = \mathbf{g} \\ \begin{pmatrix} \mathsf{U} & \mathsf{W} \\ \mathsf{W}^T & \mathsf{V} \end{pmatrix} \begin{pmatrix} \delta_{caméra} \\ \delta_{point} \end{pmatrix} = \begin{pmatrix} \mathbf{g}_{caméra} \\ \mathbf{g}_{point} \end{pmatrix}. \end{gathered} \tag{B.20}$$

A l'aide du complément de Schur, ce système peut être résolu en deux étapes :

1. Le calcul des incréments δ_{camera} à appliquer aux paramètres extrinsèques de la caméra par résolution du système linéaire suivant :
$$\left(\mathsf{U} - \mathsf{W}\mathsf{V}^{-1}\mathsf{W}^T\right)\delta_{camera} = \mathbf{g}_{camera} - \mathsf{W}\mathsf{V}^{-1}\mathbf{g}_{point}. \tag{B.21}$$
2. Le calcul direct des incréments δ_{point} applicables aux points 3D :
$$\delta_{point} = \mathsf{V}^{-1}\left(\mathbf{g}_{point} - \mathsf{W}^T\delta_{camera}\right). \tag{B.22}$$

Pour l'ajustement de faisceaux local, les paramètres optimisés sont : N_c poses de caméra et N_p points 3D visibles dans les N images correspondantes. Cela signifie que les matrice U, V et W sont réduites. Elles sont respectivement de taille $6N_c \times 6N_c$, $3N_p \times 3N_p$ et $6N_c \times 3N_p$.

Notons qu'il est possible d'utiliser un estimateur robuste au niveau de la fonction de coût afin d'assurer plus de robustesse face aux données aberrantes. Son utilisation ne change pas la structure du système. Seule une matrice de poids apparait au niveau de la Hessienne et du vecteur gradient. Par conséquent, le même principe est utilisé pour la résolution. Afin de ne pas surcharger les notions dans la suite de cette annexe, nous présenterons les fonctions de coût sans estimateur robuste.

B.2 Ajustement de faisceaux contraint par l'information de profondeur

Comme nous l'avons vu dans le chapitre 3, la principale modification apporté au SLAM se trouve dans l'ajustement de faisceaux. Le but étant d'intégrer la mesure de profondeur comme contrainte additionnelle tout en prenant en compte les contraintes multi-vues.
L'erreur de reprojection à minimiser est la suivante :

$$\mathbf{g} = \begin{pmatrix} q_1 - p_1/p_3 \\ q_2 - p_2/p_3 \end{pmatrix} = \mathbf{q}_j - \pi\left(\mathsf{K}\mathsf{P}_j\mathsf{P}_i^{-1}\pi^{-1}\left(q_i,\ \mathrm{d}\right)\right). \tag{B.23}$$

Avec $\mathsf{P}_i = \mathcal{R}_i{}^T\left[\mathsf{I}_{3\times3}|-\mathbf{t}_i\right]$: la pose de la caméra générant le point 3D $\mathcal{Q}_i$, $\mathsf{P}_j = \mathcal{R}_j{}^T\left[\mathsf{I}_{3\times3}|-\mathbf{t}_j\right]$: la pose de la caméra observant le point $\mathcal{Q}_i$.

Contrairement à l'ajustement de faisceaux du SLAM classique, les paramètres des points 3D ne sont pas optimisés explicitement dans le processus, ils seront reconstruits par triangulation après l'optimisation des poses des caméras. De ce fait nous aurons à calculer les dérivées par rapports aux paramètres des caméras seulement.
La dérivée de la fonction g par rapport aux paramètres des deux caméras prises en compte est donc donnée par :

$$d\left(\mathbf{g}\right) = \left(\frac{\partial\mathbf{g}}{\partial\mathcal{R}_j}|\frac{\partial\mathbf{g}}{\partial\mathbf{t}_j}|\frac{\partial\mathbf{g}}{\partial\mathcal{R}_i}|\frac{\partial\mathbf{g}}{\partial\mathbf{t}_i}|\right) = d\left(\pi\right)\times d\left(\mathsf{K}\mathsf{P}_j\mathsf{P}_i^{-1}\pi^{-1}\left(q_i,\mathrm{d}\right)\right) \tag{B.24}$$

