

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ FÉDÉRALE TOULOUSE MIDI-PYRÉNÉES

Délivré par :
l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)

Présentée et soutenue le *19/10/2018* **par :**

Philippe-Antoine GOHARD

De la réalité augmentée sans marqueur pour l'aménagement d'intérieur à la réalité diminuée sur plateforme mobile

---

JURY

Marie-Odile BERGER	DR HDR INRIA	Rapporteur
Michel DEVY	DR CNRS	co Directeur de thèse
David FOFI	Professeur d'Université	Rapporteur
Géraldine MORIN	Maître de Conférences HDR	Examinateur
El Mustapha MOUADDIB	Professeur d'Université	Examinateur
Bertrand VANDEPORTAELE	Maître de Conférences	co Directeur de thèse

---

École doctorale et spécialité :
EDSYS : Informatique 4200018
Unité de Recherche :
Laboratoire d'analyse et d'architecture des systèmes
Directeur(s) de Thèse :
Michel DEVY et *Bertrand VANDEPORTAELE*
Rapporteurs :
Marie-Odile BERGER et *David FOFI*

Remerciements

Je tiens en premier lieu à remercier Michel Devy, Directeur de recherche émérite au LAAS-CNRS, et Bertrand Vandeportaele, Maître de conférence, qui furent mes deux directeurs de thèse, et qui m'ont apporté soutien, conseils, ainsi qu'un regard critique sur les différents travaux développés.

Je souhaite également remercier Stéphane Mercier, président d'Innersense et co-encadrant, pour sa confiance accordée, ses nombreuses idées, son pragmatisme ainsi que son écoute au sujet des différents obstacles rencontrés. Je remercie également son implication dans la création et le financement des différents projets aux côtés de Xavier Crouilles, directeur général d'Innersense.

J'exprime ma profonde gratitude envers Marie-Odile Berger et Dadid Fofi, qui ont accepté d'être rapporteurs de cette thèse et ont apporté conseils, commentaires et observations à cette ouvrage. Je salue également Géraldine Morin ainsi que El Mustapha Mouaddib pour avoir pris part à mon jury de thèse.

Je remercie Benjamin Coudrin pour son encadrement apporté, son écoute, ses idées, et son soutien durant nos nombreux entretiens.

Un grand merci à Jorge Othon Esparza Jimenez pour sa collaboration sur les travaux de localisation et de cartographie simultanées, et en particulier sur les méthodes d'optimisation et l'utilisation de la bibliothèque g2o. Je souhaite le remercier également pour son soutien global et l'apport de son aide, de sa réflexion et de ses conseils sur de nombreux sujets abordés dans cette thèse.

Je remercie également l'ensemble de l'équipe Robotique, Action et Perception du LAAS, pour leur accueil et leur soutien ; Jessica Combier, Tristan Kemkla, Claire Labit-Bonis, William Gélard, Camille Maurice, Thomas Forgue, Julien Dufour, Gabriel Bustamante, Daniel Torteï, François Brenot, Ali Alamwhi.

Je remercie particulièrement Maxime Daisy et Julien Fayer, le pôle recherche d'Innersense, pour leur collaboration à la conception des applications de réalité diminuée et pour toute l'aide qu'ils m'ont apporté, aussi bien théorique que pratique, sur les travaux développés dans cette thèse.

J'adresse également à l'ensemble de l'équipe d'Innersense toute ma reconnaissance ; Kevin Sans et Maxime Maillet pour leurs apports dans la modélisation 3D de modèle de scène, Vincent Bourdier, David Lericolais, Julien Enocq et Alvin Bouchéraa pour leur aide dans le développement et l'intégration sur plateforme mobile, et enfin le reste de l'équipe pour la bonne humeur et la bonne ambiance apportées au sein des locaux.

Je remercie tout particulièrement Nastasia Martin, pour sa relecture attentive de ce manuscrit mais surtout pour l'ensemble de son soutien inconditionnel.

Finalement, je remercie tout ceux qui ont contribué de près ou de loin à cette thèse, et surtout ma famille et mes amis pour leur appui moral sans faille.

Resumé

La réalité augmentée en vision désigne les systèmes rendant possibles l'incrustation d'objets virtuels dans une séquence d'images en temps réel. Les applications de cette technologie sont multiples et touchent de plus en plus de domaines, mais la diffusion massive des téléphones mobiles équipés d'une caméra ont permis le déploiement de premiers services grand public exploitant la réalité augmentée. Le contexte de cette thèse concerne l'utilisation de cette technique sur plateforme mobile pour assister un usager à décorer un environnement d'intérieur, notamment en rajoutant des meubles virtuels dans les images. La réalité augmentée suppose la localisation de la caméra ainsi qu'une reconstruction partielle de la scène observée afin de pouvoir disposer le meuble de manière physiquement cohérente avec l'environnement.

Dans une première contribution, nous étudions comment exploiter les caméras équipant les téléphones mobiles, généralement à obturateur déroulant. Ces caméras exposent les lignes d'une image à des temps différents, au contraire des caméras à obturateur global qui exposent l'ensemble de l'image à un instant donné. Ainsi, dans le cas d'un mouvement relatif entre la caméra et la scène, chaque ligne est exposée d'un point de vue différent, ce qui produit des distorsions dans les images.

D'abord, nous étudions l'impact de l'utilisation de caméras à obturateur déroulant sur les algorithmes associés à la réalité augmentée, en particulier sur l'Odométrie Visuelle (VO), ou la Cartographie et Localisation Simultanées (SLAM). Nous exploitons un dispositif constitué d'une centrale inertielle bas coût (gyros et accéléros), d'une caméra à obturateur déroulant et d'une caméra à obturateur global montées de manière rigide pour avoir des points de vues très proche. Les algorithmes VO ou SLAM sont appliqués sur des séquences d'images synchronisées, acquises sous un système de capture de mouvements afin de disposer d'une vérité terrain. Ces travaux illustrent la nécessité d'une modélisation plus fidèle du capteur à obturateur déroulant, notamment lors de mouvements rapides de la caméra en rotation.

Ensuite, nous proposons un nouveau modèle en temps continu de la trajectoire effectuée par la caméra afin de pouvoir obtenir une estimée des paramètres extrinsèques pour chaque ligne d'une image. Ce modèle suppose une interpolation à partir d'un ensemble de points de contrôle, les points de contrôle pouvant être disposés de manière non uniforme dans le temps. Cette distribution temporelle non-uniforme des points de contrôle permet de les placer quand et où cela est nécessaire. Nous présentons deux méthodes permettant d'estimer les instants où la génération d'un point de contrôle apporte le plus d'information. Les points de contrôle sont ensuite raffinés par optimisation afin d'obtenir une trajectoire correspondant aux observations. Nous étendons les paramètres de l'optimisation en incluant l'horodatage des points de contrôle et montrons l'amélioration obtenue sur l'estimation de trajectoires réelles.

Dans une deuxième contribution, nous avons étudié comment modéliser une

scène. Le contexte d'intérieur suppose de grands plans de la scène généralement très peu texturés, pour lesquels les méthodes de reconstruction communes par appariement de points échouent. Afin de répondre à cette contrainte, nous proposons l'utilisation de primitives de plus haut niveau ainsi que différentes paramétrisations des amers de la carte. Nous fournissons des modèles de pièce ayant différentes propriétés. Une modélisation plus ou moins fidèle de différents types d'intérieur, est donnée par des paramètres qui sont estimés via un processus d'optimisation.

Nous présentons finalement différentes applications sur le rajout de meubles virtuels pour l'aménagement en milieu d'intérieur. Dans un premier temps, la pose caméra est obtenue à partir d'une seule image et de différentes hypothèses et informations en entrée. L'adaptation d'un algorithme de SLAM existant sur tablette Android est également partiellement abordé. Un module complet d'un processus de réalité altérée est finalement présenté, où l'environnement est en partie effacé afin de pouvoir rajouter un élément virtuel.

Mots clés : Réalité augmentée, Réalité diminuée, Localisation et Cartographie simultanées, Modélisation de caméra, Caméra à obturateur déroulant, Perspective N View, Ajustement de faisceaux, Interpolation, B-Spline non uniforme, Reconstruction, Monde Manhattan

Table des matières

1 Introduction **1**
- 1.1 Contexte . . . 1
  - 1.1.1 Réalité augmentée . . . 1
  - 1.1.2 Matériel visé et caméra à obturateur déroulant . . . 3
  - 1.1.3 Environnement d'intérieur . . . 6
  - 1.1.4 Objectifs . . . 6
- 1.2 Front-End du SLAM visuel : Détection et association de primitives . . . 8
  - 1.2.1 Détection de primitives . . . 9
  - 1.2.2 Suivi de primitives . . . 11
  - 1.2.3 Association de primitives à partir de descripteurs . . . 12
- 1.3 Back-end du SLAM visuel : filtrage et optimisation . . . 13
  - 1.3.1 Approches par filtrage . . . 14
  - 1.3.2 Approches par Optimisation . . . 15
- 1.4 Contributions . . . 19
- 1.5 Organisation du manuscrit . . . 20

2 Notions de base **21**
- 2.1 Algèbre de Lie . . . 21
  - 2.1.1 Variété, géométrie Riemanienne et groupe de Lie . . . 22
  - 2.1.2 Application logarithmique . . . 25
  - 2.1.3 Groupe Special Orthogonal . . . 25
  - 2.1.4 Groupe Special Euclidien . . . 28
- 2.2 Modèle de caméra à obturateur global . . . 30
  - 2.2.1 Paramètres intrinsèques . . . 30
  - 2.2.2 Paramètres extrinsèques . . . 32
  - 2.2.3 Paramètres de distorsions . . . 32
- 2.3 Géométrie épipolaire et recouvrement de l'information 3D . . . 33
  - 2.3.1 Homographie . . . 33
  - 2.3.2 Matrice fondamentale et matrice essentielle . . . 34
  - 2.3.3 Triangulation . . . 34
- 2.4 Interpolation . . . 35
  - 2.4.1 Interpolation linéaire . . . 35
  - 2.4.2 Courbes de Bézier . . . 36
  - 2.4.3 B-Spline . . . 37
  - 2.4.4 B-Spline cumulative . . . 38
- 2.5 Interpolation de poses caméra . . . 39
  - 2.5.1 interpolation linéaire SLERP . . . 39

2.5.2 Interpolation sur la variété 40
2.5.3 B-Splines cumulatives cubiques sur SE3 40
2.6 Outils d'optimisation 42
2.6.1 Introduction à l'ajustement de faisceaux 43
2.6.2 Linéarisation et minimisation de l'erreur 44
2.6.3 Optimisation sur la variété 46
2.6.4 Représentation par graphe 47
2.7 Conlusion 49

3 Caméras à obturateur déroulant et modèles existants 51
3.1 Caméras à obturateur déroulant 51
3.1.1 Altérations photométriques 52
3.1.2 Déformations géométriques 54
3.2 Impact des caméras OD sur les algorithmes de vision 54
3.2.1 Critère d'évaluation 55
3.2.2 Expérience sur données synthétiques 55
3.2.3 Expériences sur données réelles 59
3.2.4 Résultats du SLAM sur séquences réelles OG et OD . . 63
3.3 Modèles existants de caméras OD 68
3.3.1 Modèles de mouvement pour caméra OD 68
3.3.2 Modèle de projection pour caméra OD 70
3.4 Conclusion 72

4 Modèle d'interpolation à distribution temporelle non uniforme pour caméras OD 75
4.1 B-Splines non uniformes 77
4.1.1 Formulation matricielle des B-Splines cubiques non-uniformes 77
4.2 DTNU et génération des poses de contrôle 82
4.2.1 Méthode inertielle 84
4.2.2 Méthode par analyse de l'erreur 85
4.3 Adaption d'algorithmes d'optimisation pour modèle DTNU . . 87
4.3.1 PnP 87
4.3.2 Ajustement de faisceaux 87
4.3.3 Optimisation de trajectoires 88
4.3.4 Optimisation spatio-temporelle des poses de contrôle . 88
4.3.5 Représentation par graphe 89
4.4 Résultats 90
4.4.1 Tests sur données synthétiques 90
4.4.2 Tests sur données réelles 94
4.5 Discussion 97
4.5.1 Utilisation de primitives segments 97
4.5.2 Coût calculatoire 98

4.6 Conclusion 98

5 Reconstruction en milieux intérieurs **101**
5.1 Méthodes de reconstruction dense par vision 102
5.1.1 Reconstruction à partir d'images clés issues d'un SLAM 102
5.1.2 Limitation de la méthode 103
5.1.3 Méthodes de reconstruction en intérieur 104
5.2 Modèles de pièces 105
5.2.1 Modèle parallélépipédique 105
5.2.2 Modèle plan du sol 106
5.2.3 Définition des prédictions et des observations 109
5.2.4 Points 3D 109
5.2.5 Segments et droites 3D 110
5.3 Détection automatique des observations dans l'image 112
5.3.1 Détection des plans principaux 112
5.3.2 Estimation de la structure de la pièce 114
5.3.3 Détection automatique des coins 116
5.4 Optimisation de la structure 119
5.4.1 Données en entrées 120
5.4.2 Optimisation du modèle parallélépipédique 121
5.4.3 Optimisation du modèle "plan du sol" rectiligne 122
5.4.4 Optimisation du modèle "plan du sol" libre 123
5.4.5 Discussion 126
5.5 Perspectives 126
5.5.1 Augmentation du nombre d'images 126
5.5.2 Augmentation du nombre de segments 127
5.5.3 Appariement de segments 3D-2D 128
5.5.4 Apprentissage automatique 130
5.6 Conclusion 130

6 Applications **133**
6.1 Scénario monoculaire 133
6.1.1 Scénario feuille A4 134
6.1.2 Scénario image et centrale inertielle 135
6.2 Réalité augmentée sur plateforme Android 136
6.2.1 Performances 137
6.2.2 Réalité augmentée 138
6.3 Réalité diminuée et réalité altérée 139
6.3.1 Acquisition des images depuis différents points de vue . 141
6.3.2 Acquisition de données haut niveau fournies par l'utilisateur 142
6.3.3 Reconstruction du modèle de scène 143
6.3.4 Effacement 144

6.3.5 Limitations 144
6.4 Conclusion 145

Conclusion **147**

A Annexe **155**
A.1 Dérivées analytiques 155
A.1.1 Approximation sur SE(3) 156
A.1.2 Composition des Jacobiennes 158
A.1.3 Dérivées pour spline sous formes cumulatives 160
A.1.4 Détails des jacobiennes 161
A.2 Modèle pièce points multi-hauteurs 166

Bibliographie **169**

Chapitre 1

Introduction

Sommaire

1.1 Contexte 1
1.1.1 Réalité augmentée 1
1.1.2 Matériel visé et caméra à obturateur déroulant 3
1.1.3 Environnement d'intérieur 6
1.1.4 Objectifs 6
1.2 Front-End du SLAM visuel : Détection et association de primitives 8
1.2.1 Détection de primitives 9
1.2.2 Suivi de primitives 11
1.2.3 Association de primitives à partir de descripteurs 12
1.3 Back-end du SLAM visuel : filtrage et optimisation 13
1.3.1 Approches par filtrage 14
1.3.2 Approches par Optimisation 15
1.4 Contributions 19
1.5 Organisation du manuscrit 20

1.1 Contexte

1.1.1 Réalité augmentée

La réalité augmentée permet l'incrustation d'éléments virtuels dans une séquence d'images issues d'une caméra réelle. Grâce à cette technologie, il est possible de visualiser en temps réel un objet 3D intégré de manière fidèle à l'environnement, ou encore des indications sous la forme de graphiques (flèches, icônes...) ou de textes. Cette technologie a de nombreuses applications, pour lesquelles deux principales catégories se distinguent.

La première concerne les applications qui apportent une information sur l'environnement visualisé, où il est nécessaire de reconnaître des modèles 3D dans les images 2D afin d'y apposer les informations souhaitées. Plusieurs usages sont actuellement mis en place ou envisagés pour la médecine, la mécanique, l'automobile, l'archéologie ou encore le tourisme historique.

Pour illustrer cette catégorie d'applications mentionnons les travaux de M.Tamaazousti au CEA LIST. La figure 1.1 issue de [Tamaazousti 2013] concerne l'assistance à la conception de cuisine : la cuisine originale (à gauche) est localisée en ligne de manière précise, pour modifier de manière virtuelle les couleurs de la crédence ou du mobilier (à droite).



FIGURE 1.1 – Exemple d'applications de la Réalité Augmentée nécessitant de localiser des objets en temps réel, ici des meubles de cuisine.

La seconde catégorie relève des applications de pure visualisation, où l'environnement est a priori inconnu. Pour savoir où rajouter en ligne des informations sur les images, soit un marqueur est positionner dans la scène réelle, soit l'usager pointe où l'ajout doit être fait et une analyse au moins partielle de la scène doit être faite pour que l'image augmentée soit réaliste (par exemple, l'objet doit être sur le sol, ni en dessus, ni au dessous !). Donc en ce cas, la reconstruction de l'environnement sert à positionner l'objet virtuel et à le localiser dans la scène. Ces applications sont aujourd'hui très utilisées dans le domaine du divertissement, de l'essayage virtuel ou du marketing.



FIGURE 1.2 – Exemple d'applications de la Réalité Augmentée sur des appareils mobiles, nécessitant une analyse de la scène : aménagement d'intérieur (gauche) et divertissement (droite).

Cette thèse est effectuée en partenariat entre l'entreprise Innersense et le LAAS-CNRS. Le LAAS (Laboratoire d'Architecture et d'Analyse des Systèmes) est un institut de recherche du CNRS dont les thématiques sont centrées autour de 4 principaux domaines ; informatique, robotique, automatique

et micro-nano technologies.

Innersense est une société, créée en 2014, qui propose des solutions et des applications consacrées à l'essayage virtuel ou à la configuration 3D de meubles sur tablette et sur téléphone portable. Le savoir faire d'Innersense développé au cours des années leur a permis de créer des partenariats avec les principaux acteurs de l'ameublement français, et de s'étendre désormais aux États-Unis et en Italie. Contrairement aux applications de divertissement, l'essayage virtuel d'un objet (meuble, rideaux, bibelot...) dans un environnement réel, éventuellement déjà occupé par des meubles existants, demande une intégration de l'objet virtuel à l'échelle et suffisamment d'espace dans la scène capturée afin de le disposer de manière physiquement cohérente.

Pour être respectée, la première condition nécessite la localisation de la caméra dans la scène à une échelle métrique ainsi qu'une reconstruction partielle de l'environnement. Ces problèmes de cartographie et de localisation simultanées peuvent être résolus de différentes manières qui seront abordées par la suite.

La seconde condition requiert généralement de modifier virtuellement la pièce, voire de la vider. En ce cas, il est donc nécessaire de ne reconstruire que la structure de la pièce et d'ignorer alors les objets réels en les effaçant dans l'image. Le procédé d'effacement d'un objet dans une séquence d'images est dénommé "réalité diminuée".

Dans ce présent travail, nous appellerons "réalité altérée" l'action d'effacer un objet réel pour ensuite incruster un objet virtuel, ce qui correspond à une combinaison des méthodes de réalité diminuée et de réalité augmentée.

Ces technologies font l'objet d'une vive demande dans le milieu de l'aménagement intérieur. Elles font cependant appel à des algorithmes qui nécessitent une capacité de calcul non négligeable. Si elles restent relativement accessibles pour des ordinateurs de bureau, leur adaptation sur des plateformes mobiles telles que téléphones portables ou tablettes est en revanche plus complexe. L'augmentation rapide de la puissance de calcul sur ces plateformes a permis l'arrivée récente d'applications utilisant la réalité augmentée. En revanche, les algorithmes de réalité diminuée ou altérée sont plus coûteux et les applications ne sont pas encore suffisamment matures pour être exploitables par un utilisateur lambda.

1.1.2 Matériel visé et caméra à obturateur déroulant

La réalité augmentée sur tablettes et téléphones portables suppose l'utilisation des capteurs bas-coût intégrés sur ces appareils. Ces appareils possèdent généralement une centrale inertielle (avec accéléromètres, gyromètres et magnétomètres) et une ou plusieurs caméras bas-coût. La distinction entre une centrale inertielle bas-coût et une centrale inertielle haut-de-gamme est généralement caractérisée par une simple différence de précision, les centrales haut-



FIGURE 1.3 – Illustration d'une méthode de réalité diminuée sur un jeu de données synthétiques où l'information 3D est connue a priori. A gauche l'image initiale, à droite l'image après effacement des meubles.

de-gamme produisant des données moins bruitées, à plus haute fréquence, et dérivant moins vite dans le temps.