π représente la matrice qui permet de passer des coordonnées homogènes aux coordonnées non homogènes d'une observation 2D. Sa dérivée est donnée par :

$$d\left(\pi\right) = \begin{pmatrix} \frac{1}{p_3} & 0 & \frac{-p_1}{p_3^2} \\ 0 & \frac{1}{p_3} & \frac{-p_2}{p_3^2} \end{pmatrix}. \tag{B.25}$$

- **Dérivées par rapport aux paramètres de la caméra observant le point 3D**

Notons $\mathcal{Q}_i$ le point 3D généré par la caméra i, exprimé dans le repère monde tel que :

$$\mathcal{Q}_i = \mathsf{P}_i^{-1}\pi^{-1}\left(q_i,\ \mathrm{d}\right) \tag{B.26}$$

L'équation B.24 va s'écrire comme suit :

$$d\left(\mathbf{g}\right) = \left(\frac{\partial\mathbf{g}}{\partial\mathcal{R}_j}|\frac{\partial\mathbf{g}}{\partial\mathbf{t}_j}|\frac{\partial\mathbf{g}}{\partial\mathcal{R}_i}|\frac{\partial\mathbf{g}}{\partial\mathbf{t}_i}|\right) = d\left(\pi\right)\times d\left(\mathsf{K}\mathsf{P}_j\tilde{\mathcal{Q}}_i\right) \tag{B.27}$$

Avec $\mathsf{P}_j = \mathcal{R}_j{}^T\left[\mathsf{I}_{3\times3}|-\mathbf{t}_j\right]$.
Si on remplace P_j par son expression dans l'équation B.27 nous obtenons l'équation suivante :

$$d\left(\mathbf{g}\right) = \left(\frac{\partial\mathbf{g}}{\partial\mathcal{R}_j}|\frac{\partial\mathbf{g}}{\partial\mathbf{t}_j}|\frac{\partial\mathbf{g}}{\partial\mathcal{R}_i}|\frac{\partial\mathbf{g}}{\partial\mathbf{t}_i}|\right) = d\left(\pi\right)\times d\left(\mathsf{K}\mathcal{R}_j{}^T\left[\mathsf{I}_{3\times3}|-\mathbf{t}_j\right]\tilde{\mathcal{Q}}_i\right) \tag{B.28}$$

L'équation B.28 est identique à l'équation B.3 de l'ajustement de faisceaux standard, le calcul des dérivées est similaire. Les équations sont détaillées dans la section B.1.

- Pour les dérivées par rapport aux paramètres de la **rotation** de la caméra j, nous obtenons :
$$\frac{\partial \mathbf{g}}{\partial \omega_j} = d(\pi) \times \mathsf{K}\mathcal{R}_{j0}^T \times -\frac{\partial \left[\omega_j\right]_\times}{\partial \omega_j} \left[\mathsf{I}_{3\times 3}\right| - \mathbf{t}_j] \, \tilde{\mathcal{Q}}_i, \tag{B.29}$$
- Les dérivées de la fonction $\mathbf{g}$ par rapport aux paramètres de **translation** de la caméra j sont données par l'équation suivante :
$$\frac{\partial \mathbf{g}}{\partial \mathbf{t}_j} = -1 \times d(\pi) \times \mathsf{K}\mathcal{R}_j^T. \tag{B.30}$$

Dérivées par rapport aux paramètres de la caméra générant le point 3D Pour le calcul des dérivées de la caméra i qui génère le point 3D, nous adoptons les mêmes hypothèses que précédemment, mais la manière dont les dérivées sont calculées sera légèrement modifiée.

Si nous reprenons l'équation B.24 ; nous notons $\mathcal{Q}'_i$ le point 3D généré par la caméra i et exprimé dans le repère de cette même caméra tel que :
$$\mathcal{Q}'_i = \pi^{-1}\left(q_i,\ \mathcal{Z}\right) \tag{B.31}$$
L'équation B.24 va s'écrire :
$$d\left(\mathbf{g}\right) = \left(\frac{\partial \mathbf{g}}{\partial \mathcal{R}_j}\middle|\frac{\partial \mathbf{g}}{\partial \mathbf{t}_j}\middle|\frac{\partial \mathbf{g}}{\partial \mathcal{R}_i}\middle|\frac{\partial \mathbf{g}}{\partial \mathbf{t}_i}\middle|\right) = d\left(\pi\right) \times d\left(\mathsf{K}\mathsf{P}_j\mathsf{P}_i^{-1}\tilde{\mathcal{Q}}'_i\right) \tag{B.32}$$
Comme $\mathsf{P}_i^{-1} = \left(\mathcal{R}_i{}^T \left[\mathsf{I}_{3\times 3}\right| \ -\mathbf{t}_i]\right)^{-1} = [\mathcal{R}_i|\mathbf{t}_i]$. L'équation peut s'écrire de la manière suivante :
$$d\left(\mathbf{g}\right) = d\left(\pi\right) \times d\left(\mathsf{K}\mathsf{P}_j\left[\mathcal{R}_i|\mathbf{t}_i\right]\tilde{\mathcal{Q}}'_i\right) \tag{B.33}$$

- **Dérivées par rapport aux paramètres de la rotation** : Pour le calcul des dérivées de la caméra i qui génère le point 3D par rapport aux paramètres de la rotation nous adoptons les mêmes hypothèses que précédemment en utilisant un repère relatif au voisinage de la rotation courante $\mathcal{R}_0^i$, telle que $\mathcal{R}_0^i(\omega^i) = \mathcal{R}(\omega^i)\mathcal{R}_0^i$, avec $\omega^i = (\omega_1^i, \omega_2^i, \omega_3^i)^T$ contient les faibles angles d'Euler autour de $\mathcal{R}_0^i$.
En choisissant cette hypothèse, nous obtenons l'équation suivante :
$$\begin{aligned}\frac{\partial \mathbf{g}}{\partial \omega^i} &= d(\pi) \times \frac{\partial}{\partial \omega^i}\left(\mathsf{K}\mathsf{P}_j\left[\mathcal{R}_0^i(\omega_i)\middle|\ \mathbf{t}_i\right]\tilde{\mathcal{Q}}'_i\right) \\ &= d(\pi) \times \mathsf{K}\mathsf{P}_j \times \frac{\partial}{\partial \omega^i}\left(\left[\mathcal{R}(\omega^i)\mathcal{R}_0^i\middle|\ \mathbf{t}_i\right]\right)\tilde{\mathcal{Q}}'_i\end{aligned} \tag{B.34}$$
Notons $\tilde{\mathcal{Q}}'_i = [\mathcal{X}_i; 1]^\top$; l'équation B.34 va s'écrire :
$$\begin{aligned}\frac{\partial \mathbf{g}}{\partial \omega^i} &= d(\pi) \times \mathsf{K}\mathsf{P}_j \times \frac{\partial}{\partial \omega^i}\left(\left(\mathcal{R}(\omega^i)\mathcal{R}_0^i\right)\mathcal{X}'_i + \mathbf{t}_i\right) \\ &= d(\pi) \times \mathsf{K}\mathsf{P}_j \times \frac{\partial}{\partial \omega^i}\left(\mathcal{R}(\omega^i)\mathcal{R}_0^i\right)\mathcal{X}_i \\ &= d(\pi) \times \mathsf{K}\mathsf{P}_j \times \frac{\partial}{\partial \omega^i}\mathcal{R}(\omega^i) \times \mathcal{R}_0^i\mathcal{X}_i \\ &= d(\pi) \times \mathsf{K}\mathsf{P}_j \times \frac{\partial \left[\omega^i\right]_\times}{\partial \omega_k^i} \times \mathcal{R}_0^i\mathcal{X}_i\end{aligned} \tag{B.35}$$