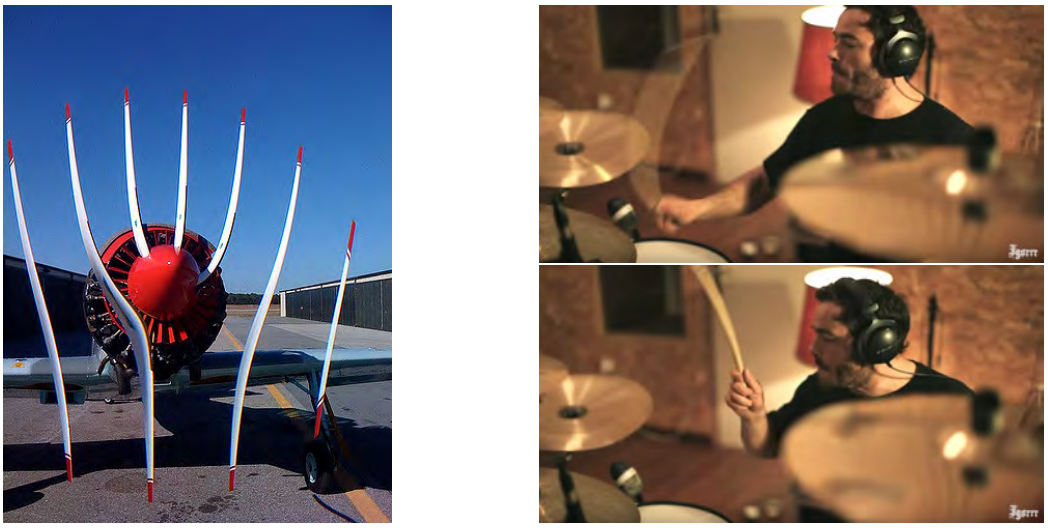


FIGURE 1.4 – Exemple des distorsions induites par l'utilisation d'une caméra à obturateur déroulant. L'exposition non simultanée des lignes de l'image entraîne une déformation des objets en mouvement, comme la baguette du batteur (droite) ou bien l'hélice de l'avion (gauche).

En revanche, les caméras bas-coût qui équipent les tablettes et les téléphones portables présentent une conception différente des caméras communément utilisées par des applications usuelles (vidéo-surveillance, contrôle de qualité, reconnaissance d'objets...) et nécessitent une modélisation adaptée. En effet, ces caméras possèdent un obturateur déroulant (abrégé OD) contrairement aux caméras à obturateur global (abrégé OG). Les lignes de l'image sont alors acquises à des instants différents (voir fig. 1.5). L'obturateur de ces

caméras expose séquentiellement les lignes de l'image (obturateur déroulant), alors que les obturateurs globaux exposent la totalité de l'image durant une même période. Ce principe est similaire à celui des appareils photos équipés d'un obturateur à rideaux. Ces appareils utilisent deux rideaux, l'un découvrant le capteur, et l'autre le recouvrant, afin que chaque ligne de l'image soit exposée pendant une même durée. L'ouverture entre ces deux rideaux permet de contrôler la durée d'exposition de chaque ligne, ce qui rend possible l'obtention de temps de pose très courts pour une faible ouverture.

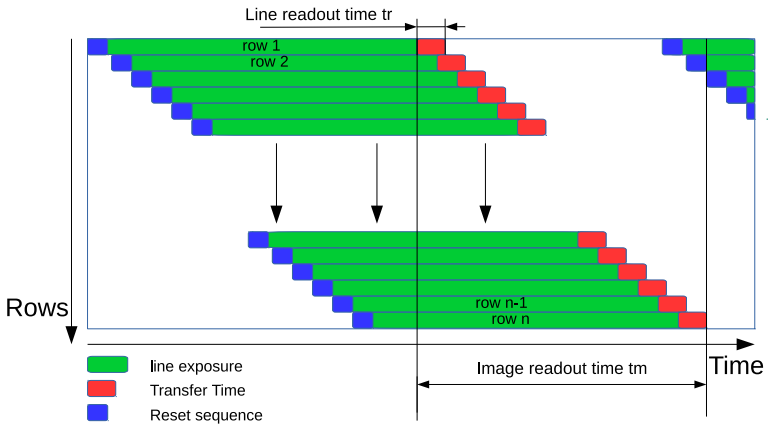


FIGURE 1.5 – Cette figure illustre l'exposition des lignes (en ordonnée) d'une caméra à OD en fonction du temps (en abscisse). Ce type de caméras expose et transfère les lignes de l'image à des instants différents. Une ligne du capteur est d'abord initialisée (bleu), exposée (vert), et ensuite transférée (rouge). Le temps de lecture d'une image OD est défini par le temps de transfert de l'ensemble de l'image.

Cette conception est particulièrement adaptée à la technologie des capteurs CMOS (Complementary Metal Oxide Semiconductors) moins sensibles à la lumière que les capteurs CCD (Charged Coupled Device). En effet, les capteurs CCD en mode OG ne permettent pas de maintenir une haute fréquence d'acquisition des images. Pour ces capteurs l'image est exposée entièrement ; puis les charges captées en chaque pixel sont transférées séquentiellement vers la sortie du circuit, rendant les photodiodes non exposables durant cette opération. L'OD permet quant à lui de transférer une ligne de l'image pendant que d'autres lignes sont exposées en vue de répartir le temps de transfert sur le temps d'acquisition de l'image. L'OD autorise donc in fine une exposition plus longue en chaque ligne de l'image avec une fréquence d'images supérieure à celle permise par les OG.

Nos travaux supposent l'utilisation d'images issues d'un dispositif monoculaire équipé d'une caméra OD.





FIGURE 1.6 – A gauche, une image acquise par une caméra évoluant en intérieur où la structure planaire par morceau de la pièce et l'aspect peu texturé des plans principaux est notable. A droite, une image d'extérieur où la structure de l'environnement est chaotique mais très texturée.

1.1.3 Environnement d'intérieur

Notre contexte applicatif nous amène à travailler sur des séquences d'images acquises dans une pièce d'un environnement intérieur. Ces environnements possèdent des propriétés particulières qui permettent de faire des hypothèses simplificatrices sur la géométrie de la pièce. Les bâtiments urbains sont généralement construits en suivant une disposition géométrique basique des murs, des sols et des plafonds afin d'en simplifier la mise en œuvre. Ainsi les murs, les plafonds et les sols sont considérés comme étant plans. De plus, les murs sont perpendiculaires au plafond, au sol et perpendiculaires entre eux. L'ensemble de ces hypothèses est dénommé hypothèse de monde-Manhattan ($Manhattan - world - assumption$).

Par contre, vis-à-vis des méthodes classiques de vision nécessaires à la réalité augmentée, pour estimer les déplacements de la caméra et pour estimer la structure de ces scènes à partir d'images RGB, deux problèmes peuvent se poser : soit le manque de texture sur les plans principaux des pièces (murs et plafonds), soit au contraire, l'existence de motifs répétitifs (tapisserie, carrelage) qui rend difficile l'association des indices visuels entre images successives de la séquence.

1.1.4 Objectifs

Au commencement de cette thèse, les services de réalité augmentée pour l'aménagement d'intérieur proposées par Innersense se basaient sur VUFORIA [LLC 2015], une solution disponible sur le marché des téléphones portables. Cet outil localise la caméra par rapport à un marqueur dont les dimensions et la texture sont connues à l'avance. Bien que fiable, cette méthode possède deux défauts majeurs :

— L'utilisation d'un marqueur est contraignant pour l'utilisateur.
— Bien qu'une reconstruction partielle de l'environnement soit effectuée,

ses données ne sont pas accessibles, ce qui rend impossible les méthodes de réalité altérée.

La problématique générale de cette thèse consiste à améliorer ou remplacer cette solution existante, puis développer des algorithmes permettant d'ouvrir la voie à des applications de réalité altérée dans un environnement d'intérieur à partir d'images issues d'une caméra OD. Les objectifs principaux sont les suivants :

— Préparation à la transition d'une solution de réalité augmentée basée marqueur vers une solution sans marqueur.
— Évaluation de l'influence de l'Obturateur Déroulant sur les algorithmes de vision, développement d'un modèle adaptée aux caméras OD, prise en compte de ce modèle dans les méthodes de localisation et de reconstruction simultanées par vision monoculaire, ou méthodes de SLAM visuel.
— Implémentation d'une méthode de reconstruction d'un environnement intérieur à partir d'images issues d'une plateforme mobile, en exploitant l'hypothèse monde-Manhattan.
— Développement d'une application de réalité altérée permettant de vider une pièce en se basant sur les contributions précédentes.

Le contexte, les contraintes et les objectifs de cette thèse ont été exposés. Notre principale contribution concerne le développement d'une méthode de SLAM visuel pour caméra OD ; aussi, un résumé des méthodes et des algorithmes nécessaires à la localisation de la caméra et à la reconstruction de l'environnement est développé par la suite.

Une synthèse récente des travaux sur le SLAM est présentée dans [Cadena 2016] ; nous reprenons en Figure 1.7, la structuration du SLAM en deux parties appelées Front-End pour les fonctions qui extraient et associent des primitives depuis les données sensorielles, et Back-End pour les fonctions qui mettent à jour d'une part la position ou la trajectoire du capteur et d'autre part, une carte d'amers dont les observations sont ces primitives. Les deux sections suivantes présentes les principales approches proposées par ces Front et Back-ends.

Les travaux précurseurs du SLAM fin des années 1980, ont été faits dans la communauté Robotique, avec les capteurs exploités sur les robots à ce moment là, essentiellement des télémètres-laser à balayage horizontal : les expérimentations se faisaient sur des trajectoires courtes ; on parlait de SLAM à court terme, qui se comporte comme l'odométrie, avec un cumul des erreurs. Ces méthodes permettent de construire des cartes locales et d'estimer les déplacements du capteur, mais vu le cumul d'erreur, il n'est pas exploitable pour cartographier de grands environnements. Pour le SLAM visuel à court terme, on parle d'odométrie visuelle.

Plus tard, dans les années 2000, on s'est intéressé au SLAM visuel en

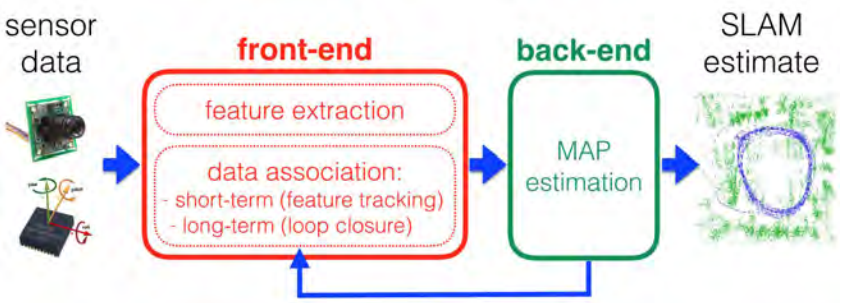


FIGURE 1.7 – Figure issue de [Cadena 2016] qui présente la structuration des méthodes SLAM en deux parties, le front-end et le back-end.

traitant de longues séquences d'images, typiquement acquises par une caméra montée sur un véhicule lors d'un parcours dans un site urbain. Pour éviter le cumul d'erreur, il est nécessaire que le capteur repasse dans des lieux déjà cartographiés, afin d'apparier des observations courantes avec des amers créés lors du passage précédent dans ce lieu. De telles associations permettent de corriger la carte. On parle alors de SLAM à long terme, qui inclue donc la détection de fermetures de boucle : la figure 1.7 inclue dans le front-end, cette fonction de "long term data association (loop closure)".

1.2 Front-End du SLAM visuel : Détection et association de primitives

Les humains se repèrent dans l'espace en utilisant l'ensemble de leurs sens, mais c'est principalement la vue qui permet à l'homme d'évaluer l'environnement et de se localiser dans l'espace. Les yeux envoient des images au cerveau sous formes de signaux qui sont ensuite traités afin d'en extraire des informations utiles notamment pour la localisation. Les yeux peuvent être considérés comme des capteurs images, et le cerveau comme le centre de traitement qui résout les problèmes de la localisation et de la reconstruction à partir d'informations fournies par ces capteurs. Notons que l'être humain possède deux yeux qui permettent de déduire instantanément une information de profondeur (stéréo-vision), information qui n'est pas disponible dans notre configuration monoculaire.

De manière similaire au cerveau, les algorithmes de vision ont pour but d'extraire et de déduire des informations relatives à la position du capteur et à l'environnement depuis les images couleur acquises par une seule caméra montée dans un dispositif déplacé par un usager. Le mouvement spatial effectué par la caméra entre deux images peut être déduit à partir du mouvement pixellique (flot optique) des projections de la scène dans l'image.

Il est donc nécessaire de mettre en correspondance des pixels associés aux

mêmes parties de la scène entre plusieurs images. A partir de ces correspondances, la position relative entre les deux caméras et le modèle géométrique de l'environnement peuvent être estimés.

1.2.1 Détection de primitives

Afin de faciliter la mise en correspondance de pixels, il est nécessaire d'abstraire dans les images des primitives qui sont invariantes à des transformations affines ou perspectives. Le terme "amer visuel" désigne des éléments singuliers de l'environnement 3D, se projettant dans des primitives géométriques 2D qui ont des propriétés les rendant identifiables dans plusieurs images. Différentes primitives peuvent être extraites, telles que des points 2D, des patchs 2D, des segments 2D, des droites 2D et même des objets de plus haut niveau (sémantique). Les primitives les plus utilisées sont les primitives points, notamment pour l'efficacité de leur détection dans les images, la simplicité de leur modèle de projection dans les images et leur aspect localement invariant selon différents points de vue proches. Ces primitives points sont appelées "points d'intérêts".

1.2.1.1 Détection de points d'intérêts

La détection de points d'intérêts permet l'identification de pixels de l'image possédant des propriétés locales intéressantes. L'image peut ainsi être utilisée non plus dans sa totalité (traitement dense), mais être analysée à partir d'un ensemble d'informations locales (traitement épars) représenté par ces points d'intérêts.

De nombreuses méthodes de détection existent, la plus connue étant le détecteur de Harris [Harris 1988] qui conserve les pixels possédant un fort gradient dans plusieurs directions. Cette méthode permet ainsi de détecter des coins texturaux et géométriques. La méthode Good Feature To Track (GFTT) [Shi 1994] améliore légèrement les points de Harris mais repose toujours sur l'utilisation des images de gradient. Le détecteur FAST [Rosten 2006][Rosten 2010] est un détecteur binaire qui analyse un cercle de pixels autour du pixel courant afin d'estimer s'il peut être considéré comme un coin. Enfin le détecteur DOG (pour Difference of Gaussians) est populaire car proposé initialement par D.Lowe comme détecteur des points associés à un descripteur SIFT [Lowe 2004].

Les contours liés à des variations de texture sur un plan varient entre les images selon des projections affines identifiables et aisément modélisables (voir homographie). En revanche, les contours liés à la géométrie ont une apparence qui ne peut être prédite, étant donné qu'un contour géométrique occulte une partie de la scène qui n'est pas encore observée. Ainsi, l'apparence d'un patch (les pixels autour d'un pixel du contour) varie de manière imprévisible. C'est

pourquoi les algorithmes de vision s'adaptent mieux à des scènes très texturées plutôt qu'à des scènes géométriques complexes mais peu texturées.

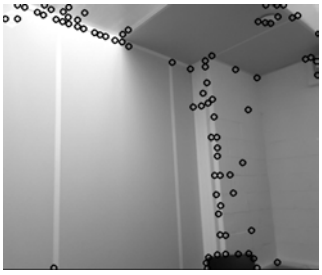


FIGURE 1.8 – Détection de points d'intérêts en utilisant la méthode GFFT de openCV.

1.2.1.2 Détection de segments

Les segments sont un autre type de primitives très utilisées dans la communauté vision. Ils apportent plus d'informations que les points et sont généralement très présents et identifiables dans les scènes d'intérieur. [Duda 1972] propose l'utilisation d'un filtre de Canny pour détecter les contours dans l'image, puis l'application d'une transformée de Hough sur l'image obtenue des contours afin de passer dans l'espace des paramètres de droites. Les droites qui obtiennent le plus de votes dans cet espace sont ensuite sélectionnées. Plus récemment, [von Gioi 2010] expose une nouvelle approche permettant la détection d'un plus grand nombre de segments dans les images.

Ces méthodes sont amplement utilisées pour la reconstruction de structure à partir de lignes détectées dans les images [Bartoli 2003, Klein 2008, Esparza-Jiménez 2016]. La représentation des segments ou des droites dans l'espace n'est pas aussi intuitive que celles des points. Pour une droite, diverses représentations peuvent être utilisées, comme les coordonnées de Plücker [Solà 2012, Esparza-Jiménez 2014] ou la représentation orthonormale [Bartoli 2005, Serafin 2017]. Une segment peut être simplement paramétré par deux points 3D représentant ses extrémités.

Pour pouvoir reconstruire un amer à partir de plusieurs observations dans des images d'une séquence, des méthodes de mise en correspondance de primitives présentes dans ces images doivent être introduites. Il existe deux principales catégories de méthodes de mise en correspondance : (1) une primitive extraite dans l'image I(t) peut être suivie dans I(t+1), ce qui suppose un mouvement pixellique relativement faible entre les images ; (2) toutes les primitives sont détectées en I(t) et I(t+1), et des approches par corrélation brute-force testent deux à deux tous les couples de primitives pour déterminer si elles sont appariées.

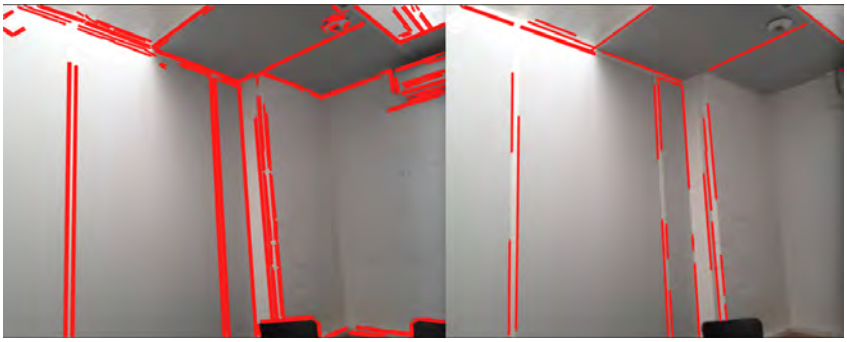


FIGURE 1.9 – Extraction de segment 2D sur une image d'intérieur à partir de la méthode de Hough (droite) et de LSD (gauche).

1.2.2 Suivi de primitives

Il est possible de suivre les primitives détectées dans l'image I(t) en les recherchant dans un voisinage sur l'image suivante I(t+1). Ce voisinage peut être défini de diverses manières :

— sans exploiter des informations venant de la carte 3D, on peut exploiter un modèle de vitesse constante du mouvement apparent du pixel dans l'image ;

— si l'amer 3D correspondant à cette primitive a déjà été reconstruit, alors, on peut exploiter la reprojection de cet amer pour prédire son observation depuis la position courante de la caméra, estimée avec un modèle de vitesse constante de la caméra elle-même.

Les points d'intérêts étant les primitives les plus utilisées dans les algorithmes de vision, il existe de nombreux algorithmes permettant leur suivi 2D. Le KLT proposé par [Lucas 1981] permet de suivre des pixels de fort gradient dans les images en calculant la corrélation avec différents patchs d'image dans le voisinage du pixel considéré. Des approches similaires sont avancées dans [Tomasi 1991] [Shi 1994]. [Bouguet 2000] expose une implémentation pyramidale du KLT. La corrélation entre les patchs est calculée sur une pyramide d'images, afin notamment d'améliorer la robustesse au flou de bouger et d'optimiser l'invariance à l'échelle.

Le suivi de segments est naturellement plus complexe que le suivi de points. [Comport 2006] propose de sous-échantillonner le segment en un sous ensemble de points 2D. Les pixels de fort gradient sont ensuite cherchés dans les directions perpendiculaires au segment et partant de ces points 2D. Une droite est ensuite estimée à partir de ces pixels, depuis laquelle un segment peut être extrait.