Avec les $\frac{\partial [\omega]_\times}{\partial \omega_k}$, $k \in \{1,2,3\}$ sont :

$$\frac{\partial \left[\omega\right]_\times}{\partial \omega_1} \approx \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \tag{B.36}$$

$$\frac{\partial \left[\omega\right]_\times}{\partial \omega_2} \approx \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \tag{B.37}$$

$$\frac{\partial \left[\omega\right]_\times}{\partial \omega_3} \approx \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{B.38}$$

- **Dérivées par rapport aux paramètres de la translation** : Les dérivées par rapport aux paramètres de translation sont données par les équations suivantes :

$$\begin{aligned} \frac{\partial \mathbf{g}}{\partial \mathbf{t}_i} &= d(\pi) \times \frac{\partial}{\partial \mathbf{t}_i} \left(\mathsf{KP}_j \left[\mathcal{R}_i | \, \mathbf{t}_i\right] \tilde{\mathcal{Q}}'_i \right) \\ &= d(\pi) \times \mathsf{KP}_j \times \frac{\partial}{\partial \mathbf{t}_i} \left(\left[\mathcal{R}_i | \, \mathbf{t}_i\right] \right) \tilde{\mathcal{Q}}'_i \\ &= d(\pi) \times \mathsf{KP}_j \end{aligned} \tag{B.39}$$

Bibliographie

Agarwal, P., Grisetti, G., Tipaldi, G. D., Spinello, L., Burgard, W., and Stachniss, C. (2014). Experimental analysis of dynamic covariance scaling for robust map optimization under bad initial estimates. In *International Conference on Robotics and Automation*. IEEE.

Audras, C., Comport, A. I., Meilland, M., and Rives, P. (2011). Real-time dense rgb-d localisation and mapping. In *Australian Conference on Robotics and Automation*.

Barron, J. T. and Malik, J. (2013). Intrinsic scene properties from a single rgb-d image. In *Conference on Computer Vision and Pattern Recognition*. IEEE.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf : Speeded up robust features. In *European conference on computer vision*. Springer.

Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-d shapes. In *Robotics-DL tentative*. International Society for Optics and Photonics.

Caruso, D., Engel, J., and Cremers, D. (2015). Large-scale direct slam for omnidirectional cameras. In *International Conference on Intelligent Robots and Systems*. IEEE.

Castellanos, J. A. and Tardos, J. D. (2012). *Mobile robot localization and map building : A multisensor fusion approach*. Springer Science & Business Media.

Castle, R. O., Klein, G., and Murray, D. W. (2011). Wide-area augmented reality using camera tracking and mapping in multiple regions. *Computer Vision and Image Understanding*, 115(6) :854–867.

Chen, J., Bautembach, D., and Izadi, S. (2013). Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics*, 32(4) :113.

Chen, Y. and Medioni, G. (1992). Object modelling by registration of multiple range images. *Image and vision computing*, 10(3) :145–155.

Chum, O., Matas, J., and Obdrzalek, S. (2004). Enhancing ransac by generalized model optimization. In *Asian Conference on Computer Vision*. IEEE.

Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM.

Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *International Conference on Computer Vision*. IEEE.

Davison, A. J. and Murray, D. W. (2002). Simultaneous localization and map-building using active vision. *IEEE transactions on pattern analysis and machine intelligence*, 24(7) :865–880.

Dellaert, F., Fox, D., Burgard, W., and Thrun, S. (1999). Monte carlo localization for mobile robots. In *International Conference on Robotics and Automation*. IEEE.

Dellaert, F. and Kaess, M. (2006). Square root sam : Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12) :1181–1203.

Dementhon, D. F. and Davis, L. S. (1995). Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15 :123–141.

Dou, M., Khamis, S., Degtyarev, Y., Davidson, P., Fanello, S. R., Kowdle, A., Escolano, S. O., Rhemann, C., Kim, D., Taylor, J., et al. (2016). Fusion4d : Real-time performance capture of challenging scenes. *ACM Transactions on Graphics*, 35(4) :114.

Ellum, C. (2006). Integration of raw gps measurements into a bundle adjustment. *IAPRS series vol. XXXV*, 3025.

Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., and Burgard, W. (2012). An evaluation of the rgb-d slam system. In *International Conference on Robotics and Automation*. IEEE.

Engel, J., Schöps, T., and Cremers, D. (2014). Lsd-slam : Large-scale direct monocular slam. In *European Conference on Computer Vision*. Springer.

Engel, J., Stückler, J., and Cremers, D. (2015). Large-scale direct slam with stereo cameras. In *International Conference on Intelligent Robots and Systems*. IEEE.

Estrada, C., Neira, J., and Tardós, J. D. (2005). Hierarchical slam : Real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4) :588–596.

Fallon, M. F., Johannsson, H., and Leonard, J. J. (2012). Efficient scene simulation for robust monte carlo localization using an rgb-d camera. In *International Conference on Robotics and Automation*. IEEE.

Fang, Z. and Scherer, S. (2015). Real-time onboard 6dof localization of an indoor mav in degraded visual environments using a rgb-d camera. In *International Conference on Robotics and Automation*. IEEE.

Geva, A., Briskin, G., Rivlin, E., and Rotstein, H. (2015). Estimating camera pose using bundle adjustment and digital terrain model constraints. In *International Conference on Robotics and Automation*. IEEE.

Guivant, J. and Nebot, E. (2001). Compressed filter for real time implementation of simultaneous localization and map building. In *International Conference on Field and Service Robots*, volume 1.

Gutmann, J.-S. and Konolige, K. (1999). Incremental mapping of large cyclic environments. In *International Symposium on Computational Intelligence in Robotics and Automation*. IEEE.

Handa, A., Newcombe, R., Angeli, A., and Davison, A. (2012). Real-time camera tracking : When is high frame-rate best ? pages 222–235. Springer.

Handa, A., Whelan, T., McDonald, J., and Davison, A. J. (2014). A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *International Conference on Robotics and Automation*. IEEE.

Haralick, B. M., Lee, C.-N., Ottenberg, K., and Nölle, M. (1994). Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision*, 13 :331–356.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Alvey vision conference*. Citeseer.

Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

Hartley, R. I. (1997). In defense of the eight-point algorithm. *Transactions on pattern analysis and machine intelligence*, 19 :580–593.

Henry, P., Fox, D., Bhowmik, A., and Mongia, R. (2013). Patch volumes : Segmentation-based consistent mapping with rgb-d cameras. In *International Conference on 3D Vision*. IEEE.

Henry, P., Krainin, M., Herbst, E., Ren, X., and Fox, D. (2012). Rgb-d mapping : Using kinect-style depth cameras for dense 3d modeling of indoor environments. *International Journal of Robotics Research*, 31(5) :647–663.

Horaud, R. and Monga, O. (1995). *Vision par ordinateur : outils fondamentaux*. Editions Hermès.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). Octomap : An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34(3) :189–206.

Huang, A. S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., and Roy, N. (2011). Visual odometry and mapping for autonomous flight using an rgb-d camera. In *International Symposium on Robotics Research*. IFRR.

Johannsson, H., Kaess, M., Fallon, M., and Leonard, J. J. (2013). Temporally scalable visual slam using a reduced pose graph. In *International Conference on Robotics and Automation*. IEEE.

Julier, S. J. and Uhlmann, J. K. (2001). A counter example to the theory of simultaneous localization and map building. In *International Conference on Robotics and Automation*. IEEE.

Kaess, M., Ranganathan, A., and Dellaert, F. (2008). isam : Incremental smoothing and mapping. *Transactions on Robotics*, 24(6) :1365–1378.

Kahlmann, T. (2007). *Range imaging metrology : Investigation, calibration and development*. PhD thesis, University of Hannover.