Notons que, dans l'idéal, le détecteur est exploité une fois dans I(t) pour initialiser un ensemble de primitives, puis le suivi met à jour les positions de ces primitives dans les images suivantes. En fait, le suivi va perdre des primitives

au fil du temps, ne serait-ce que parce que certaines sortent du champ de vue ; il faut donc relancer le détecteur pour réinitialiser l'ensemble des primitives. Ici aussi, plusieurs stratégies ont été proposées : on remplace immédiatement les primitives perdues en appliquant le détecteur dans des zones image bien choisies, ou bien on attend d'en avoir perdu N% (typiquement 50%) avant de rappeler le détecteur sur toute l'image.

1.2.3 Association de primitives à partir de descripteurs

Les méthodes de suivi de points utilisent un patch d'image autour du point à suivre dans I(t) pour rechercher son correspondant dans l'image suivante I(t+1). Les méthodes d'appariement supposent que le détecteur est exploité sur toutes les images de la séquence ; un descripteur est attaché à chaque point détecté, en codant son voisinage, via des traitements complémentaires ; les points extraits dans deux images (pas forcément consécutives) sont ensuite appariés en comparant leurs descripteurs. La méthode sera d'autant plus efficace que le descripteur sera invariant par rapport aux mouvements de la caméra ; un grand nombre de méthodes existent.

Par exemple, les descripteurs SIFT (pour Scale Invariant FeaTures) [Lowe 2004] utilisent des histogrammes de gradients construits dans des patchs orientés ; ils sont réputés pour leur invariance à l'échelle et à la rotation, mais ils sont longs à calculer. Le descripteur SURF (pour Speeded Up Robust Features) [Bay 2008] s'inspire de SIFT, mais est beaucoup plus rapide à calculer grâce aux images intégrales ; un point d'intérêt est décrit par des sommes de réponses de son voisinage à des ondelettes de Harr. Le descripteur BRIEF (pour Binary Robust Independent Elementary Features) [Calonder 2010] est un descripteur binaire utilisant la comparaison entre paires de pixels prédéfinies pris dans le voisinage du point d'intérêt ; BRIEF n'est invariant ni à l'orientation, ni au changement d'échelle. Aussi sont apparues deux variantes : ORB (pour Oriented fast and Rotated BRIEF) [Rublee 2011] qui est une amélioration de BRIEF invariante à la rotation, et BRISK (pour Binary Robust Invariant Scalable Keypoints) [Leutenegger 2011] qui est invariant aussi au changement d'échelle. Le descripteur FREAK (pour Fast Retina Keypoint) [Ortiz 2012] est aussi un descripteur binaire, mais plus original puisqu'il s'inspire d'un modèle rétinien pour décrire avec une résolution fovéale. Plus récemment [Yi 2016] a proposé LIFT, qui intègre de l'apprentissage profond pour l'estimation de l'orientation et la description des points d'intérêts.

Ces méthodes varient grandement en robustesse, répétabilité et rapidité d'exécution. Il est important de choisir avec précaution la méthode utilisée selon l'application. De nombreuses comparaisons ont été réalisées et sont accessibles dans la littérature [Lefaudeux 2013].

Le cas des segments est plus complexe. Les patchs d'image centrés sur les pixels du segment ne sont généralement pas invariants mais sont très si-

milaires entre eux. [Zhang 2013] propose un descripteur binaire de segments qui est invariant à la translation et relativement robuste aux transformations perspectives.

A partir de divers appariements de pixel à pixel (2D-2D) ou de pixel à points 3D (2D-3D), il est possible d'une part lde recouvrir les poses successives de la caméra, ou bien d'estimer le déplacement relatif entre poses successives, et d'autre part de reconstruction les amers 3D observés par des pixels appariés (2D-2D). Ces méthodes qui sont abordées dans le chapitre suivant, sont utilisées par les algorithmes de SLAM afin d'obtenir une estimée initiale des poses caméras et de l'environnement.

1.3 Back-end du SLAM visuel : filtrage et optimisation

Dans cette section sont décrits les systèmes capables d'estimer la trajectoire caméra et le modèle géométrique de l'environnement, sans connaissance a priori de la scène .

Les algorithmes de cartographie et localisation simultanées par Vision, ou SLAM visuel, sont largement utilisés dans le contexte de la réalité augmentée afin d'estimer la position et l'orientation d'une caméra en mouvement dans une scène fixe. Ces paramètres estimés de la pose caméra peuvent être utilisés pour projeter des objets virtuels dans l'image de manière réaliste et cohérente avec le mouvement réel effectué par la caméra. Le modèle géométrique de l'environnement, est aussi exploité pour incruster un objet virtuel à une position spatiale cohérente dans la scène. Ces méthodes permettent ainsi de se passer du marqueur dans les approches de réalité augmentée.

Initialement, les problèmes de localisation et de cartographie étaient résolus indépendamment. Ignorer la corrélation entre la position des amers de la carte et la pose du capteur mobile exploité pour observer l'environnement [Harris 1987] cause rapidement des inconsistances dans les estimées, essentiellement du fait inéluctable que les observations sont bruitées . L'incertitude, ne prenant pas en compte l'interdépendance entre les amers et la trajectoire, échoue à représenter fidèlement la confiance dans l'estimée produite [Davison 1998].

Il est bien connu qu'il est indispensable de résoudre conjointement les problèmes de localisation et de cartographie, car la carte ne peut être créée que lorsque la pose de la caméra est connue, mais par ailleurs une carte précise est nécessaire pour localiser de ces caméras, d'où l'introduction de l'appelation “Simultaneous Localization and Mapping” (SLAM) notamment par H.Durrant-White.

De manière générale, il s'agit d'estimer la probablibité jointe a posteriori de la carte M_t et de la trajectoire du capteur ou du dispositif mobile (véhicule, robot...) sur lequel il est monté $(x_t, ...x_0)$, connaissant l'ensemble des obser-

vations $(z_t, ...z_0)$ et des commandes transmises au dispositif pour déplacer le capteur dans la scène $(u_t, ...u_0)$

$$P(x_t, ...x_0, M_t | z_t, ...z_0, u_t, ...u_0)$$

Différentes approches ont été proposées afin de résoudre ce problème ; nous évoquons surtout le SLAM visuel monoculaire. Les premières méthodes proposées dans la communauté robotique exploitaient le filtrage pour fusionner surtout des observations par télémétrie-laser.

1.3.1 Approches par filtrage

Le problème du SLAM visuel monoculaire en ligne (calcul des poses et du modèle en temps réel) a premièrement été résolu à l'aide de filtres bayésiens en introduisant un modèle d'observation ou vraisemblance $P(z_t|x_t)$, et un modèle de mouvement $P(x_t|x_{t-1}, u_{t-1})$, ce qui conduit à marginaliser les anciens états, donc à estimer de manière incrémentale dans le temps $P(x_t, M_t|z_t, u_{t-1})$.

Les premiers résultats de l'estimation de la trajectoire d'une caméra et d'une carte de l'environnement en ligne furent montrées dans [Harris 1987]. Cependant, leur méthode utilisait une représentation indépendante de la carte et de la pose. Ce n'est que quinze années plus tard que [Davison 2003] présente MonoSLAM, un processus de SLAM monoculaire complet et temps réel se basant sur un filtre de Kalman étendu (EKF). Leur système utilise une représentation conjointe de la pose courante de la caméra et des points reconstruits dans le vecteur d'état du filtre. L'incertitude sur les points 3D est projetée dans l'image afin d'obtenir l'incertitude sur la prédiction de la position des points d'intérêts. Cette incertitude est ensuite utilisée comme zone de recherche afin de réduire le coût calculatoire de la mise en correspondance des amers dans les images. MonoSLAM est un processus entièrement automatique, incorporant le suivi de points d'intérêts, l'initialisation de nouveaux amers de la carte et l'estimation de leurs profondeurs associées ainsi que la fermeture de boucle pour des trajectoires courtes. En revanche, l'initialisation des amers n'est pas immédiate et nécessite encore des a priori sur la scène ; de ce fait les amers lointains ne sont jamais rajoutés à la carte.

Afin de résoudre ce problème, [Solà 2005] propose l'utilisation pour chaque amer, de multiples hypothèses gaussiennes qui recouvrent l'ensemble du cône de vue de l'amer, quelquesoit sa profondeur vis-à-vis de la position courante de la caméra ; cette modélisation permet d'inclure les amers dans la carte dès leur première observation, ce qui permet de prendre en compte des amers lointains, même si cette méthode suppose cependant un a priori sur la profondeur maximale de ces amers, en fonction du nombre de gaussiennes. [Civera 2006] expose une paramétrisation inverse de la profondeur où les amers sont alors initialisés avec une incertitude infinie le long du rayon optique associé au pixel

où ils sont observés.

Plusieurs systèmes basés sur ces recherches ont abouti par la suite en étendant les domaines d'application (multi-capteurs, multi-robots) tels que RT-SLAM [Roussillon 2011] [Roussillon 2012] et son pendant en odométrie visuelle pour systèmes embarqués C-SLAM [Gonzalez 2013].

Néanmoins, la complexité algorithmique en $O(n^3)$ en fonction du nombre d'amers dans la carte, a motivé la communauté à développer des méthodes permettant d'éviter la prédiction et la ré-estimation des amers qui ne sont plus observés. Ainsi des approches utilisant des sous-cartes contenant les amers non-observés ont été proposées, permettant de se rapprocher d'une complexité en $O(n^2)$ pour l'EKF. Cependant, dans le cas où tous les amers de la carte sont observés, la complexité reste en $O(n^3)$.

Malgré la simplicité et l'efficacité offertes par les approches basées filtrage, des méthodes plus robustes basées optimisation, inspirées par les techniques de Vision géométrique, en particulier de Structure from Motion, furent implémentées plus tardivement en temps réel grâce à l'augmentation des capacités de calcul.

1.3.2 Approches par Optimisation

Les méthodes par optimisation, anciennes dans la communauté Vision ou Photogrammétrie, se sont imposées dans le SLAM visuel depuis 2010 environ. Comme elles s'appuient sur l'ajustement de faisceaux (Bundle Adjustment) [Lourakis 2009, Triggs 2000], ce sont des méthodes plus coûteuses en temps de calcul.

Les principales différences entre filtrage et optimisation sont les suivantes :

— à l'inverse des méthodes de filtrage où les états passés du système sont marginalisés et résumés par une covariance associée à l'état courant, les méthodes par optimisation utilisent l'ensemble ou un sous-ensemble des états passés du système. De ce fait elles estiment la trajectoire suivie par la caméra, ou ses N dernières poses.

— alors qu'une donnée aberrante introduite dans la carte estimée par une méthode par filtrage, sera entièrement intégrée et ne peut donc être corrigée par la suite, l'intégration d'une telle donnée dans un ajustement de faisceaux, aura un effet limité, ou ne sera pas prise en compte en exploitant des techniques robustes d'optimisation.

— de plus, les méthodes par filtrage linéarisent la fonction de coût en fonction des amers de la carte et de la pose courante de la caméra. La fonction de coût n'est pas une fonction linéaire, et sa linéarisation entraîne une erreur qui s'accumule et qui n'est jamais ré-estimée. Dans les méthodes par optimisation, la linéarisation est effectuée par rapport à un ensemble de poses précédentes, et est donc ré-estimée plusieurs fois, permettant de limiter l'accumulation de cette erreur.

— enfin, pour le SLAM à long terme, la solution fournie par les approches de filtrage, vu le cumul d'erreur, n'est pas intégre, c'est-à-dire que les incertitudes, représentées par lea matrices de variance-covariance, sont optimistes, et ne représentent plus les erreurs entre posesestimées et poses réelles du capteur ou des amers. De ce fait les approches de filtrage ne peuvent pas corriger la carte après détection d'une fermeture de boucle.

[Strasdat 2010] établit la supériorité des méthodes par optimisation dans la plupart des configurations face aux méthodes par filtrage, notamment en complexité algorithmique en fonction du nombre d'amers dans la carte ; le filtrage, reste exploité dans des situations spécifiques, par exemple avec l'approche Gmap populaire pour construire la carte d'un environnement de petite dimension.

Afin de pouvoir minimiser une erreur de reprojection dans un temps satisfaisant, les premières méthodes temps-réel basées d'optimisation utilisaient des fenêtres glissantes d'images [Mouragnon 2006]. Ces méthodes se basent sur l'optimisation des n dernières images acquises par la caméra (voir fig 1.10). Cependant, cette approche permet uniquement une odométrie visuelle, étant donné que dans la démarche de base de l'optimisation par fenêtre glissante, les anciennes poses sorties de la fenêtre glissante ne sont plus optimisées. [Sibley 2006], puis [Mourikis 2007], proposent l'utilisation d'un filtre EKF à multiples états, qui s'apparente à une méthode d'optimisation sur une fenêtre glissante ; cette technique donnera d'excellents résultats par la suite [Li 2014].

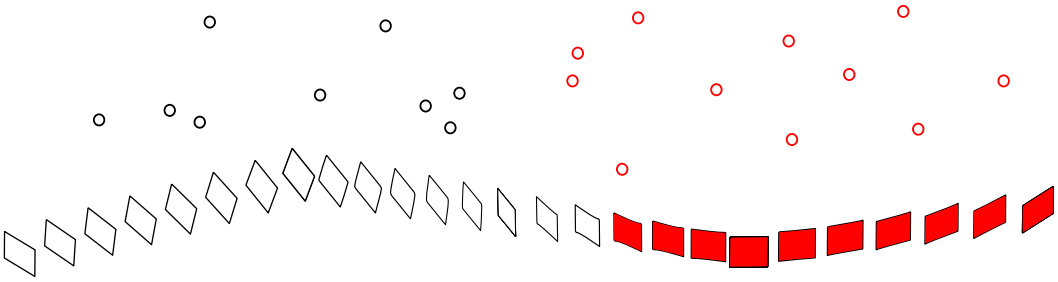


FIGURE 1.10 – Les poses caméra (rectangles) estimées à l'aide d'un ensemble de points 3D reconstruits. Les méthodes par fenêtre glissante optimisent uniquement les n dernières poses caméra et les m derniers points (rouge).

Une autre approche, initialement proposée par [Klein 2007] utilise un sous-ensemble d'images, dénommées "images-clés" [Konolige 2008, Strasdat 2012]. Cette méthode permet de réaliser une optimisation globale sur les images-clés représentatives de la trajectoire (voir fig 1.11). Ces images-clés sont enregistrées lorsque le nombre de points suivis de la précédente image-clé devient trop faible ou lorsqu'une certaine distance a été parcourue. Différentes heuristiques

et seuils peuvent être mis en place pour trouver le meilleur moment où prendre une image clé [Mur-Artal 2015a]. De plus, il est possible de traiter a posteriori les images-clés afin de ne garder que celles apportant de l'information, comme le propose [Mur-Artal 2015a].

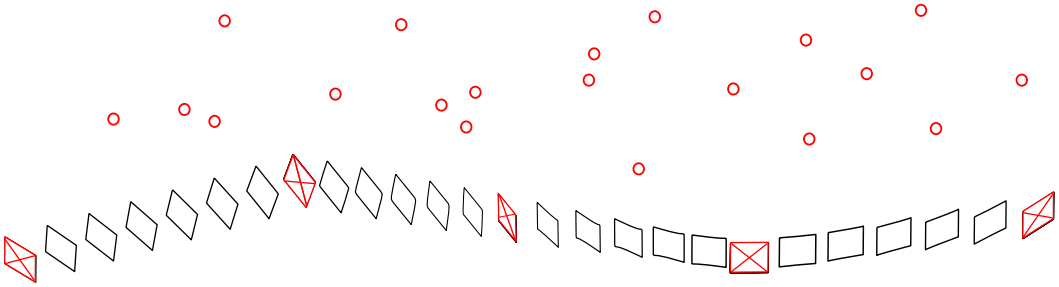


FIGURE 1.11 – Les méthodes utilisant des images-clés (rouge) optimisent un sous ensemble des poses caméras estimées.

Les méthodes par optimisation en SLAM visuel sont aujourd'hui popularisées par de nombreuses implémentations Open Source ; citons ORB-SLAM, LSD-SLAM et SVO.

ORB-SLAM [Mur-Artal 2015a] propose une initialisation automatique en estimant dans des processus concurrents la matrice fondamentale et l'homographie entre deux images-clés, et choisit la méthode la plus adaptée selon une heuristique définie à l'avance. Cette méthode permet d'éviter les dégénérescences produites par l'estimation d'une matrice fondamentale en cas de scènes planaires, et celles produites par l'estimation d'une homographie dans des scènes non planaires. Ils proposent également l'utilisation de 4 processus concurrents, un pour le suivi de points par ORB, un pour l'optimisation locale, un pour à la relocalisation (ou détection des fermetures de boucle) ainsi qu'un dédié à l'ajustement de faisceaux global, activé en particulier, si une boucle a été détectée.

S'appuyant sur [Strasdat 2012], l'ajustement de faisceaux global est réalisé en paramétrant les poses sur $\mathbb{SIM}3$ afin de gérer la dérive de l'échelle. De plus ils présentent une relocalisation robuste basée sur l'utilisation de sac de mots visuels ([Gálvez-López 2012]), facilitant la fermeture de boucle et améliorant la robustesse du suivi. Dans la version ORB-SLAM2, ce système a été étendu pour des dispositifs stéréo et pour des caméras RGB-D [Mur-Artal 2016]. Dans des travaux plus récents, [Mur-Artal 2015b] présente une méthode de reconstruction dense probabiliste où la profondeur inverse est filtrée directement entre images-clés. Enfin il propose d'intégrer les capteurs inertiels afin d'obtenir un SLAM visuel-inertiel robuste estimant l'échelle absolue de la scène [Mur-Artal 2017].

L'ensemble des travaux cités précédemment sont des méthodes utilisant des informations éparses de l'image, généralement des points d'intérêts ou des segments, occultant alors une grande partie de l'information disponible. Les méthodes directes reposent sur l'alignement direct, utilisant tout ou partie des pixels de l'image. Ces approches "d'alignement d'image direct" permettent d'éviter l'étape coûteuse d'extraction de descripteurs en utilisant directement les pixels de l'image. La transformation entre les deux poses caméra est alors estimée en minimisant l'erreur photométrique.

LSD-SLAM [Engel 2014] inclut une méthode d'alignement semi-dense afin d'estimer le déplacement en minimisant l'erreur photométrique. Cette méthode se restreint uniquement aux régions de l'image où l'intensité du gradient est suffisamment élevée, simplifiant ainsi l'alignement et le coût calculatoire en occultant les parties de l'image où l'information est négligeable. L'utilisation d'un grand nombre de pixels des images permet l'estimation d'une carte de profondeur semi-dense pour chaque image clé et d'obtenir ainsi une carte précise et fiable de l'environnement.

SVO [Forster 2015] propose une méthode d'odométrie visuelle semi-directe, utilisant dans un premier temps une méthode d'alignement minimisant une erreur photométrique de patch entre les images, puis minimisant ensuite une erreur de reprojection. Cette méthode, à l'instar de LSD-SLAM, permet d'éviter l'extraction de primitives et le calcul des descripteurs associés. La carte est paramétrée à partir d'un filtre probabiliste associé à chaque amer initialisé dans une image-clé. Ce filtre utilise une loi uniforme cumulée à une loi gaussienne afin de modéliser la profondeur des amers d'une image-clé. L'utilisation de ces deux lois permet de gérer les faux-positifs qui obéissent généralement à une loi uniforme, contrairement aux mesures correctes qui sont distribuées selon une loi normale autour de la vrai profondeur. Les auteurs arrivent à atteindre plus de 300 fps sur un ordinateur commun, et 70 fps sur système embarqué. Cette haute fréquence est cependant essentielle au bon fonctionnement de l'algorithme, l'approche par alignement semi-directe échoue pour l'estimation de mouvements trop rapides de la caméra entre deux images. Se basant sur cette méthode, [Pizzoli 2014] présente une reconstruction dense en temps réel, se rapprochant des résultats obtenus dans [Newcombe 2011].

Rappelons que, bien que nous n'ayons décrit que le SLAM visuel monoculaire, étant donné notre contexte, il existe de nombreux méthodes SLAM adaptées à différents types de capteurs. Ainsi, certains SLAM utilisent des bancs multi-caméras, des caméras RGB-D, des lidars ou encore des caméras ayant des optiques particulières (catadioptriques, fish-eyes).