Kerl, C., Sturm, J., and Cremers, D. (2013a). Dense visual slam for rgb-d cameras. In *International Conference on Intelligent Robots and Systems*. IEEE.

Kerl, C., Sturm, J., and Cremers, D. (2013b). Robust odometry estimation for rgb-d cameras. In *International Conference on Robotics and Automation*. IEEE.

Khoshelham, K. and Elberink, S. O. (2012). Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2) :1437–1454.

Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. In *International Symposium on Mixed and Augmented Reality*. IEEE.

Konolige, K. and Agrawal, M. (2008). Frameslam : From bundle adjustment to real-time visual mapping. *Transactions on Robotics*, 24(5) :1066–1077.

Konolige, K. and Bowman, J. (2009). Towards lifelong visual maps. In *International Conference on Intelligent Robots and Systems*. IEEE.

Konolige, K., Bowman, J., Chen, J., Mihelich, P., Calonder, M., Lepetit, V., and Fua, P. (2010). View-based maps. *International Journal of Robotics Research*, 29(8) :941–957.

Kummerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). g 2 o : A general framework for graph optimization. In *International Conference on Robotics and Automation*. IEEE.

Larnaout, D., Gay-Bellile, V., Bourgeois, S., and Dhome, M. (2013). Vehicle 6-dof localization based on slam constrained by gps and digital elevation model information. In *International Conference on Image Processing*. IEEE.

Latif, Y., Cadena, C., and Neira, J. (2013). Robust loop closing over time for pose graph slam. *International Journal of Robotics Research*, 32(14) :1611–1626.

Lavest, J.-M., Viala, M., and Dhome, M. (1998). Do we really need an accurate calibration pattern to achieve a reliable camera calibration ? In *European Conference on Computer Vision*. Springer.

Lerner, R., Rivlin, E., and Rotstein, H. P. (2006). Pose and motion recovery from feature correspondences and a digital terrain map. *IEEE transactions on pattern analysis and machine intelligence*, 28(9) :1404–1417.

Lhuillier, M. (2011). Fusion of gps and structure-from-motion using constrained bundle adjustments. In *International Conference on Computer Vision and Pattern Recognition*. IEEE.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2) :91–110.

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

McDonald, J., Kaess, M., Cadena, C., Neira, J., and Leonard, J. J. (2013). Real-time 6-dof multi-session visual slam over large-scale environments. *Robotics and Autonomous Systems*, 61(10) :1144–1158.

Meilland, M. and Comport, A. I. (2013). On unifying key-frame and voxel-based dense visual slam at large scales. In *International Conference on Intelligent Robots and Systems*. IEEE.

Melbouci, K., Collette, S. N., Gay-Bellile, V., Ait-Aider, O., Carrier, M., and Dhome, M. (2015). Bundle adjustment revisited for slam with rgbd sensors. In *International Conference on Machine Vision Applications*. IEEE.

Melbouci, K., Collette, S. N., Gay-Bellile, V., Ait-Aider, O., and Dhome, M. (2016). Model based rgbd slam. In *International Conference on Image Processing*. IEEE.

Mikolajczyk, K. and Schmid, C. (2002). An affine invariant interest point detector. *Computer Vision*, pages 128–142.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *Transactions on pattern analysis and machine intelligence*, 27(10) :1615–1630.

Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B., et al. (2002). Fastslam : A factored solution to the simultaneous localization and mapping problem. In *National conference on Artificial intelligence*. LAAAI.

Montiel, J. M., Civera, J., and Davison, A. J. (2006). Unified inverse depth parametrization for monocular slam. Robotics : Science and Systems.

Moré, J. J. (1978). The levenberg-marquardt algorithm : implementation and theory. In *Numerical analysis*, pages 105–116. Springer.

Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., and Sayd, P. (2006a). Monocular vision based slam for mobile robots. In *International Conference on Pattern Recognition*. IEEE.

Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., and Sayd, P. (2006b). Real time localization and 3d reconstruction. In *International Conference on Computer Vision and Pattern Recognition*. IEEE.