La fusion des données visuelles et inertielles est également très étudiée, aussi bien pour les méthodes par filtrage [Mourikis 2007], RT-SLAM [Roussillon 2011], C-SLAM [Gonzalez 2013] que pour les méthodes basée optimisation [Forster 2015, Mur-Artal 2017, Leutenegger 2015].

Nous nous intéressons dans nos travaux aux méthodes éparses, pour leur coût calculatoire moindre et leur meilleure robustesse aux effets des mouvements rapides sur les images acquises par une caméra OD.

1.4 Contributions

Ce manuscrit présente les contributions de cette thèse résumées dans les points suivants :

- L"influence de la technique d'obturateur, global (OG) ou local (OD), sur les algorithmes de SLAM est étudiée. Des jeux de données réelles et synthétiques sont créés afin de comparer le comportement d'un algorithme standard de SLAM, appliqué sur des séquences d'images issues de caméras OD et OG ayant des points de vue et des optiques similaires.
- Une méthode d'interpolation par B-Splines utilisant des points de contrôle répartis non uniformément dans le temps est proposée. Cette méthode est utilisée pour la représentation en temps continu de la trajectoire caméra, et donc pour la modélisation des caméras OD.
- Des méthodes dynamiques de génération de points de contrôle des B-Splines sont développées, permettant la création de ces points où ils sont nécessaires sur la trajectoire.
- Une méthode d'optimisation spatio-temporelle des points de contrôle est également proposée qui intègre les horodatages de ces points de contrôle dans une démarche d'optimisation.
- Une méthode de reconstruction en intérieur est développée, qui s'appuie sur des jeux de données épurés et des modèles de pièces simples afin de reconstruire la structure d'une pièce.

Outre ces contributions scientifiques, un ensemble d'applications pour l'aménagement d'intérieur ont été réalisées. De plus, un système complet de réalité altérée a été développé permettant de virtuellement vider intégralement une pièce. Notons que ces travaux sur la suppression de meubles ont été développés dans le cadre d'un projet collaboratif regroupant, Innersense, le LAAS-CNRS et l'IRIT : la problématique de “remplissage” par rendu synthétique, des zones vidées dans l'image a été traitée par l'IRIT.

Les travaux réalisés durant cette thèse ont donné lieu à trois publications scientifiques internationales [Vandeportaele 2017],[Gohard 2018a],[Gohard 2018b] qui se concentrent sur les approches développées pour les caméras OD.

1.5 Organisation du manuscrit

Le chapitre 2 détaille les notions et outils nécessaires à la compréhension des travaux effectués. Le chapitre 3 se concentre sur l'influence des caméras à obturateur déroulant sur les algorithmes de SLAM en comparant les trajectoires obtenues sur des images issues de caméras OD et OG. Le chapitre 4 expose un modèle proposé pour une caméra OD utilisant des B-Splines non uniformes, et détaille l'intégration de ce modèle dans une méthode SLAM exploitant l'ajustement de faisceaux. De plus, des méthodes d'optimisation de l'horodatage des points de contrôle de B-Splines permises par notre modèle non uniforme sont introduites. Une méthode de reconstruction pour les environnements d'intérieur est proposée chapitre 5 : elle permet d'estimer simplement la structure d'une pièce. Finalement, le chapitre 6 présente les applications développées utiles pour l'aménagement d'intérieur, intégrant en partie les travaux réalisés. Un bilan des travaux est finalement dressé et différentes perspectives et travaux futurs envisageables sont énoncés.

Chapitre 2

Notions de base

Sommaire

2.1 Algèbre de Lie 21
2.1.1 Variété, géométrie Riemanienne et groupe de Lie 22
2.1.2 Application logarithmique 25
2.1.3 Groupe Special Orthogonal 25
2.1.4 Groupe Special Euclidien 28
2.2 Modèle de caméra à obturateur global 30
2.2.1 Paramètres intrinsèques 30
2.2.2 Paramètres extrinsèques 32
2.2.3 Paramètres de distorsions 32
2.3 Géométrie épipolaire et recouvrement de l'information 3D 33
2.3.1 Homographie 33
2.3.2 Matrice fondamentale et matrice essentielle 34
2.3.3 Triangulation 34
2.4 Interpolation 35
2.4.1 Interpolation linéaire 35
2.4.2 Courbes de Bézier 36
2.4.3 B-Spline 37
2.4.4 B-Spline cumulative 38
2.5 Interpolation de poses caméra 39
2.5.1 interpolation linéaire SLERP 39
2.5.2 Interpolation sur la variété 40
2.5.3 B-Splines cumulatives cubiques sur SE3 40
2.6 Outils d'optimisation 42
2.6.1 Introduction à l'ajustement de faisceaux 43
2.6.2 Linéarisation et minimisation de l'erreur 44
2.6.3 Optimisation sur la variété 46
2.6.4 Représentation par graphe 47
2.7 Conlusion 49

2.1 Algèbre de Lie

L'une des tâches principales de la réalité augmentée consiste à localiser la caméra dans l'espace. Il est nécessaire d'utiliser une paramétrisation intuitive

pour l'orientation et la position de ces caméras. Plusieurs paramétrisations sont disponibles et possèdent des propriétés différentes pouvant être utiles ou au contraire contre-indiquées selon les applications. La pose caméra est représentée par une transformation rigide comprenant une translation et une rotation. Une transformation rigide appliquée à un objet 3D rigide suppose la préservation des angles, distances et orientations relatives de cet objet. La translation est généralement exprimée par un vecteur 3D a dans un espace euclidien de dimension 3 $\mathbf{a} \in \mathbb{R}3$. En revanche, la rotation peut s'exprimer de différentes manières. Les premières paramétrisations sont données par Euler, avec les angles d'Euler, définies par 3 angles qui expriment trois rotations consécutives sur les 3 axes du repère monde. Cette paramétrisation est problématique car elle nécessite de connaître l'ordre dans lequel les rotations sont effectuées, l'espace des rotations n'étant pas commutatif. Une autre paramétrisation usuelle est donnée par la matrice des rotations. Cette représentation n'est pas sujette à des singularités, mais nécessite néanmoins 9 paramètres qui s'avèrent coûteux à évaluer dans un cadre d'optimisation. La représentation matricielle est en revanche très utilisée pour sa simplicité de composition avec des points 3D ou d'autres matrices de rotation par simple multiplication. Les quaternions (voir pour un résumé [Solà 2017]) peuvent également être utilisés sous leur forme unitaire pour la représentation des rotations. Un quaternion est défini par 4 paramètres, et peut être représenté comme un point à la surface d'une hypersphère normalisée à 4 dimensions. Trois paramètres du quaternion définissent un axe, et le quatrième défini l'angle de rotation autour de cet axe. Enfin, la paramétrisation sur la variété pour les rotations $\mathbb{SO}3$ a également gagné en popularité pour sa formulation minimale utile aux problématiques d'optimisation. Pour des problèmes de SLAM ou d'odométrie, on utilise généralement en concurrence plusieurs paramétrisations selon l'application, une paramétrisation minimale pour l'optimisation (quaternion, Euler, Rodrigues) et une paramétrisation par matrice de rotation pour les calculs et changement de repère.

Les rotations ne peuvent être modélisées dans un espace vectoriel Euclidien. Elles sont définies comme un groupe de Lie. Ces groupes de Lie ont fortement gagné en popularité au cours des dernières décennies. Divers ouvrages traitant du sujet sont disponibles mais s'adressent à un public mathématiquement averti [Chirikjian 2012],[Renaud 2016]. Des travaux introduisant et détaillant de manière plus intuitive ces groupes sont également disponibles [Blanco 2014],[Strasdat 2012],[Eade 2013].

2.1.1 Variété, géométrie Riemanienne et groupe de Lie

Une variété est une entité définie localement euclidienne mais possédant une structure globale plus complexe. L'étude de ces entités est généralement appelée géométrie Riemannienne, la géométrie des surfaces, qui considère un

espace local Euclidien et un espace global courbe. L'exemple le plus courant et intuitif étant l'étude de la surface d'une sphere. A chaque point de la surface de la sphere peut être associé un plan tangent, approximant localement la surface. Ce plan tangent est une approximation locale, plus on s'éloigne du point initial en parcourant le plan, plus l'erreur d'approximation est élevée (voir figure 2.1). En effet la structure globale de la sphère n'a rien à voir avec celle d'un plan. L'approximation par un plan tangent local est plus intuitif si la sphère est suffisamment grande, comme pour le cas de la Terre, où l'on perçoit localement le sol comme étant plan, alors que la surface globale est courbe.

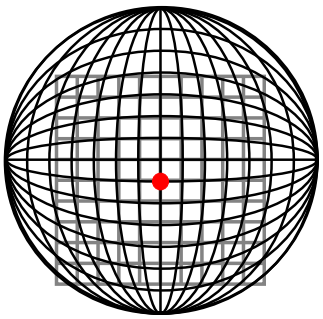


FIGURE 2.1 – Illustration de l'erreur d'approximation du plan tangent défini en un point (en rouge) par rapport à la surface de la sphere. Plus on s'éloigne du point central, plus l'erreur d'approximation est élevée.

On s'intéresse ici à un type particulier de variétés possédant de bonnes propriétés pour modéliser les rotations et les transformations rigides, les groupes matriciels de Lie. Un groupe de lie est un groupe algébrique qui est également une variété différentielle (ou variété différentiable). Ces groupes matriciels de Lie sont une sous catégorie des groupes de Lie.

2.1.1.1 Groupes algébriques

Les groupes de Lie sont des groupes algébriques. Soit un groupe $(G, e, \oplus)$ composé d'un ensemble G incluant un élément neutre $e \in G$ et d'une opération de composition $\oplus : (G \oplus G) \in G$, alors :

— $\forall g, h, i \in G \; g \oplus (h \oplus i) = (g \oplus h) \oplus i$ (associativité)
— $\forall g \in G \; g \oplus e = e \oplus g = g$ (élément neutre)
— $\forall g \in G$ il existe un et un seul $h \in G$ tel que $g \oplus h = h \oplus g = e$ (Élément inverse unique)

Un groupe est qualifié de commutatif uniquement si l'ensemble de ses éléments sont commutatifs, ce qui n'est pas le cas des groupes représentant les rotations. Dans le cas des groupes matriciels de Lie utiles pour représenter les transformations rigides, l'opérateur associé est la multiplication matricielle, qui n'est pas commutatif pour tout élément $g \neq e$. Notons également que la

multiplication par l'élément neutre e, la matrice identité $\mathbb{I}_{n\times n}$ dans le cas des groupes matriciels, est commutative.

2.1.1.2 Espace tangent

La plupart des groupes de Lie ne sont pas commutatifs, particulièrement les deux groupes qui nous intéressent, $\mathbb{SO}3$ le groupe des rotations et $\mathbb{SE}3$ le groupe des transformations rigides. En revanche, comme précisé précédemment, l'opérateur matriciel est commutatif pour l'élément neutre . Ainsi, un élément $g \in G$ proche de l'identité peut être défini sur un espace commutatif Euclidien. On parle alors d'espace tangent d'un groupe de Lie défini à l'identité.

Cette espace tangent permet la composition des éléments d'un groupe par simple addition Dans le cas des groupes matriciels de Lie, on parle d'algèbre de Lie pour définir ce plan tangent.

Soit un chemin $C(t)$ sur la variété engendrée par G, avec $C(0) = \mathbf{y}$, alors le vecteur tangent $\mathbf{x}$ à ce chemin au point $\mathbf{y}$ s'exprime en dérivant $C(t)$ pour $t \to 0$:

$$\mathbf{x} = \frac{\delta}{\delta t} C(t)|_{t=0} \tag{2.1}$$

L'ensemble des vecteurs tangents d'un groupe de Lie G en un point $\mathbf{y}$ engendre un espace tangent plan. De plus, l'ensemble des vecteurs tangents à l'identité d'un groupe G engendre $\mathfrak{g}$, l'espace tangent à l'identité.

2.1.1.3 Application exponentielle

L'application exponentielle permet d'associer un élément de l'espace tangent avec un élément de la variété dont est issu cet espace. L'application exponentielle est une généralisation de la fonction exponentielle adaptée aux matrices $exp(\mathbf{X}) : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$:

$$exp(\mathbf{X}) = \sum_{n=0}^{\infty} \frac{\mathbf{X}^n}{n!} \tag{2.2}$$

Cette application possède des propriétés analogues à la fonction exponentielle sur $\mathbb{R}$, en particulier :

$$\frac{\delta}{\delta t} exp(t\mathbf{X}) = \mathbf{X} exp(t\mathbf{X}) = exp(t\mathbf{X})\mathbf{X} \tag{2.3}$$

Notons également les propriétés suivantes qui seront utiles par la suite (voir [Strasdat 2012],[Solà 2017],[Chirikjian 2012]) :

— $exp(\mathbf{0}) = \mathbb{I}$

— $exp(\mathbf{X})^{-1} = exp(-\mathbf{X})$

Pour résumer formellement, l'application exponentielle permet de faire correspondre (de projeter) un élément d'un espace tangent à l'identité $\mathfrak{g}$ vers le groupe matriciel de Lie associé $\mathbb{G}$:

$$\mathbf{X} \in \mathfrak{g} \Rightarrow exp(\mathbf{X}) \in \mathbb{G} \tag{2.4}$$

2.1.2 Application logarithmique

L'application exponentielle admet une relation inverse sur les groupes de Lie étudiés. Cette relation inverse est caractérisée par l'application logarithmique ayant les propriétés suivantes :

$$exp(log(\mathbf{\Omega})) = \mathbf{\Omega} \tag{2.5}$$
$$log(exp(\mathbf{X})) = \mathbf{X} \tag{2.6}$$

Notons que pour le cas des rotations, l'application exponentielle n'est pas injective car elle fait correspondre l'ensemble des rotations de l'espace tangent sur l'espace des rotations comprises sur la sphère 2π. Ainsi un vecteur tangent représentant une rotation d'un angle de plus de $2pi$ se verra reprojetée sur l'espace des rotations avec un angle de $[-2\pi, 2\pi]$, d'où le fait qu'un élément du groupe des rotations possède plus d'un antécédent sur l'espace tangent. Par la suite, et tant que ce ne sera pas clairement explicité, les angles des rotations définies seront restreintes sur l'intervalle $[-2\pi, 2\pi]$, rendant les équations précédentes valides.

2.1.3 Groupe Special Orthogonal

Le groupe special orthogonal $\mathbb{SO}3$, ou groupe des rotations 3D est un sous groupe du groupe orthogonal $\mathbb{O}3$. Ce dernier groupe comprend également les réflexions, transformations qui ne préservent pas la rigidité des objets auxquels elles sont appliquées, et n'est donc pas adapté à la représentation des transformations rigides.

Le groupe des rotations 3D est généralement représenté par l'ensemble des matrices de rotation 3D $\mathbf{R} \in \mathbb{SO}3$. Il est soumis aux propriétés suivantes :

— Les rotations préservent la norme des vecteurs (ou points) auxquels elles sont appliquées

$$\|\mathbf{g} \oplus \mathbf{p}\| = \|\mathbf{p}\|, \forall \mathbf{g} \in \mathbb{SO}3, \mathbf{p} \in \mathbb{R}3 \tag{2.7}$$

— Les rotations préservent l'angle entre les vecteurs

$$(\mathbf{g} \oplus \mathbf{p}) \cdot (\mathbf{g} \oplus \mathbf{v}) = \mathbf{p} \cdot \mathbf{v}, \forall \mathbf{g} \in \mathbb{SO}3, \mathbf{p}, \mathbf{v} \in \mathbb{R}3 \tag{2.8}$$

— Les rotations préservent l'orientation relative des vecteurs

$$(\mathbf{g}\oplus\mathbf{p})\times(\mathbf{g}\oplus\mathbf{v}) = \mathbf{g}\oplus\mathbf{w} \iff \mathbf{p}\times\mathbf{v} = \mathbf{w}, \forall \mathbf{g} \in \mathbb{SO}3, \mathbf{p}, \mathbf{v}, \mathbf{w} \in \mathbb{R}3 \quad (2.9)$$

Il est également possible de représenter ce groupe par les quaternions, qui définissent un sous-groupe particulier $\mathbb{SU}2$. Pour plus de détails sur les propriétés des quaternions, voir [Solà 2017].

Les matrices de rotation $\mathbf{R} \in \mathbb{R}_{3\times3}$ utilisées ici pour représenter le groupe des rotations obéissent à la propriété d'orthogonalité qui découle des propriétés de $\mathbb{SO}3$:

$$\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbb{I}_3 \quad (2.10)$$

A partir de cette propriété et de la définition de l'élément neutre, on a donc :

$$\mathbf{R}^T = \mathbf{R}^{-1} \quad (2.11)$$

Et sachant la propriété de conservation de l'orientation relative des vecteurs, on a :

$$det(\mathbf{R}) = 1 \quad (2.12)$$

2.1.3.1 Algèbre de Lie de SO3

Considérons un chemin $\mathbf{C}(t)$ sur $\mathbb{SO}3$, avec t défini sur un intervalle réel tel que $t \in [a, b]; a, b \in \mathbb{R}$ et $\mathbf{C}(0) = \mathbb{I}_3$. L'ensemble des matrices $\mathbf{C}(t)$ sont par définition orthogonales :

$$\mathbf{C}(t)\mathbf{C}(t)^T = \mathbb{I}_3 \quad (2.13)$$

En dérivant cette équation on obtient :

$$\frac{\delta\mathbf{C}(t)}{\delta t}(\mathbf{C}(t)^T) + \mathbf{C}(t)\frac{\delta(\mathbf{C}(t))^T}{\delta t} = 0 \quad (2.14)$$

Le vecteur tangent à l'identité de $\mathbf{C}$ est obtenu en dérivant par rapport à t autour de $t = 0$, $(\mathbf{C}(0) = I)$:

$$\left.\frac{\delta\mathbf{C}(t)}{\delta t}\right|_{t=0} + \left.\frac{\delta(\mathbf{C}(t))^T}{\delta t}\right|_{t=0} = 0 \quad (2.15)$$

L'espace tangent $\mathfrak{so}3$ est donc engendré par l'ensemble des vecteurs tangents $\mathbf{\Omega} := \frac{\delta\mathbf{C}(t)}{\delta t}|_{t=0}$ respectant l'équation $\mathbf{\Omega} + \mathbf{\Omega}^T = 0 \iff \mathbf{\Omega} = -\mathbf{\Omega}^T$. Les éléments de l'espace tangent sont donc anti-symétriques. L'espace tangent $\mathfrak{so}3$ est alors engendré par les 3 matrices de bases suivant appelés générateurs de $\mathbb{SO}3$:

$$\frac{\delta \mathbf{R}_x(t)}{\delta t}|_{t=0} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -sin(0) & -cos(0) \\ 0 & cos(0) & -sin(0) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} = \mathbf{G}_0 \tag{2.16}$$

$$\frac{\delta \mathbf{R}_y(t)}{\delta t}|_{t=0} = \begin{pmatrix} -sin(0) & 0 & cos(0) \\ 0 & 0 & 0 \\ -cos(0) & 0 & -sin(0) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} = \mathbf{G}_1 \tag{2.17}$$

$$\frac{\delta \mathbf{R}_z(t)}{\delta t}|_{t=0} = \begin{pmatrix} -sin(0) & -cos(0) & 0 \\ cos(0) & -sin(0) & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \mathbf{G}_2 \tag{2.18}$$

L'espace tangent étant tridimensionnel, il est possible d'utiliser une représentation vectorielle minimale $\omega \in \mathbb{R}^3$ pour les éléments de l'espace tangent. Il est alors nécessaire d'introduire les opérateurs permettant de passer de $\mathbb{R}^3$ vers $\mathfrak{so}3$ et inversement. Ainsi on définit deux opérateur, l'opérateur *vee* $(\cdot)^\vee_{\mathfrak{so}3} : \mathbb{R}^3 \to \mathfrak{so}3$, et *hat* $\hat{\cdot}_{\mathfrak{so}3} : \mathbb{R}^3 \to \mathfrak{so}3$:

$$(\mathbf{\Omega})^\vee_{\mathfrak{so}3} = \frac{1}{2} \begin{pmatrix} \mathbf{\Omega}_{2,1} - \mathbf{\Omega}_{1,2} \\ \mathbf{\Omega}_{0,2} - \mathbf{\Omega}_{2,0} \\ \mathbf{\Omega}_{1,0} - \mathbf{\Omega}_{0,1} \end{pmatrix} \tag{2.19}$$

$$\hat{\omega}_{\mathfrak{so}3} = \begin{pmatrix} 0 & -w_2 & w_1 \\ w_2 & 0 & -w_0 \\ -w_1 & w_0 & 0 \end{pmatrix} := [\omega]_\times \tag{2.20}$$

2.1.3.2 Application exponentielle sur so3

L'application exponentielle pour le groupe spécial orthogonal $\mathbb{SO}3$ est donné par la formule de Rodrigues[Rodrigues 1840] qui à l'avantage de posséder une forme finie (pour la démonstration, voir [Renaud 2016]) :

$$exp([\omega]_\times) = \begin{cases} I + [\omega]_\times + \frac{1}{2}[\omega]^2_\times = I, & pour\ \theta \to 0 \\ I + \frac{\sin(\theta)}{\theta}[\omega]_\times + \frac{1-\cos(\theta)}{\theta^2}[\omega]^2_\times, & sinon \end{cases} \quad avec\ \theta = \|\omega\|_2 \tag{2.21}$$

La paramétrisation minimale de la rotation par les 3 paramètres du vecteur ω possède une interprétation géométrique proche de celle des quaternions. L'amplitude de la rotation θ est définie par la norme du vecteur tangent $\|\omega\|_2$. Similairement aux quaternions, la matrice $\mathbf{R} = exp(\hat{\omega})$ appliquée à un point effectue une rotation autour d'un axe défini par $\frac{\omega}{\theta}$.

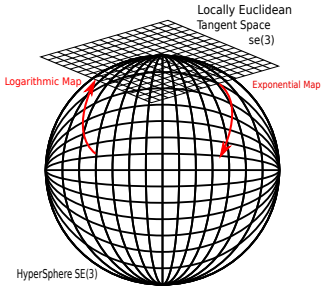


FIGURE 2.2 – Cette figure illustre le concept d'application exponentielle et logarithmique. Ici l'espace associé à un groupe de Lie est schématisé par une sphère. L'application logarithmique permet de passer d'un élément de la surface vers un élément du plan tangent défini en un point. L'application exponentielle à l'inverse permet de projeter un élément du plan tangent sur la variété (surface).

2.1.3.3 Application logarithmique sur so3

Similairement à l'application exponentielle, l'application logarithmique possède une forme finie sur $\mathbb{SO}3$:

$$log(\mathbf{R}) = \begin{cases} \frac{1}{2}(\mathbf{R} - \mathbf{R}^T) = 0, & pour\ d \to 1 \\ \frac{\arccos(d)}{2\sqrt{1-d^2}}(\mathbf{R} - \mathbf{R}^T), & pour\ d \in [-1, 1] \end{cases} \quad avec\ d = \frac{1}{2}(trace(\mathbf{R}) - 1) \tag{2.22}$$

Une formulation légèrement différente est donnée dans [Eade 2013], qui est simplement une réécriture possédant certains avantages calculatoires dans des cas particuliers.

2.1.4 Groupe Special Euclidien

Le groupe spécial euclidien $\mathbb{SE}3$ est le groupe des transformations rigides 3D qui sont généralement représentées par une matrice de transformation T ayant une composante en rotation $\mathbf{R} \in \mathbb{SO}3$ et en translation $a \in \mathbb{R}3$.

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{a} \\ \mathbf{0}^T & 1 \end{pmatrix}, \mathbf{T}_{wc} \in \mathbb{SE}3, \mathbf{R} \in \mathbb{SO}3, \mathbf{a} \in \mathbb{R}^3 \tag{2.23}$$

Le groupe spécial euclidien est donc par définition :

$$\mathbb{SE}3 = \mathbb{SO}3 \times \mathbb{R}3 \tag{2.24}$$

Les opérations et propriétés du groupe $\mathbb{SE}3$ sont analogues à celles du groupe $\mathbb{SO}3$:

— L'élément neutre est la matrice identité :

$$e_{\mathbb{SE}3} = \mathbf{I}_4 \in \mathbb{R}_{4\times4} \tag{2.25}$$

— La composition de deux éléments $\mathbf{g}, h \in \mathbb{SE}3$ est obtenue par multiplication matricielle :

$$\mathbf{g} \oplus h = \begin{pmatrix} \mathbf{R}_g & \mathbf{a}_g \\ \mathbf{0}^T & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R}_h & \mathbf{a}_h \\ \mathbf{0}^T & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R_g R}_h & \mathbf{a_g} + \mathbf{R_g a_h} \\ \mathbf{0}^T & 1 \end{pmatrix} \tag{2.26}$$

— L'inverse d'une matrice de transformation est donc :

$$g^{-1} = \begin{pmatrix} \mathbf{R}_g^T & -\mathbf{R}_g^T \mathbf{a}_g \\ \mathbf{0}^T & 1 \end{pmatrix} \tag{2.27}$$

2.1.4.1 Algèbre de Lie de SE3

L'algèbre de Lie correspond à l'ensemble des matrices 4×4 exprimant une translation différentielle (vitesse) et une rotation. Par définition :

$$\mathfrak{se}3 = \mathfrak{so}3 \times \mathbb{R}3 \tag{2.28}$$

Il est donc engendré par l'ensemble des vecteurs tangents issus des générateurs de $\mathbb{SO}3$ ainsi que 3 générateurs pour la translation :

$$\mathbf{G}_0 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \mathbf{G}_1 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \mathbf{G}_2 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.29}$$

$$\mathbf{G}_3 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \mathbf{G}_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \mathbf{G}_5 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.30}$$

Un élément de $\mathfrak{se}3$ est alors représenté par une composition linéaire de ces générateurs :

$$\begin{aligned} (\omega, \mathbf{a})^T \in \mathbb{R}^6 \\ \omega_0 * \mathbf{G}_0 + \omega_1 * \mathbf{G}_1 + \omega_2 * \mathbf{G}_2 + a_0 * \mathbf{G}_3 + a_1 * \mathbf{G}_4 + a_2 * \mathbf{G}_5 \in \mathfrak{se}3 \end{aligned} \tag{2.31}$$

Dans la suite, on notera $(\mathbf{w}, \mathbf{a}) \in \mathfrak{se}3$ par commodité, la multiplication par les générateurs étant sous-entendue.

2.1.4.2 Application exponentielle sur se3

L'application exponentielle sur les éléments de l'espace tangent $\mathfrak{se}3$ se résume par la formule suivante :

$$exp(\mathbf{\Omega}) = \begin{pmatrix} exp(\hat{\omega}) & \mathbf{Va} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \tag{2.32}$$

Avec :

$$\mathbf{V} = \begin{cases} I + \frac{1}{2}[\omega]_\times + \frac{1}{6}[\omega]_\times^2 = I, & pour\ \theta \to 0 \\ I + \frac{1-\cos(\theta)}{\theta^2}[\omega]_\times + \frac{\theta-\sin(\theta)}{\theta^3}[\omega]_\times^2, & sinon \end{cases} \quad avec\ \theta = \|\omega\|_2 \tag{2.33}$$

2.1.4.3 Application logarithmique sur se3

Similairement à l'application exponentielle, l'application logarithmique s'opère sur la rotation de la même manière que pour $\mathbb{SO}3$, et la composante en translation est multipliée par l'inverse de la matrice V définie précédemment :

$$log(\mathbf{T}) = \begin{pmatrix} log(\mathbf{R}) & \mathbf{V}^{-1}\mathbf{a} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \tag{2.34}$$

La matrice $\mathbf{V}$ admet également une forme finie de son inverse :

$$\begin{aligned} A &= \frac{\sin(\theta)}{\theta} \\ B &= \frac{1-\cos(\theta)}{\theta^2} \\ \mathbf{V}^{-1} &= \mathbf{I}_3 - \frac{1}{2}[\omega]_\times + \frac{1}{\theta^2}(1 - \frac{A}{2B})[\omega]_\times^2 \end{aligned} \tag{2.35}$$

2.2 Modèle de caméra à obturateur global

Les modèles de caméra à obturateur global sont les modèles de caméra usuellement utilisés dans la plupart des algorithmes de vision par ordinateur. Ce modèle de caméra est le modèle sténopé (ou modèle trou d'épingle) qui régit la formation d'une image par un modèle linéaire (voir fig. 2.3).

Ce modèle de caméra est représenté par un ensemble de paramètres définissant les propriétés internes (paramètres intrinsèques) et la position dans l'espace (paramètres extrinsèques) de la caméra. Ce modèle suppose une caméra ayant une optique et un capteur parfait, qui n'est pas le cas des capteurs réels. C'est pourquoi il est généralement enrichi par un modèle de distorsions gérant les problématique classique des capteurs et des lentilles.

2.2.1 Paramètres intrinsèques

Nous choisissons ici comme convention le coin haut-gauche de l'image comme origine de l'image, l'axe u balayant la largeur de l'image de gauche

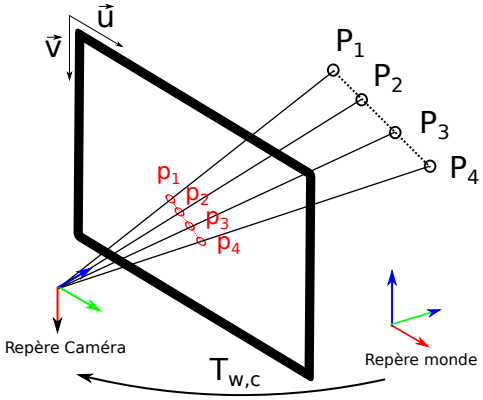


FIGURE 2.3 – Illustration de la projection de points 3D dans l'image par le modèle sténopé. La projection d'une ligne dans l'espace se projette en une ligne dans l'image. Les points sont exprimés dans un repère monde, et la caméra est exprimée par sa transformation du repère caméra vers le repère monde (World from Camera).

à droite et l'axe v la hauteur de l'image de haut en bas.

Les paramètres intrinsèques de la caméra définissent une projection perspective transformant un point 3D de l'espace $\mathbf{P}_c = [XYZ]^T$ exprimé dans le repère caméra en un point image $\mathbf{p} = \begin{bmatrix} x \\ y \end{bmatrix}$, reliant ainsi le repère caméra au repère associé du plan image :

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.36}$$

La distance focale f exprime la distance entre le centre optique de la caméra et le plan image. Cette première transformation nous donne une projection d'un point 3D sur un plan image 2D en coordonnées métriques. Ces coordonnées métriques doivent être converties en coordonnées pixelliques afin d'être utilisables. Cette conversion est effectuée par un produit matricielle :

$$\begin{bmatrix} p_u \\ p_v \\ 1 \end{bmatrix} = \begin{bmatrix} kx & 0 & u_0 \\ 0 & ky & v_0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.37}$$

avec u_0 et v_0 les coordonnées du point principal décrivant l'intersection entre l'axe optique et le plan image et k_x et k_y définissant le nombre de pixels par unité de longueur pour les direction x et y.

Ces deux transformations sont généralement exprimées par la matrice des paramètres intrinsèques K, avec $f_x = f * k_x$ et $f_y = f * k_y$:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.38}$$

2.2.2 Paramètres extrinsèques

Afin de projeter des points 3D du repère monde dans le plan image, il est nécessaire de transformer ces points dans le repère caméra. Nous supposons un système de coordonnées orthonormé direct pour la caméra, avec l'axe x pointant vers la droite, l'axe y vers le bas et l'axe z vers l'avant. La pose de la caméra est représentée par une matrice de transformation rigide $\mathbf{T}_{wc} \in \mathbb{SE}3$ composée d'une rotation $\mathbf{R}$ et d'une translation $\mathbf{a}$:

$$\mathbf{T}_{wc} = \begin{pmatrix} \mathbf{R} & \mathbf{a} \\ \mathbf{0}^T & 1 \end{pmatrix}, \mathbf{T}_{wc} \in \mathbb{SE}3, \mathbf{R} \in \mathbb{SO}3, \mathbf{a} \in \mathbb{R}^3 \tag{2.39}$$

Appliquer cette matrice à un point permet de passer du repère caméra au repère monde . Un point en coordonnées homogènes dans le repère monde $P_w \in \mathbb{R}4$ est obtenu dans le repère caméra par :

$$\mathbf{P}_c = \mathbf{T}_{wc}^{-1} * \mathbf{P}_w = \mathbf{R}_{cw}^T \mathbf{P}_w - \mathbf{R}_{cw}^T \mathbf{a}_{cw} \tag{2.40}$$

Le modèle de mesure globale de l'observation d'un point dans l'image est donc défini par :

$$\hat{\mathbf{p}}_m = \pi(\mathbf{K} * \mathbf{T}_{cw} * \mathbf{P}_w) \tag{2.41}$$

Avec $\pi(\mathbf{P}) = \frac{1}{Z} * \begin{bmatrix} X \\ Y \end{bmatrix}$ la fonction projetant un point de l'espace projectif $\mathbb{P}^2$ vers le plan image.

2.2.3 Paramètres de distorsions

Le modèle de caméra sténopé décrit précédemment est un modèle idéal. La manufacture de l'optique des caméras, et parfois leurs principes même (caméra fish eye, grand angle) engendrent des distorsions s'exprimant par des déformations dans l'image. Deux types de distorsions peuvent alors être pris en compte. La distorsion radiale et la distorsion tangentielle. La distorsion radiale produit une déformation rendant les droites courbes dans l'image. Elle est modélisée par une fonction liant les coordonnées réelles mesurée $\mathbf{p}_m = [x_m y_m]$ dans l'image aux coordonnées idéales théoriques $p = [xy]$

$$\begin{bmatrix} x_m \\ y_m \end{bmatrix} = D_r(r) * \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.42}$$

Avec $D_r(r)$ la fonction de distorsion radiale, fonction de la distance radiale définie par $r = \sqrt{x^2 + y^2}$. Elle est définie par un polynôme de degré 6 :

$$D_r(r) = (1 + k_{r1} * r^2 + k_{r2} * r^4 + k_{r3} * r^6) \tag{2.43}$$

La distorsion tangentielle est induite par un mauvais alignement du capteur photo-sensible de la caméra par rapport à la lentille [Brown 1966]. Elle est modélisée par un vecteur s'additionnant à la projection image parfaite (qui peut s'apparenter à un changement de repère), et est définie par deux polynôme de degré 2 :

$$\mathbf{D}_t(r) = \begin{bmatrix} 2 * k_{t1} * x * y + k_{t2} * (r^2 + 2x^2) \\ k_{t1} * (r^2 + 2y^2) + 2 * k_{t2} * x * y \end{bmatrix} \tag{2.44}$$

Les modèles de distorsions radiales et tangentielles peuvent être combinés et appliqués aisément à la projection théoriques du point afin d'en obtenir une projection plus réaliste :

$$\begin{bmatrix} x_m \\ y_m \end{bmatrix} = D_r(r) * \begin{bmatrix} x \\ y \end{bmatrix} + \mathbf{D}_t(r) \tag{2.45}$$

2.3 Géométrie épipolaire et recouvrement de l'information 3D

Le SLAM, afin de pouvoir localiser le système courant, nécessite une estimée de la carte. L'environnement, pour être initialement reconstruit par triangulation, nécessite la localisation des caméras. Nous sommes ici en présence d'un problème type "oeuf ou la poule". Il est cependant possible d'estimer simultanément la transformation relative entre deux caméras ainsi qu'un nuage de point 3D de l'environnement à partir de correspondances 2D entre les images. Ces méthodes nécessitent la présence d'un mouvement en translation entre les caméras et dégénèrent pour des translations nulles.

2.3.1 Homographie

Une première méthode consiste à considérer l'environnement comme étant partiellement planaire [Faugeras 1988]. Dans ce cas particulier, un point 3D $\mathbf{P}_k$ appartenant à un environnement plan Π projeté dans deux images ι et ι' en deux pixels $\mathbf{p}_k$ et $\mathbf{p}'_k$ suit la loi suivante :

$$\mathbf{p}'_k = \mathbf{H}\mathbf{p}_k => \mathbf{P}_k \in \Pi \tag{2.46}$$

L'homographie $\mathbf{H}$ est une application bijective et peut être décomposée en une suite d'opérations appliquée à un pixel $\mathbf{p}_k$. Soit un plan 3D exprimé sous

la forme d'une normale n et d'une distance à l'origine d, $\Pi = [nd]$. Soit $\mathbf{K}_\iota$ et $\mathbf{K}_{\iota'}$ les matrices des paramètres intrinsèques des caméras ι et ι', $\mathbf{R}$ la rotation et $\mathbf{a}$ la translation de la caméra ι vers ι'. Ainsi la matrice $\mathbf{H}$ peut être calculée par la formule suivante :

$$\mathbf{H} = \mathbf{K}_{\iota'}(\mathbf{R} - \frac{\mathbf{a}n^T}{d})\mathbf{K}_\iota^{-1} \tag{2.47}$$

Cette matrice possède 8 degrés de liberté et s'exprime par une matrice 3×3. L'ensemble de ses paramètres est estimé à partir de 4 correspondances de points dans 2 images. L'utilisation de plus de correspondances sur-contraint le problème et nécessite alors des méthodes approchées (moindre-carrés). De plus amples détails concernant l'estimation sont données dans [Hartley 2004, Dubrofsky 2009].

2.3.2 Matrice fondamentale et matrice essentielle

Dans le cas d'un environnement non planaire, les projection $\mathbf{p}_k$ et $\mathbf{p}'_k$ d'un même point 3D dans deux images obéissent à la géométrie épipolaire, généralement résumé par la contrainte épipolaire (ou loi de coplanarité) :

$$\mathbf{p}'_k\mathbf{F}\mathbf{p}_k = 0 \tag{2.48}$$

Avec $\mathbf{F}$ la matrice fondamentale, et $det(\mathbf{F}) = 0$. Cette matrice peut être obtenue en connaissant les paramètres intrinsèques et extrinsèques des deux caméras :

$$\mathbf{F} = \mathbf{K}_{\iota'}[\mathbf{a}]_\times\mathbf{R}\mathbf{K}_\iota^{-1} \tag{2.49}$$

Diverses méthodes utilisant 8,7 ou même 5 correspondances de point [Nistér 2004] peuvent être utilisées pour l'estimation des paramètres de $\mathbf{F}$.

Si uniquement les paramètres intrinsèques des caméras sont connus, il est possible de pré-multiplier les projections $\mathbf{p}_k$ et $\mathbf{p}'_k$ par les matrices $\mathbf{K}$ correspondantes, et simplifier ainsi la contrainte épipolaire en :

$$\mathbf{p}'_k\mathbf{E}\mathbf{p}_k = 0 \tag{2.50}$$

Avec $\mathbf{E}$ la matrice essentielle définie par :

$$\mathbf{E} = [\mathbf{a}]_\times\mathbf{R} \tag{2.51}$$

2.3.3 Triangulation

A partir de correspondances 2D entre les images et du déplacement inter-caméra estimé, il est possible de reconstruire un nuage de points 3D par triangulation. La triangulation suppose la reconstruction d'un point 3D par l'inter-

section des rayons optiques issus des pixels mis en correspondance. Les rayons ne s'intersectent pas toujours en présence de bruit, le point 3D choisi est alors le point équidistant entre les deux rayons (voir fig. 2.4).

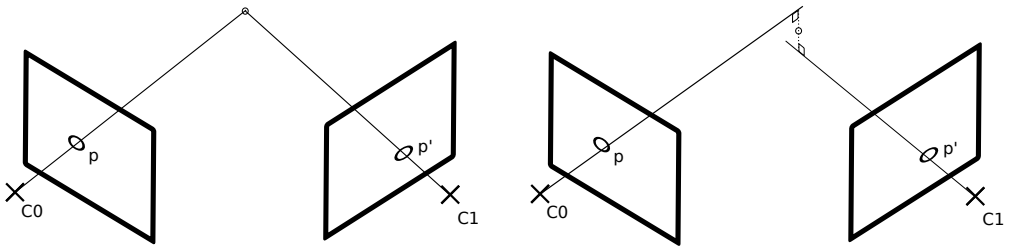


FIGURE 2.4 – Illustration du concept de la triangulation de points par intersection de rayons optiques pour des données parfaites (gauche) et bruitées (droite).