Mur-Artal, R. and Tardos, J. D. (2016). Orb-slam2 : an open-source slam system for monocular, stereo and rgb-d cameras.

Newcombe, R. A., Fox, D., and Seitz, S. M. (2015). Dynamicfusion : Reconstruction and tracking of non-rigid scenes in real-time. In *International Confernce On Computer Vision and Pattern Recognition*. IEEE.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011a). Kinectfusion : Real-time dense surface mapping and tracking. In *international symposium on Mixed and augmented reality*. IEEE.

Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. (2011b). Dtam : Dense tracking and mapping in real-time. In *International conference on Computer Vision*. IEEE.

Newman, P. (1999). On the structure and solution of the simultaneous localisation and map building problem. *Doctoral diss., University of Sydney*, 41.

Nilsson, J., Fredriksson, J., and Ödblom, A. C. (2013). Bundle adjustment using single-track vehicle model. In *International Conference on Robotics and Automation*. IEEE.

Nistér, D. (2004). An efficient solution to the five-point relative pose problem. *Transactions on pattern analysis and machine intelligence*, 26(6) :756–770.

Oishi, S., Jeong, Y., Kurazume, R., Iwashita, Y., and Hasegawa, T. (2013). Nd voxel localization using large-scale 3d environmental map and rgb-d camera. In *International Conference of Robotics and Biomimetics*. IEEE.

Olson, E. and Agarwal, P. (2013). Inference on networks of mixtures for robust robot mapping. *International Journal of Robotics Research*, 32(7) :826–840.

Olson, E., Leonard, J., and Teller, S. (2006). Fast iterative alignment of pose graphs with poor initial estimates. In *International Conference on Robotics and Automation*. IEEE.

Ozog, P. and Eustice, R. M. (2016). Large-scale model-assisted bundle adjustment using gaussian max-mixtures. In *International Conference on Robotics and Automation*. IEEE.

Pirker, K., Rüther, M., Schweighofer, G., and Bischof, H. (2011). Gpslam : Marrying sparse geometric and dense probabilistic visual mapping. In *British Machine Vision Conference*.

Raguram, R., Frahm, J.-M., and Pollefeys, M. (2008). A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus. *Computer Vision*, pages 500–513.

Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H., and Davison, A. J. (2013). Slam++ : Simultaneous localisation and mapping at the level of objects. In *International Conference on Computer Vision and Pattern Recognition*. IEEE.

Santos, F. (2015). L'algorithme em : une courte présentation.

Sarbolandi, H., Lefloch, D., and Kolb, A. (2015). Kinect range sensing : Structured-light versus time-of-flight kinect. *Computer vision and image understanding*, 139 :1–20.

Scherer, S. A., Dube, D., and Zell, A. (2012). Using depth in visual simultaneous localisation and mapping. In *International Conference on Robotics and Automation*. IEEE.

Schmid, C., Mohr, R., and Bauckhage, C. (2000). Evaluation of interest point detectors. *International Journal of computer vision*, 37(2) :151–172.

Sibley, G., Mei, C., Reid, I., and Newman, P. (2010). Vast-scale outdoor navigation using adaptive relative bundle adjustment. *International Journal of Robotics Research*, 29(8) :958–980.

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

Steinbrucker, F., Kerl, C., and Cremers, D. (2013). Large-scale multi-resolution surface reconstruction from rgb-d sequences. In *International Conference on Computer Vision*. IEEE.

Strasdat, H., Montiel, J., and Davison, A. J. (2010). Real-time monocular slam : Why filter ? In *International Conference on Robotics and Automation*. IEEE.

Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of rgb-d slam systems. In *International Conference on Intelligent Robots and Systems*. IEEE.

Sünderhauf, N. and Protzel, P. (2012). Switchable constraints for robust pose graph slam. In *International Conference on Intelligent Robots and Systems*. IEEE.