Dans les approches probabilistes, le point 3D est reconstruit avec une incertitude issue du bruit du capteur et de l'incertitude sur la pose (voir fig. 2.5). Le problème de la triangulation d'un point est résolu selon le cas par des méthodes linéaires (DLT) ou par des méthodes de minimisation dans les cas bruités. Une description et une comparaison de ces méthodes sont fournies dans [Hartley 1997].

Les différentes méthodes décrites dans cette section permettent à partir de correspondances 2D entre deux images d'obtenir une estimée initiale de l'environnement et des paramètres extrinsèques de la caméra. Cependant, ces approches supposent une pose unique de la caméra pour toutes les lignes des images, ce qui n'est pas le cas pour les images issues des caméra OD. L'estimation de la pose relative entre deux caméras à partir de correspondances disposées sur une ligne d'image étant impossible, les poses caméras associées aux lignes d'une image OD sont obtenues par des méthodes d'interpolation.

2.4 Interpolation

L'interpolation est utilisée afin d'estimer des données à un échantillonnage supérieur aux informations initiales. Généralement, elle permet d'obtenir une fonction continue à partir d'informations discrètes, et dans le cas géométrique, d'obtenir une courbe continue à partir d'un ensemble de points. Ces points sont appelés points de contrôle, et appartiennent généralement à un espace euclidien $\mathbb{R}^n$.

2.4.1 Interpolation linéaire

L'interpolation linéaire est la plus simple des méthodes d'interpolation. Elle suppose une trajectoire linéaire entre les points de contrôle. Ainsi, l'interpolation entre deux points $\mathbf{P}_0$ et $\mathbf{P}_1$ pour un temps t est définie par :

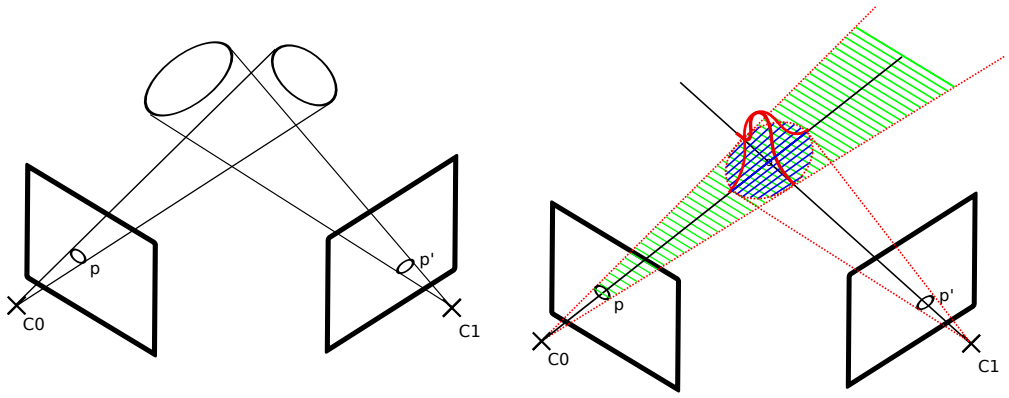


FIGURE 2.5 – Illustration de l'utilisation de données bruitées pour la triangulation. Le point 3D (gauche) est considéré comme une densité de probabilité issue de l'intersection des deux cônes d'incertitude associés aux rayons optiques (droite).

$$\begin{aligned} \mathbf{p}(0) &= \mathbf{P}_0 \\ \mathbf{p}(1) &= \mathbf{P}_1 \\ \mathbf{p}(t) &= (1-t)\mathbf{P}_0 + t\mathbf{P}_1 \end{aligned} \tag{2.52}$$

Elle est donc continue et dérivable à l'infini par morceau. Cependant elle n'est pas dérivable aux points de contrôle. Des méthodes d'ordre supérieur utilisent plus de points de contrôle pour l'interpolation entre deux points de contrôle afin de garantir une continuité voulue sur ce tronçon. Ainsi il est possible de faire correspondre une fonction quadratique, cubique ou autres à un sous ensemble des points de contrôle afin d'obtenir la continuité souhaitée. Cependant, garantir la continuité pour un changement de tuples de points de contrôle est nécessaire. C'est pour ce principe qu'ont été développées les courbes de Bézier et les Splines.

2.4.2 Courbes de Bézier

Les courbes de Bézier sont des courbes polynomiales paramétriques définies par un ensemble de points de contrôle. Elles permettent l'interpolation d'une courbe d'un degré prédéfini, où chaque point de la courbe est calculé par une somme pondérée des points de contrôle. Les $n+1$ points de contrôle $\mathbf{P}_0 \dots \mathbf{P}_n$ servant à définir la courbe de Bézier sont appelés le polygone de contrôle de Bézier. Un point $\mathbf{p}$ interpolé au temps t est alors calculé par :

$$\mathbf{p}(t) = \sum_{i=0}^{n} B_i^n(t)\mathbf{P}_i \tag{2.53}$$

avec $t \in [0,1]$ et $B_i^n(t)$ les polynômes de Berstein. Ces polynômes de Bern-

stein sont une partition de l'unité, qui expriment l'influence associée à chaque point de contrôle en fonction d'un paramètre t. Ils sont définis par la formule :

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \tag{2.54}$$

avec $\binom{n}{i}$ le coefficient binomiale associé :

$$\binom{n}{i} = C_i^n = \frac{n!}{i!(n-i)!} \tag{2.55}$$

Les polynômes de Bernstein possèdent plusieurs propriétés :

— Ils sont une partition de l'unité, i.e. :$\sum_{i=0}^{n} B_i^n(t) = 1$
— Ils sont toujours définis positifs : $\forall i \in \{1, ..., n\}, B_i^n(t) \geq 0$
— Ils sont symétriques : $\forall i \in \{1, ..., n\}, B_i^n(t) = B_{n-i}^n(1-t)$

Ils peuvent également être définis par la formule de récurrence suivante :

$$B_i^n(t) = \begin{cases} (1-t) * B_i^{n-1}(t) & if \quad i = 0 \\ (1-t) * B_i^{n-1}(t) + t * B_{i-1}^{n-1}(t) & \forall i \in \{1, ..., n-1\} \\ t * B_{i-1}^{n-1}(t) & si \quad i = 1 \end{cases} \tag{2.56}$$

2.4.3 B-Spline

Les B-Splines sont une généralisation des courbes de Bézier. Elles peuvent être pensées comme des combinaisons de courbes de Bézier, et sont définies localement (à support local). Le degré de la spline est défini par le polynôme de plus haut degré. Si tout les polynômes définissant la spline partagent le même degré, on parle de B-Spline uniforme.

Une courbe B-Spline de degré k est définie par une somme pondérée de fonctions de base $B_{i,k}(t)$ qui sont C^{k-1} continues. Étant donné un ensemble de points de contrôle P_i, une courbe B-Spline est définie de la manière suivante :

$$p(t) = \sum_{i=0}^{n} B_{i,k}(t) P_i \tag{2.57}$$

Les fonctions de bases sont définies par la relation de récurrence de De Boor ([de Boor 1972]) :

$$B_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(t) \tag{2.58}$$

Avec la condition d'arrêt définie par :

$$B_{i,1} = \begin{cases} 1 & \text{si} \quad t_i < t < t_{i+1} \\ 0 & \text{sinon} \end{cases} \tag{2.59}$$

Nous nous intéressons dans cette thèse principalement aux B-Splines cubiques. Le degré cubique du polynôme d'interpolation est adapté pour une trajectoire effectuée par une caméra tenue à la main. La continuité $C2$ induite par ce degré de B-Spline permet l'interpolation de la vitesse et de l'accélération au cours du temps.

2.4.3.1 Représentation matricielle des B-Splines cubiques

La formulation récursive du calcul des fonctions de base pour les B-Splines reste peu pratique à implémenter et difficile à visualiser. Pour une interpolation cubique, $k = 4$ fonctions de base sont définies non nulles. Il est possible de représenter les fonctions de bases sous forme matricielle[Yang 2009] [Kim 1995], afin d'obtenir une formule d'interpolation appropriée. Soit $u(t) = \frac{t-t_i}{\Delta_t} \in [0,1]$ le temps intermédiaire entre deux points de contrôle ayant pour horodatages t_i et t_{i+1}, et $\Delta_t = t_{i+1} - t_i$ la variation temporelle entre ces deux points de contrôle. L'interpolation par B-Spline cubique peut alors se formuler sous forme matricielle par :

$$\mathbf{P}(t) = \overbrace{\frac{1}{6}\begin{pmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}}^{\mathbf{M}} \begin{pmatrix} 1 \\ u(t) \\ u(t)^2 \\ u(t)^3 \end{pmatrix} \begin{pmatrix} \mathbf{P}_k \\ \mathbf{P}_{k+1} \\ \mathbf{P}_{k+2} \\ \mathbf{P}_{k+3} \end{pmatrix} \tag{2.60}$$

2.4.4 B-Spline cumulative

Les B-Splines cumulatives sont une reformulation des B-splines. Contrairement à la formulation classique, elles expriment un point interpolé sur la courbe non pas par une somme pondérée de points de contrôle, mais par une somme de variation de points de contrôle appliquée à un point de contrôle initial.

Les fonctions de bases cumulatives sont calculées récursivement à partir des fonctions de bases des splines :

$$\tilde{B}_{i,k} = \sum_{j=i}^{n} B_{j,k} \tag{2.61}$$

$$\tilde{B}_{i,k} = B_{i,k} + B_{i+1,k} + B_{i+2,k} + B_{i+3,k} + \cdots + B_{i+(n-i),k} \tag{2.62}$$

Notons que les fonctions de bases sont nulles pour $i > k - 1$. [Kim 1995] donne une définition prenant en compte cette notion dans sa formulation :

$$\tilde{B}_{i,k} = \begin{cases} \sum_{j=i}^{i+k} B_{i,k} & \text{si} \quad t_i < t < t_{i+k-1} \\ 1 & \text{si} \quad t \geq t_{i+k-1} \\ 0 & \text{si} \quad t \leq t_i \end{cases} \tag{2.63}$$

A partir de cette formulation, on peut réécrire la définition des fonctions de base pour les B-Spline cumulatives cubiques :

$$\tilde{B}_{i,k} = B_{i,k} + B_{i+1,k} + B_{i+2,k} + B_{i+3,k} \tag{2.64}$$

En se basant sur les équation 2.60 et 2.64, la forme cumulative de la matrice de coefficients se calcule en réalisant une addition cumulative sur les lignes r_j de $\mathbf{M}$:

$$\mathbf{C} = \begin{pmatrix} r_1 + r_2 + r_3 + r_4 \\ r_2 + r_3 + r_4 \\ r_3 + r_4 \\ r_4 \end{pmatrix} = \frac{1}{6} \begin{pmatrix} 6 & 0 & 0 & 0 \\ 5 & 3 & -3 & 1 \\ 1 & 3 & 3 & -2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.65}$$

2.5 Interpolation de poses caméra

La pose d'une caméra est représentée par une matrice de transformation $\mathbf{T}_{w,c} \in \mathbb{SE}3$ composée d'une translation et une rotation. Cette transformation appliquée à gauche à un élément de $\mathbb{P}^3$ permet de passer d'un repère caméra vers un repère monde (World from Camera). L'espace des translations $\mathbb{R}3$ est un espace euclidien pour lequel les opérations de compositions se réalisent par addition/soustraction. En revanche, l'espace des rotations $\mathbb{SO}3$ est un espace courbe défini localement, non Euclidien, sur lequel la géométrie Riemannienne s'applique. Les points de contrôle utilisés dans ce cas sont appelés poses de contrôles d'indice i $\mathbf{T}_{w,i} \in \mathbb{SE}3$ (abrégées PC dans la suite). L'indice c associé au repère caméra de cette transformation est abandonné par la suite par soucis de clarté.

2.5.1 interpolation linéaire SLERP

Certains auteurs ([Forssén 2010], [Hedborg 2012], [Ramadasan 2015]) choisissent d'interpoler séparément la rotation et la translation. Dans le cas de [Hedborg 2012], la translation est interpolée linéairement dans $\mathbb{R}3$, et la rotation est interpolée en utilisant SLERP(Spherical Linear Interpolation). Cette méthode interpole la rotation sur la surface de l'hypersphère (l'espace des quaternions unitaires) le long d'un grand arc reliant les points représentant les deux rotations entre lesquels on souhaite interpoler.

Soit deux quaternions $\mathbf{q}_0$ et $\mathbf{q}_1$, l'interpolation sphérique linéaire de la rotation $\mathbf{q}(t)$ entre $\mathbf{q}_0$ et $\mathbf{q}_1$ pour un temps $t \in [0, 1]$ est donnée par :

$$\mathbf{q}(t) = \mathbf{q}_0(\mathbf{q}_0^{-1}\mathbf{q}_1)^t \tag{2.66}$$

L'avantage de cette approche est de pouvoir avoir une distribution des points de contrôle pour la rotation et la translation différente, adapté selon les cas.

2.5.2 Interpolation sur la variété

[Steven Lovegrove 2013] suggère d'interpoler directement dans l'espace des transformations rigides $\mathbb{SE}3$ en se basant sur les équations de [Kim 1995]. La formulation cumulative des B-Splines permet d'adapter les formules d'interpolation aux espaces non Euclidien.

Ainsi, similairement à [Furgale 2012][Furgale 2015], ils obtiennent une formulation en temps-continu de la trajectoire, qui permet d'estimer la pose d'une caméra à tout instant de la trajectoire, utile pour la modélisation de caméra à obturateur déroulant. De plus, [Steven Lovegrove 2013] propose également l'utilisation de B-Spline cubique, permettant de conserver une continuité C2 le long de la trajectoire. Cette continuité autorise l'estimation de l'accélération et de la vitesse angulaire de la caméra, et admet la possibilité de la fusion de capteurs. Les auteurs utilisent ainsi l'interpolation des vitesses et accélérations comme modèle prédictif d'une centrale inertielle. Il est alors possible d'intégrer les données inertielles dans l'estimation de la trajectoire caméra en minimisant l'erreur entre la prédiction par interpolation et l'observation fournie par la centrale.

Dans le travail présenté ici, nous utiliserons principalement des B-Splines cumulatives cubiques afin de modéliser la trajectoire caméra.

2.5.3 B-Splines cumulatives cubiques sur SE3

Les matrices de transformations rigides utilisées pour représenter les poses de contrôle, de par leur appartenance à un espace non-Euclidien, ne peuvent être composées par simple addition comme c'est le cas pour les points 3D. En revanche, la formulation cumulative permet d'appliquer des variations de points de contrôle à une première pose de référence. Cette formulation, décrite dans la section précédente, peut alors s'adapter pour l'interpolation sur la variété de poses de contrôle.

Dans le cas des B-Splines cumulatives cubiques, les fonctions de bases cumulatives $\tilde{\mathbf{B}}(t)_k$ pondérant les variations de poses sont exprimées par la k^{ieme} composante du vecteur $\tilde{\mathbf{B}}(t)$ décrit par :

$$\tilde{\mathbf{B}}(t) = \frac{1}{6}\begin{pmatrix} 6 & 0 & 0 & 0 \\ 5 & 3 & -3 & 1 \\ 1 & 3 & 3 & -2 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ u(t) \\ u(t)^2 \\ u(t)^3 \end{pmatrix} \tag{2.67}$$

où $u(t) = \frac{t-t_{i+1}}{\Delta t}$ le temps intermédiaire entre les deux PC $\mathbf{T}_{w,i+1}, \mathbf{T}_{w,i+2}$ qui encadrent le temps d'interpolation. Δt définit l'intervalle temporel entre les PC, considéré constant. La différence entre les fonctions de base normales et les fonctions de base cumulatives est illustrée figure 2.6.

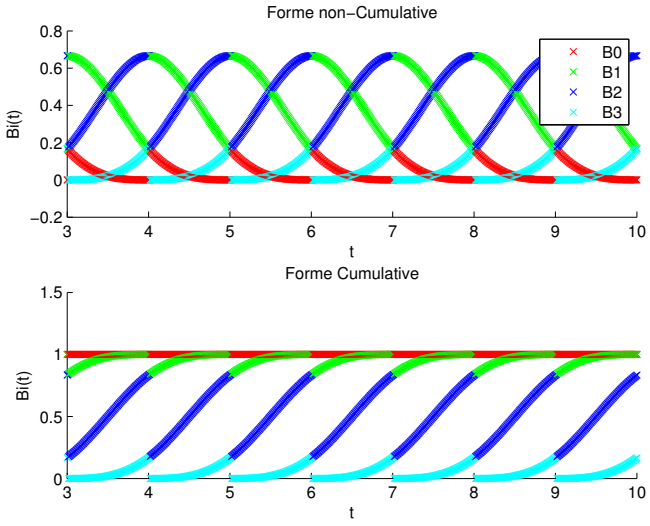


FIGURE 2.6 – Cette figure décrit l'évolution de la valeur des $k = 4$ fonctions de base non cumulatives $B_{0..k}(t)$(haut) et cumulatives $\tilde{B}_{0..k}(t)$(bas) en fonction du temps. A chaque fonction de base correspond une couleur (rouge,vert,bleue,cyan), illustrant l'influence d'une PC(haut) ou d'une variation de PC(bas) sur la trajectoire interpolée au cours du temps.

L'interpolation sur la variété $\mathbb{SE}3$ est définie comme une composition de variations de PC $\mathbf{\Omega}_j = log(\mathbf{T}_{w,j-1}^{-1}\mathbf{T}_{w,j})$ pondérées par les fonctions de base $\tilde{B}(t)$ appliquée à une pose de référence $\mathbf{T}_{w,i}$. Ainsi, dans le cas des B-Splines cumulatives cubiques $k = 4$, l'interpolation entre deux PC $\mathbf{T}_{w,i+1}$ et $\mathbf{T}_{w,i+2}$ nécessite les 4 PC avoisinantes $\mathbf{T}_{w,i..i+3}$. La fonction d'interpolation est alors exprimée par (voir [Steven Lovegrove 2013]) :

$$\mathbf{T}_{w,c}(t) = \mathbf{T}_{w,i}\prod_{j=i+1}^{i+k-1}\exp(\tilde{B}(t)_{j-1}\mathbf{\Omega}_j) \tag{2.68}$$

L'interprétation géométrique pour la formulation est la suivante. Sur chaque variation de pose de contrôle $\mathbf{T}_{w,j-1}^{-1}\mathbf{T}_{w,j}$ est appliqué l'opérateur lo-

garithmique afin de projeter le chemin sur l'espace tangent. A cette variation est ensuite appliquée la fonction de pondération, puis la variation pondérée sur l'espace tangent est à nouveau projeté sur la variété $\mathbb{SE}3$ par application exponentielle. Les variations étant à nouveau exprimées par des matrices de transformation rigide, elles sont ensuite composées entre elle par produit matriciel afin d'obtenir la pose interpolée.

2.5.3.1 Interpolation vitesse et accélération

Comme introduit précédemment, la continuité C^2 des B-Splines cubiques admet l'interpolation des vitesses et des accélérations par dérivation de l'équation 2.68 en fonction du temps.

Les fonctions de bases dérivées en fonction de t s'expriment par :

$$\dot{\tilde{\mathbf{B}}}(t) = \frac{1}{\Delta t}\mathbf{C}\begin{bmatrix}0\\1\\2u(t)\\3u(t)^2\end{bmatrix}, \ddot{\tilde{\mathbf{B}}}(t) = \frac{1}{\Delta t^2}\mathbf{C}\begin{bmatrix}0\\0\\2\\6u(t)\end{bmatrix} \tag{2.69}$$

En se basant sur l'équation 2.3 et en utilisant les règles de dérivation pour les compositions de fonctions, la dérivé première de la formule d'interpolation s'expriment de la manière suivante [Steven Lovegrove 2013] :

$$\dot{\mathbf{T}}_{w,c}(t) = \mathbf{T}_{w,i}(\dot{\mathbf{A}}_1\mathbf{A}_2\mathbf{A}_3 + \mathbf{A}_1\dot{\mathbf{A}}_2\mathbf{A}_3 + \mathbf{A}_1\mathbf{A}_2\dot{\mathbf{A}}_3) \tag{2.70}$$

Et la dérivé seconde :

$$\ddot{\mathbf{T}}_{w,c}(t) = \mathbf{T}_{w,i}\begin{pmatrix}\ddot{\mathbf{A}}_1\mathbf{A}_2\mathbf{A}_3 + \mathbf{A}_1\mathbf{A}_2\ddot{\mathbf{A}}_3 + \mathbf{A}_1\mathbf{A}_2\ddot{\mathbf{A}}_3)+\\2*(\dot{\mathbf{A}}_1\dot{\mathbf{A}}_2\mathbf{A}_3 + \dot{\mathbf{A}}_1\mathbf{A}_2\dot{\mathbf{A}}_3 + \mathbf{A}_1\dot{\mathbf{A}}_2\dot{\mathbf{A}}_3)\end{pmatrix} \tag{2.71}$$

$$\mathbf{A}_j = \exp(\tilde{B}(t)_j\boldsymbol{\Omega}_{j+i}), \dot{\mathbf{A}}_j = \mathbf{A}_j\dot{\tilde{B}}(t)_j\boldsymbol{\Omega}_{j+i} \tag{2.72}$$

$$\ddot{\mathbf{A}}_j = \dot{\mathbf{A}}_j\dot{\tilde{B}}(t)_j\boldsymbol{\Omega}_{j+i} + \mathbf{A}_j\ddot{\tilde{B}}(t)_j\boldsymbol{\Omega}_{j+i} \tag{2.73}$$

A l'aide de ces formulations, l'accélération linéaire et la vitesse angulaire au temps t peuvent être respectivement extraites des matrices $\ddot{\mathbf{T}}_{w,c}(t)$ et $\dot{\mathbf{T}}_{w,c}(t)$.

2.6 Outils d'optimisation

Le modèle de caméra ainsi que les méthodes de recouvrement de l'information 3D introduits dans la section précédente permettent l'obtention une estimée initiale de l'environnement ainsi que de la pose de caméra. Ces méthodes autorisent le recouvrement de la position relative entre uniquement deux caméras à partir de correspondances entre les images. Il est également possible d'obtenir des modèles utilisant trois ou quatre images (tenseur tri-focal,

quadri-focal). Ces méthodes ne sont cependant pas adaptées pour assurer une cohérence globale de le structure et de la localisation. Comme dit précédemment, deux méthodes existent dans le cas du SLAM. Les méthodes par filtrage qui marginalisent les informations précédentes en une estimée mise à jour avec une covariance associée, où les méthodes par optimisation qui minimisent une erreur de reprojection d'un ensemble d'observation. Cette méthode d'optimisation, nommée ajustement de faisceaux, est relativement ancienne dans le milieu de la photogrammétrie mais popularisée dans la communauté vision par [Triggs 1999]. Elle est désormais présentée dans de nombreuses ressources accessibles[Triggs 2000],[Hartley 2004],[Grisetti 2010].

2.6.1 Introduction à l'ajustement de faisceaux

Soit un ensemble d'images $I_1, I_2, ...I_i$ acquises par le biais d'une caméra monoculaire en déplacement dans une scène statique. Le but d'un SLAM visuel est d'estimer la trajectoire de cette caméra dans l'environnement, et donc de retrouver un ensemble de matrice de transformation $\mathbf{T}_{wc,1}$, $\mathbf{T}_{wc,2}$ ... $\mathbf{T}_{wc,i}$ ainsi que la géométrie de la scène définie par un ensemble de points (et éventuellement de segments) $\mathbf{P}_1$, $\mathbf{P}_2$ ... $\mathbf{P}_k$.

Étant donné un ensemble d'observations $\mathbf{p}_{i,k}$ décrivant l'observation d'un point $\mathbf{P}_k$ dans une image I_i avec pour transformation $\mathbf{T}_{wc,i}$, il est possible de réaliser une estimation jointe des paramètres de caméra $\mathbf{T}_{wc,i}$ et des points 3D $\mathbf{P}_k$, communément nommé ajustement de faisceaux.

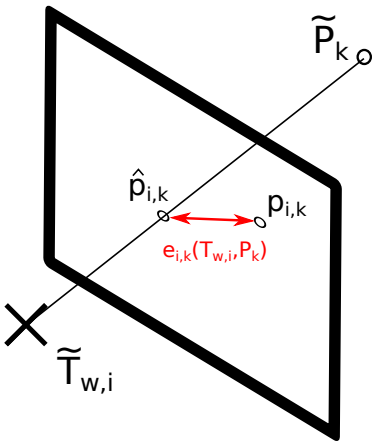


FIGURE 2.7 – L'erreur de reprojection $e_{i,k}$ est obtenue en calculant la distance entre $\hat{\mathbf{p}}_{i,k}$, la projection du point $\tilde{\mathbf{P}}_k$ en utilisant l'estimée de la pose caméra $\tilde{\mathbf{T}}_{w,i}$, et l'observation du point $\mathbf{p}_{i,k}$.

Ces paramètres sont estimés en minimisant une erreur de reprojection(voir figure 2.7), qui correspond à la distance entre la mesure d'un point $p_{i,k}$ et sa prédiction selon un modèle d'observation $\hat{p}_{i,k}$:

$$e_{i,k}(\mathbf{T}_{wc,i}, \mathbf{P}_k) = \mathbf{p}_{i,k} - \hat{\mathbf{p}}_{i,k}(\mathbf{T}_{wc,i}, \mathbf{P}_k) \tag{2.74}$$

Dans le cas monoculaire avec un obturateur global, la prédiction est obtenue en projetant le point 3D dans l'image à partir des paramètres intrinsèques et extrinsèques de la caméra :

$$\hat{\mathbf{p}}_{i,k}(\mathbf{T}_{wc,i}, \mathbf{P}_k) = \pi([\mathbf{K}|0]\mathbf{T}_{wc,i}^{-1}\mathbf{P}_k) \tag{2.75}$$

En assumant une distribution gaussienne de l'erreur de mesure, le problème d'estimation jointe s'exprime comme une somme des erreurs de reprojection pondérée par la covariance $\mathbf{Q}$ du modèle de mesure. La formulation de cette somme correspond à la fonction négative de vraisemblance (negative likelihood) :

$$F(\theta) = e^T\mathbf{Q}e = (e_{1,1}^T \cdots e_{i,1}^T e_{1,2}^T \cdots e_{i,k}^T)\mathbf{Q}\begin{pmatrix} e_{1,1}^T \\ \vdots \\ e_{i,1}^T \\ e_{1,2}^T \\ \vdots \\ e_{i,k}^T \end{pmatrix} \tag{2.76}$$

Où θ est l'ensemble des paramètres $\mathbf{T}_{wc,i}, P_k$, et Q est la matrice de covariance du modèle de mesure. On peut également l'exprimer de la manière suivante, avec l'ensemble des paires d'indices pour lesquels une contrainte d'observation existe :

$$F(\theta) = \sum_{(i,k)} \underbrace{e_{i,k}^T\mathbf{Q}_{i,k}e_{i,k}}_{F_{ik}} \tag{2.77}$$

Cette fonction exprime ainsi une mesure de la cohérence entre les observations et le modèle estimée. Plus la valeur de cette fonction est proche de zéro, plus le modèle est fidèle aux observations. Elle correspond donc à une fonction de coût qu'il est nécessaire de minimiser. Pour résoudre le problème d'estimation par maximum de vraisemblance, il est donc nécessaire de résoudre :

$$\hat{\theta} = \arg\min F(\theta) \tag{2.78}$$

2.6.2 Linéarisation et minimisation de l'erreur

Si une estimée initiale $\tilde{\theta}$ des paramètres θ est connue à priori, l'équation précédente de minimisation peut être résolue par les méthodes itératives de Gauss-Newton ou Levenberg-Marquardt. Ces méthodes d'optimisation non linéaires permettent de linéariser l'erreur et donc d'approximer l'erreur par

un développement limité à l'ordre 1 autour de l'estimé initial $\tilde{\theta}$:

$$e_{i,k}(\tilde{\theta}_i + \Delta\theta_i, \tilde{\theta}_k + \Delta\theta_k) = e_{i,k}(\tilde{\theta} + \Delta\theta) \simeq e_{i,k} + J_{i,k}\Delta\theta \tag{2.79}$$

Avec $J_{i,k}$ la jacobienne de l'erreur $e_{i,k}(\theta)$ évaluée en $\tilde{\theta}$. Le terme $e_{i,k} := e_{i,k}(\tilde{\theta})$ est utilisé afin d'alléger la notation. La linéarisation de la fonction de coût 2.77 s'obtient simplement par la somme des erreurs linéarisées 2.79 :

$$F(\tilde{\theta} + \Delta\theta) \simeq \sum_{(i,k)} (e_{i,k} + J_{i,k}\Delta\theta)^T \mathbf{Q}_{i,k}(e_{i,k} + J_{i,k}\Delta\theta) \tag{2.80}$$

$$= \sum_{(i,k)} \underbrace{e_{i,k}^T \mathbf{Q}_{i,k} e_{i,k}}_{c_{i,k}} + 2\underbrace{e_{i,k}^T \mathbf{Q}_{i,k} J_{i,k}}_{b_{i,k}}\Delta\theta + \Delta\theta^T \underbrace{J_{i,k}^T \mathbf{Q}_{i,k} J_{i,k}}_{H_{i,k}}\Delta\theta \tag{2.81}$$

$$= \sum_{(i,k)} c_{i,k} + 2b_{i,k}\Delta\theta + \Delta\theta^T H_{i,k}\Delta\theta \tag{2.82}$$

A partir de cette approximation locale du terme de l'erreur, nous pouvons réécrire l'équation 2.77 :

$$F(\tilde{\theta} + \Delta\theta_j) \simeq c + 2b\Delta\theta + \Delta\theta^T H\Delta\theta \tag{2.83}$$

Avec $c = \sum c_{i,j}$, $b = \sum b_{i,j}$, $H = \sum H_{i,j}$. Cette approximation linéaire locale de la fonction de coût permet alors une minimisation en $\Delta\theta$ par résolution d'un simple système linéaire :

$$H\Delta\hat{\theta} = -b \tag{2.84}$$

La solution linéarisée est ensuite obtenue en ajoutant l'incrément à l'estimée initiale

$$\hat{\theta} = \tilde{\theta} + \Delta\hat{\theta} \tag{2.85}$$

Les algorithmes de minimisation (Gauss-Newton, Levenberg-Marquardt) opèrent la linéarisation, la résolution du système pour l'incrément des paramètres(eq. 2.84) et la mise à jour des paramètres (eq. 2.85), puis itèrent ces trois opérations jusqu'à ce qu'une condition soit satisfaite (Erreur en dessous d'un certain seuil).

La plupart des opérations développées ici supposent que les données soient exprimées dans un espace Euclidien. Si cette condition est vrai pour l'expression des points 3D de la carte, ce n'est pas le cas pour les poses caméras exprimées dans $\mathbb{SE}(3)$. Il est donc nécessaire de procéder à une optimisation sur cette variété, problème abordé dans [Hertzberg 2008] et résumé dans [Grisetti 2010].

2.6.3 Optimisation sur la variété

La variété issue de l'espace des transformations admet un espace Euclidien local. Nous nous concentrons ici sur les poses caméras $\mathbf{T}_{wc,i}$ exprimées sur la variété $\mathbb{SE}(3)$. Ainsi les perturbations infinitésimales ε_i sur les paramètres des poses caméras sont définies dans un espace euclidien $\mathbb{R}6$. ε_i représente l'approximation linéarisée de l'incrément.

La composition de cette perturbation $\varepsilon \in \mathfrak{se}(3)$ avec une pose $\mathbf{T} \in \mathbb{SE}(3)$ n'est pas possible par une simple addition comme dans l'équation 2.85. Il est d'abord nécessaire de projeter la perturbation sur la variété, par le biais de l'application exponentielle (eq. 2.33). Notons que l'adaptation de l'équation 2.85 sur la variété peut se traduire par deux équations équivalentes et valides selon les conventions utilisées :

$$\hat{\theta} = \tilde{\theta} + \Delta\hat{\theta} \rightarrow \begin{cases} \hat{\mathbf{T}} = exp(\varepsilon)\mathbf{T} \\ \hat{\mathbf{T}} = \mathbf{T}exp(\varepsilon) \end{cases}$$

Les jacobiennes doivent désormais être évaluées par rapport à cette perturbation pour $\varepsilon = 0$. Nous ne détaillons pas ici le calcul de la jacobienne de l'erreur de coût en fonction des perturbations, et précisons simplement la jacobienne de $exp(\varepsilon)T$:

$$\frac{\delta exp(\varepsilon)\mathbf{T}}{\delta\varepsilon}\Big|_{\varepsilon=0} = \frac{\delta\mathbf{A}\mathbf{T}}{\delta\mathbf{A}}\Big|_{\mathbf{A}=\mathbf{I}_4=exp(\varepsilon)} \frac{\delta exp(\varepsilon)}{\delta\varepsilon}\Big|_{\varepsilon=\mathbf{0}} \tag{2.86}$$

$$[\mathbf{T}^T \otimes \mathbf{I}_3]\frac{\delta exp(\varepsilon)}{\delta\varepsilon}\Big|_{\varepsilon=\mathbf{0}} \tag{2.87}$$

Avec $\otimes$ le produit de Kronecker. Finalement la jacobienne sous forme matricielle s'exprime par :

$$\frac{\delta exp(\varepsilon)\mathbf{T}}{\delta\varepsilon}\Big|_{\varepsilon=\mathbf{0}} = \begin{pmatrix} 0_3 & -\mathbf{T}_{c1} \\ 0_3 & -\mathbf{T}_{c2} \\ 0_3 & -\mathbf{T}_{c3} \\ \mathbf{I}_3 & -\mathbf{T}_{c4} \end{pmatrix} \tag{2.88}$$

Avec $\mathbf{T}_{ci}$ les 3 premiers éléments de la colonne i de $\mathbf{T}$. Plus de détails concernant les jacobiennes et l'optimisation sur la variété sont données dans la section 10 de [Blanco 2014]. A partir des notions abordées précédemment, il est possible de réaliser un ajustement de faisceaux afin d'optimiser la position des caméras et des points de la carte. En revanche, une implémentation naïve provoque une augmentation du coût mémoire et de la complexité calculatoire insoutenable même pour des machines haut de gamme et pour des problèmes relativement restreint. Il est donc nécessaire d'aborder la manière de représenter le problème.

Les matrices du système à résoudre sont généralement très éparses, et tenir

compte de cette propriété des matrices permet de gagner un temps considérable et de rendre supportable de telles méthodes même pour des systèmes modestes disponibles sur le marché. De nombreuses implémentations libres prenant en compte les propriétés éparses des matrices sont disponibles tels que SBA [Lourakis 2009], g2o [Kuemmerle 2011], Bundler [Snavely 2006] ou encore Ceres [Agarwal 2010]. Nous choisissons de nous intéresser à g2o, et plus généralement au SLAM par optimisation de graphe. La représentation par un graphe d'interdépendance entre les observations permet de gérer de manière ad hoc la propriété éparse des matrices.

2.6.4 Représentation par graphe

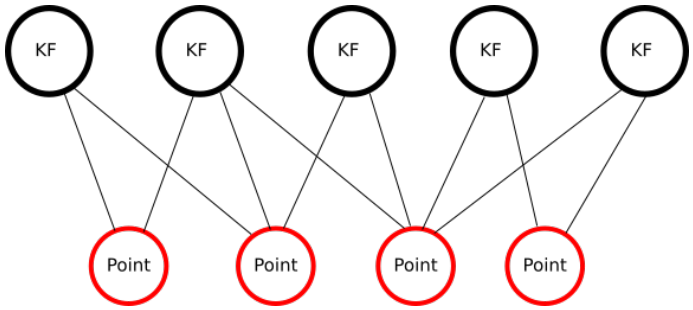


FIGURE 2.8 – Représentation d'un problème de SLAM par graphe. En noir les nœuds représentant des images clés, en rouge les points 3D de la carte. Une arête reliant un point 3D à une image clé signifie que ce point 3D est observé par cette image.

Le problème du SLAM peut être représenté par un graphe, où les nœuds représentent les paramètres à optimiser, et les arêtes décrivent les contraintes entre les nœuds. Cette représentation a été introduite pour la première fois dans le cadre du SLAM par [Lu 1997] et démocratisée plus récemment. Comme on peut l'observer sur la figure 2.8, les nœuds sont soit associés aux paramètres spatiaux d'un point 3D, où à la pose d'un capteur, en l'occurrence une caméra. Les arêtes ont un bruit Gaussien associé, et représentent les observations ou les mouvements qui contraignent spatialement les nœuds entre eux. On a donc généralement deux types d'arêtes (voir fig. 2.9) :

— Les arêtes d'observation, qui sont issues de données acquises par des capteurs extéroceptifs et qui relient donc la position du capteur à des amers.
— Les arêtes de mouvement, qui sont issues de données proprioceptives et qui relient donc deux nœuds de pose consécutifs.

Comme illustré figure 2.9, il est possible d'ajouter les paramètres intrinsèques du capteur à l'optimisation. En considérant ces paramètres unique sur l'ensemble d'une séquence, ce qui est envisageable dans le cas de la réalité

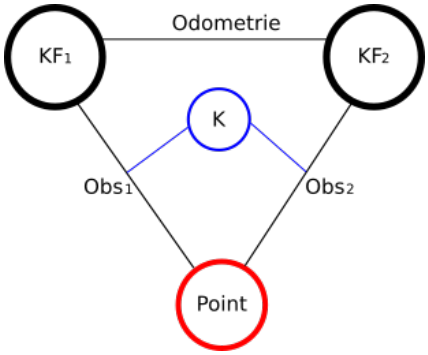


FIGURE 2.9 – Une représentation simple de l'observation d'un point 3D (rouge) par 2 images clé. La position relative des deux images clés est également contrainte par une arête de mouvement caractérisée par une donnée inertielle. Les paramètres intrinsèques de la caméra (bleu) peuvent également être ajoutés au graphe comme un nœud à optimiser, qui est alors relié par chaque arête d'observation.

augmentée, alors ce nœud est unique, et chaque arête d'observation relie alors une pose caméra, un point 3D et ce nœud des paramètres intrinsèques. Nous considérons dans notre cas que la caméra est étalonnée suffisamment précisément à l'avance, et nous choisissons de ne pas optimiser ces paramètres.

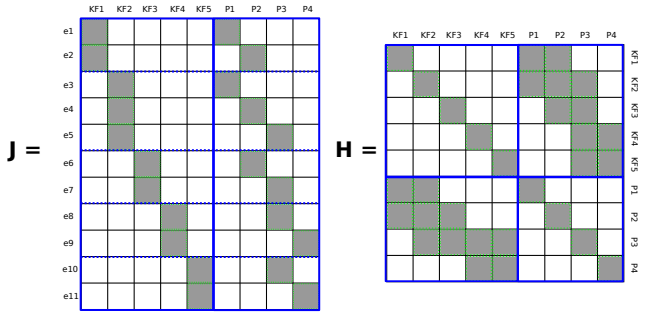


FIGURE 2.10 – Représentation des matrices jacobienne J (gauche) et hessienne H (droite) associées au graphe présenté dans la figure 2.8. Les zones grises correspondent aux zones non nulles.

La figure 2.10 représente les matrices jacobienne et hessienne issues du graphe de la figure 2.8. On peut d'ores et déjà constater que ces matrices sont éparses malgré un problème très restreint (5 images et 4 points). L'aspect creux de ces matrices augmente avec la taille du problème, si bien qu'il devient impossible et surtout inutile de stocker la matrice complète. Le coût de l'inversion de la matrice Hessienne issue d'un graphe à plusieurs milliers de nœuds pour la résolution du système est ingérable tel quel. La structure de la matrice Hessienne est heureusement connue à l'avance grâce à la connectivité du graphe, ce qui permet d'allouer la mémoire nécessaire avant le début du processus itératif. La matrice Hessienne est de plus symétrique définie posi-

tive, ce qui permet de ne stocker qu'une partie de la matrice. La résolution du système est effectuée par blocs connus à l'avance grâce à la connectivité du graphe. Diverses méthodes sont disponibles afin de réduire la complexité en prenant en compte la structure de ces matrices, comme les factorisations QR et de Cholesky [Dellaert 2006]. Ces méthodes sont largement abordées dans la littérature [Triggs 2000],[Triggs 1999],[Dellaert 2006] et ne seront pas détaillées ici.