Tamaazousti, M. (2013). *L'ajustement de faisceaux contraint comme cadre d'unification des méthodes de localisation : application à la réalité augmentée sur des objets 3D*. PhD thesis, Université Blaise Pascal-Clermont-Ferrand II.

Tamaazousti, M., Gay-Bellile, V., Collette, S. N., Bourgeois, S., and Dhome, M. (2011a). Nonlinear refinement of structure from motion reconstruction by taking advantage of a partial knowledge of the environment. In *Confernce on Computer Vision and Pattern Recognition*. IEEE.

Tamaazousti, M., Gay-Bellile, V., Naudet-Collette, S., Bourgeois, S., and Dhome, M. (2011b). Nonlinear refinement of structure from motion reconstruction by taking advantage of a partial knowledge of the environment. In *Conference on Computer Vision and Pattern Recognition*. IEEE.

Tardós, J. D., Neira, J., Newman, P. M., and Leonard, J. J. (2002). Robust mapping and localization in indoor environments using sonar data. *The International Journal of Robotics Research*, 21 :311–330.

Thrun, S., Fox, D., Burgard, W., and Dellaert, F. (2001). Robust monte carlo localization for mobile robots. *Artificial intelligence*, 128(1-2) :99–141.

Torr, P. H. and Murray, D. W. (1997). The development and comparison of robust methods for estimating the fundamental matrix. *International Journal of Computer Vision*, 24 :271–300.

Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W. (1999). Bundle adjustment a modern synthesis. In *International workshop on vision algorithms*. Springer.

Tykkälä, T., Comport, A. I., and Kämäräinen, J.-K. (2013). Photorealistic 3d mapping of indoors by rgb-d scanning process. In *International Conference on Intelligent Robots and Systems*. IEEE.

Wasenmüller, O., Meyer, M., and Stricker, D. (2016). Corbs : Comprehensive rgb-d benchmark for slam using kinect v2. In *Winter Conference on Applications of Computer Vision*. IEEE.

Wasenmüller, O. and Stricker, D. (2016). Comparison of kinect v1 and v2 depth images in terms of accuracy and precision. In *Asian Conference on Computer Vision*. Springer.

Whelan, T., Johannsson, H., Kaess, M., Leonard, J. J., and McDonald, J. (2013a). Robust real-time visual odometry for dense rgb-d mapping. In *International Conference on Robotics and Automation*. IEEE.

Whelan, T., Kaess, M., Johannsson, H., Fallon, M., Leonard, J. J., and McDonald, J. (2015). Real-time large-scale dense rgb-d slam with volumetric fusion. *International Journal of Robotics Research*, 34(4-5) :598–626.

Whelan, T., Kaess, M., Leonard, J. J., and McDonald, J. (2013b). Deformation-based loop closure for large scale dense rgb-d slam. In *International Conference on Intelligent Robots and Systems*. IEEE.

Williams, S. B. (2001). *Efficient solutions to autonomous mapping and navigation problems*. PhD thesis, University of Sydney. Aerospace, Mechanical and Mechatronic Engineering.

Winterhalter, W., Fleckenstein, F., Steder, B., Spinello, L., and Burgard, W. (2015). Accurate indoor localization for rgb-d smartphones and tablets given 2d floor plans. In *International Conference on Intelligent Robots and Systems*. IEEE.

Zeng, M., Zhao, F., Zheng, J., and Liu, X. (2012). A memory-efficient kinectfusion using octree. In *Computational Visual Media*. Springer.

Zhang, J., Kaess, M., and Singh, S. (2014). Real-time depth enhanced monocular odometry. In *International Conference on Intelligent Robots and Systems*. IEEE.

Zhou, Q.-Y. and Koltun, V. (2013). Dense scene reconstruction with points of interest. *Transactions on Graphics*, 32(4) :112.

Zhou, Q.-Y., Miller, S., and Koltun, V. (2013). Elastic fragments for dense scene reconstruction. In *International Conference on Computer Vision*. IEEE.

Table des figures

Liste des tableaux

Liste des Algorithmes

Table des matières