L'utilisation d'un graphe permet donc d'obtenir une représentation simple et intuitive du problème de l'ajustement de faisceaux. L'utilisation d'un outil logiciel libre tel que g2o permet de très simplement implémenter un problème d'optimisation tant qu'il peut être représenté par un graphe.

2.7 Conlusion

Dans ce chapitre, un ensemble de notions avancées ont été présentées qui sont la base théoriques pour la localisation, l'interpolation de la caméra et la reconstruction d'un modèle géométrique de l'environnement. Cependant, toutes ces notions utilisent un modèle de caméra à obturateur global qui n'est pas adapté au présent contexte. La modélisation des caméra OD suppose l'interpolation de la pose caméra dans l'image et l'adaptation de la plupart des notions à ces caméras n'est pas aisé et entraîne un surcoût calculatoire. Nous proposons dans la suite d'étudier l'impact des distorsions OD sur les algorithmes de localisation et cartographie simultanées, et exposons les modèles de caméras OD existants dans la littérature.

Chapitre 3

Caméras à obturateur déroulant et modèles existants

Sommaire

3.1 Caméras à obturateur déroulant 51
3.1.1 Altérations photométriques 52
3.1.2 Déformations géométriques 54
3.2 Impact des caméras OD sur les algorithmes de vision . . . 54
3.2.1 Critère d'évaluation 55
3.2.2 Expérience sur données synthétiques 55
3.2.3 Expériences sur données réelles 59
3.2.4 Résultats du SLAM sur séquences réelles OG et OD 63
3.3 Modèles existants de caméras OD 68
3.3.1 Modèles de mouvement pour caméra OD 68
3.3.2 Modèle de projection pour caméra OD 70
3.4 Conclusion 72

3.1 Caméras à obturateur déroulant

Les caméras à obturateur déroulant (caméra Rolling Shutter) sont des caméras relativement peu étudiées dans la littérature par rapport à leur proéminence au sein du marché des appareils grand public. En effet, la majorité des caméras équipant les téléphones portables et les tablettes sont aujourd'hui des caméras à obturateur déroulant (abrégées caméras OD).

Ces caméras ont la particularité d'exposer les lignes de l'image à des temps différents (voir fig 3.1). Ainsi, une image issue d'une caméra à obturateur déroulant (abrégée image OD) peut être vue comme une combinaison d'images 1D, percevant l'état de la scène à un temps d'exposition et selon un point de vue différent. Ce principe d'exposition séquentielle des lignes entraîne deux types d'altérations de l'image par rapport aux caméras à obturateur global (abrégées caméras OG) :

— Des altérations photométriques lorsque les conditions d'illumination de la scène changent au cours de l'acquisition de l'image.
— Des déformations géométriques lors de mouvements relatifs entre la scène et la caméra.

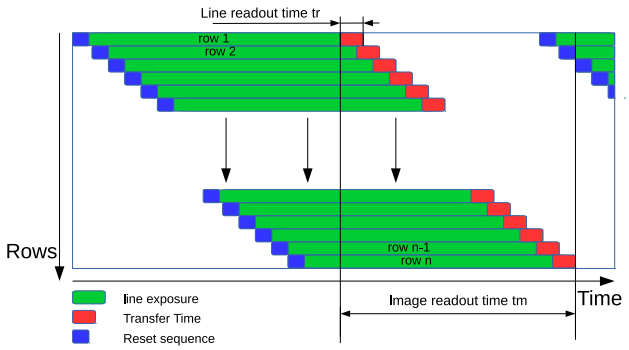


FIGURE 3.1 – Illustration de l'acquisition des lignes de l'image par une caméra à obturateur déroulant

3.1.1 Altérations photométriques

En environnement d'intérieur, les luminaires ont deux types de fréquences standardisées selon les régions, 60 Hz pour l'Amérique et 50Hz pour l'Europe. En réalité, la fréquence réelle est plus exactement de 100Hz pour les néons. En effet, le courant délivré par le secteur est polarisée et les néons n'étant pas polarisés s'allument alors à chaque crête minimale et maximale de la tension. La fréquence dédoublée est alors équivalente à 100Hz. Les variations d'intensité lumineuse sont invisibles à l'œil nu mais font apparaître des altérations photométriques dans les images OD. Ces altérations s'illustrent par des bandes plus sombres et plus claires dans l'image (voir figure 3.2). Elles sont causées par la différence entre la durée d'intégration d'une image de la caméra OD et la période de l'éclairage.

Ces altérations peuvent être outre-passées en ajustant la durée d'exposition des lignes de l'image à un multiple de la période de l'éclairage. Ces corrections de la durée d'exposition sont désormais disponibles et accessibles directement sur la plupart des plateformes mobiles (Option Anti-Bandwidth sur Android par exemple). Elles sont accessibles par le biais de deux options relatives à la fréquence électrique de la région (Anti-Bandwidth 60Hz ou 50Hz).

Ces altérations photométriques peuvent également être utilisées pour étalonner la caméra OD ([Meingast 2005]). Par exemple, en prenant l'image de la figure 3.2 obtenue par une caméra OD qui observe une texture globalement unie éclairée par un néon, il est possible d'analyser l'intensité moyenne des pixels sur les lignes (voir figure 3.2). Environ trois périodes sont observées, avec quatre crêtes basses correspondant aux quatre zones sombres sur l'image, et les trois crêtes hautes aux trois zones éclairées de l'image. Soustraire la valeur moyenne de l'intensité permet de recentrer la courbe autour de 0. Il est ensuite possible de faire correspondre une courbe sinusoïdale en optimisant les paramètres associés d'amplitude A, de pulsation du signal w, et de phase φ. Pour cela, il est nécessaire de minimiser l'erreur entre l'intensité moyenne

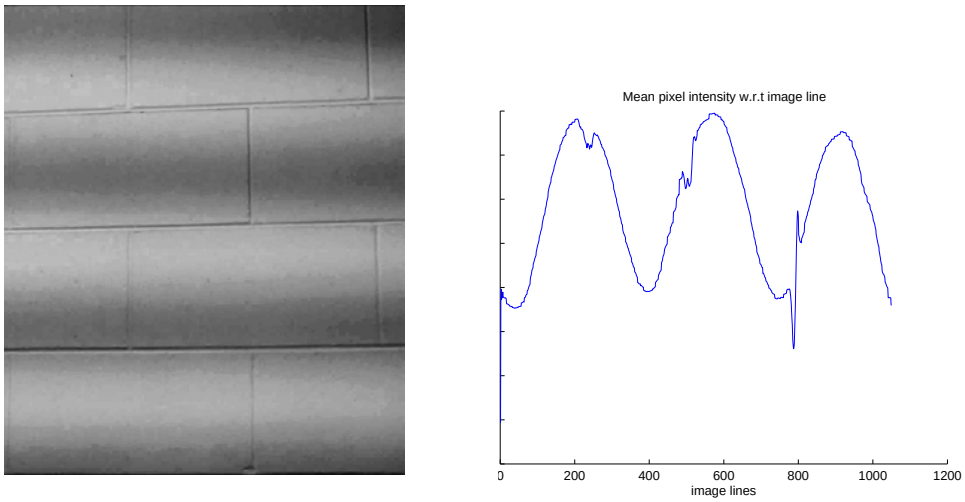


FIGURE 3.2 – A gauche une image acquise par une caméra à obturateur déroulant observant un mur illuminé par un éclairage d'intérieur (néon 100Hz). A droite l'intensité moyenne des pixels pour chaque ligne de l'image, où est observé un signal périodique de la valeur moyenne de l'intensité pixellique.

à la ligne i et la valeur du signal sinusoïdal au temps i défini par :

$$s(t) = A\sin(w * i + \varphi) \tag{3.1}$$

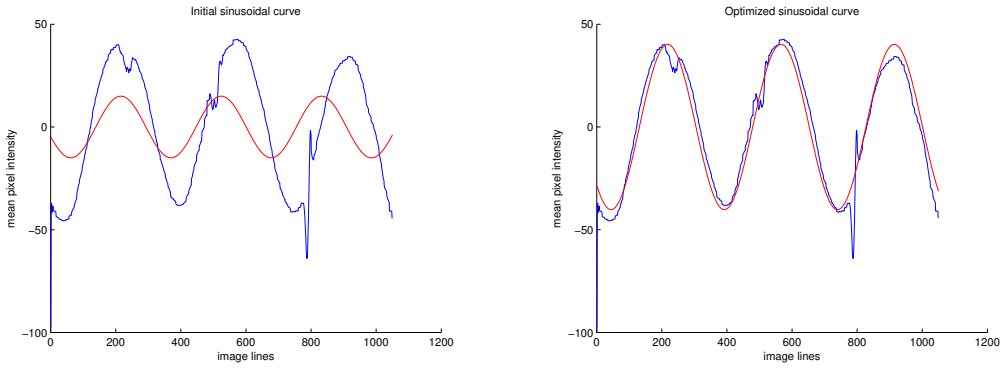


FIGURE 3.3 – A gauche, l'intensité moyenne pixellique (bleu) avec le signal sinusoïdal initial (rouge). A droite, le signal sinusoïdal après optimisation des paramètres par la méthode des moindres carrés.

En partant d'une estimée initiale satisfaisante, il est possible d'ajuster le signal sinusoïdal (voir fig. 3.3). Sur cet exemple, la pulsation obtenue après optimisation vaut $w = 0.0181$, ce qui correspond à période de $T_{im} = \frac{2\pi}{w} = 347$ pixels. L'image possède $rows = 1080$ lignes, et environ 3.11 périodes sont observables dans une image. La fréquence du luminaire est de 100Hz, soit une

période $T_{lum} = 0.01s$. Le temps de lecture de l'image OD r_t peut aisément être déduit :

$$r_t = T_{lum} * \frac{rows}{T_{im}} = \frac{1}{100} * \frac{1080}{347} = 0.0311s \tag{3.2}$$

La valeur est légèrement inférieure à la fréquence de la caméra ($30im/s \rightarrow 0.033s$) ce qui représente un résultat cohérent et attendu. En effet, en intérieur, l'éclairage est plus faible qu'en extérieur, ce qui nécessite un temps d'exposition plus long de chaque ligne de l'image, d'où l'utilisation de la quasi-totalité du temps disponible pour l'exposition. En vue d'obtenir des résultats plus précis, il est préférable de mettre en œuvre le protocole décrit dans [Meingast 2005]. Ce protocole suppose l'utilisation d'une diode dont la fréquence est parfaitement contrôlée, et d'ôter l'optique de la caméra afin de s'affranchir de toutes les contraintes pour étalonner uniquement le capteur. Il est également possible de synchroniser plusieurs caméras OD entre elles en utilisant des procédés similaires [Šmıd 2017].

Nous choisissons dans cette thèse de ne pas tenir compte de ces altérations photométriques étant donné qu'elles sont généralement éludées par les options d'exposition disponibles sur la plupart des plateformes mobiles. Les déformations géométriques sont en revanche plus problématiques et nécessitent l'utilisation d'un modèle adapté aux caméras OD.

3.1.2 Déformations géométriques

Dans le cadre d'une scène et d'une caméra statique, il n'y a pas de différence entre les images fournies par une caméra OD ou une caméra OG. En revanche, dans le cas d'un mouvement relatif entre la scène et la caméra, chaque ligne de l'image est une projection depuis un point de vue différent. Tandis que la pose camera $\mathbf{T}_{wc} \in \mathbb{SE}3$ est commune à tous les pixels d'une image OG, dans le cas d'un OD la pose varie pour chaque ligne de l'image.

Les paramètres extrinsèques d'une caméra OD sont donc dépendants du temps pour les différentes lignes d'une même image, et s'expriment sous la forme d'une matrice de transformation fonction du temps $\mathbf{T}_{wc}(t)$.

Dans la section suivante, l'impact de la non modélisation de la caméra OD est exposé en comparant les trajectoires estimées par un SLAM sur des images OD et des images OG.

3.2 Impact des caméras OD sur les algorithmes de vision

Les caméras OD, par leur conception différentes des caméras OG, génèrent des images faisant apparaître des déformations dans l'image. La plupart des



FIGURE 3.4 – Images intégrant des distorsions géométriques provoquées par l'obturateur déroulant lors de rapides mouvements relatifs entre la scène et la caméra. Images issues de [Li 2013] (gauche), et [Guo 2014] (droite).

algorithmes de SLAM et de vision ne prennent pas en compte l'utilisation de ce type de caméra.

Nous proposons dans cette section d'étudier l'influence d'une caméra OD sur des algorithmes de Vision par ordinateur, et particulièrement sur le SLAM. Dans le but d'observer et de quantifier l'effet des capteurs bas coût inclus dans les téléphones portables et tablettes actuels, il est nécessaire de mettre en place un protocole de comparaison des capteurs OD avec les capteurs OG utilisés dans la plupart des développements en laboratoire.

3.2.1 Critère d'évaluation

Afin de pouvoir comparer les trajectoires, il est nécessaire de définir une métrique. Deux méthodes sont généralement utilisées. La plus utilisée est l'erreur moyenne des distances au carré calculée entre les positions de chaque caméra, dans un référentiel global (voir fig. 3.5). La seconde est l'erreur odométrique, qui consiste à mesurer la dérive pour un ensemble de sous-trajectoires de longueur arbitrairement pré-définie, dans un repère défini localement (voir fig. 3.6).

Les trajectoires considérées de notre contexte supposent des ré-observations fréquentes de la scène qui permettent de minimiser la dérive au cours du temps. De plus, les trajectoires sont généralement courtes aussi bien temporellement ($< 1mn$) que spatialement ($< 5m$). L'erreur RMSE est donc adaptée en tant que critère d'évaluation.

3.2.2 Expérience sur données synthétiques

Les données synthétiques ont l'avantage de permettre la réalisation de séquences idéales impossibles à reproduire dans la réalité, avec des paramètres

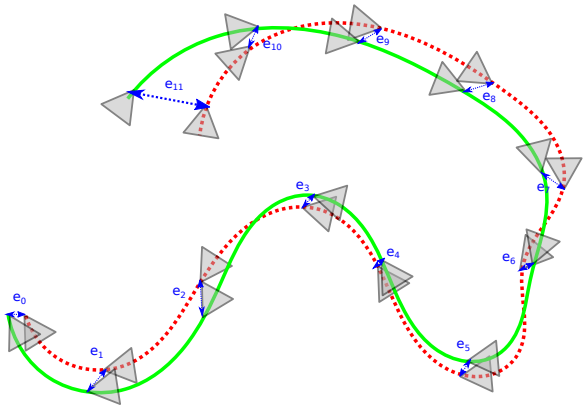


FIGURE 3.5 – L'erreur RMSE (Root Mean Squared Error) entre deux trajectoires est définie par la somme des distances aux carrés entre chaque position caméra estimée et chaque position de référence. L'erreur de la trajectoire 2D estimée (vert) par rapport à la trajectoire vérité terrain (pointillés rouges) est calculée en sommant les carrés des distances e_n (bleu) entre les positions des caméras.

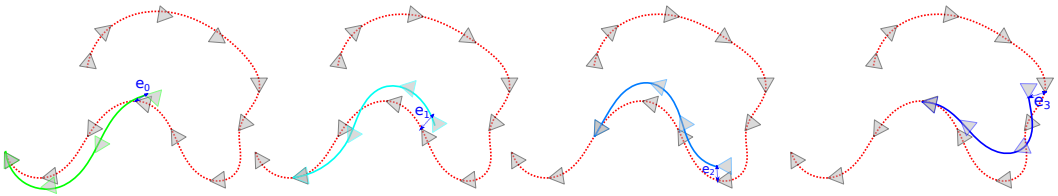


FIGURE 3.6 – L'erreur odométrique entre deux trajectoires est calculée comme la somme des déviations locales des sous-trajectoires. Ainsi, la trajectoire est "sectionnée" en sous-partie d'une longueur prédéfinie, ici 4 poses caméras, la première pose caméra est alignée sur la vérité terrain. La déviation est alors calculée comme la distance entre la dernière pose de la sous trajectoire et la pose de la vérité terrain associée.

parfaitement connus. Il est par exemple possible de recréer deux séquences ayant exactement la même trajectoire et le même point de vue, mais acquises par des caméras différentes. Les jeux de données simulées permettent donc de s'abstraire de contraintes secondaires, et de tester dans des conditions parfaites le comportement d'un algorithme de SLAM selon les modèles de caméras OG et OD.

3.2.2.1 Jeux de données cuisine synthétique

Le jeu de données utilisé est une séquence réalisée sous Blender du parcours d'une caméra dans une cuisine. Une première séquence est réalisée en utilisant un modèle de caméra OG et la seconde est réalisée en utilisant le modèle OD de Blender qui utilise des B-Splines quadratiques (continuité C^1) pour l'interpolation de la pose caméra (disponible à



FIGURE 3.7 – Exemple d'images de séquences réalisées sous Blender avec un modèle de caméra à obturateur global (en haut) et un modèle de caméra à obturateur déroulant (en bas).

partir de la version 2.77, voir `http://adaptivesamples.com/2015/12/08/rolling-shutter-coming-to-cycles/`).

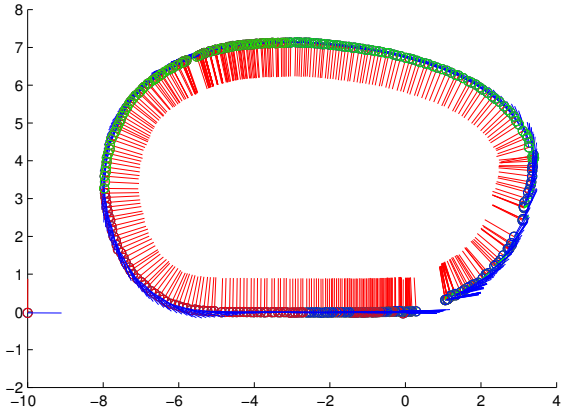


FIGURE 3.8 – Illustration des différentes positions de la caméra au cours du temps. La trajectoire est observée de dessus. Les données des poses caméras sont issues de Blender et sont utilisées comme vérité terrain.

3.2.2.2 Résultats sur la séquence synthétique

Le mouvement de départ de la trajectoire est lent et dénué de rotation afin de faciliter l'initialisation des algorithmes de SLAM par des méthodes pour caméra OG (estimation d'homographie ou de matrice fondamentale). Le SLAM open-souce ORB-SLAM [Mur-Artal 2015a] est utilisé pour l'estimation de la trajectoire, choisi après divers tests qualitatifs pour sa simplicité, son

efficacité et sa robustesse de suivi sur des images OD acquises à 30Hz face à d'autres algorithmes (LSD-SLAM, PTAM, SVO).

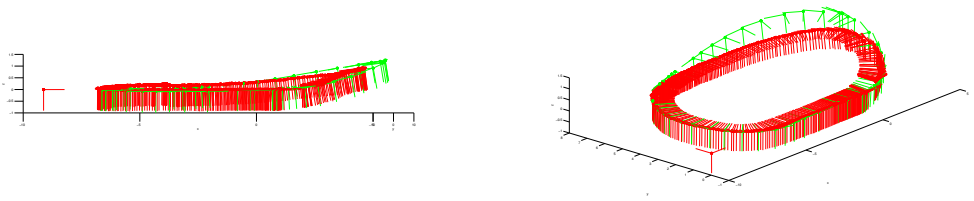


FIGURE 3.9 – La trajectoire estimée par ORBSLAM (vert) sur les images GS générées par Blender. La trajectoire vérité terrain est affichée en rouge. En vue de côté (gauche) une légère dérive de la pose s'observe sur la fin de la trajectoire. En vue du dessus (droite), une dérive de la trajectoire s'observe principalement sur l'axe x et y.

Cette séquence étant simulée, la vérité terrain de la pose caméra est donc de facto disponible pour chaque image. De plus, le SLAM s'initialise sur les séquences OG et OD à partir de la première image. L'initialisation est réalisée entre la première image qui devient la première image clé et une seconde image acquise postérieurement. Ceci permet d'éviter le recalage entre les trajectoires OD et OG étant donné que la pose de la première image-clé est considérée comme le repère monde. Nous paramétrons le SLAM pour extraire et suivre 1000 points entre chaque image-clé. La matrice des paramètres intrinsèques de la caméra est donnée par Blender, qui est nécessaire pour réaliser le rendu des images. Les images sont générées sans distorsions radiales et tangentielles.

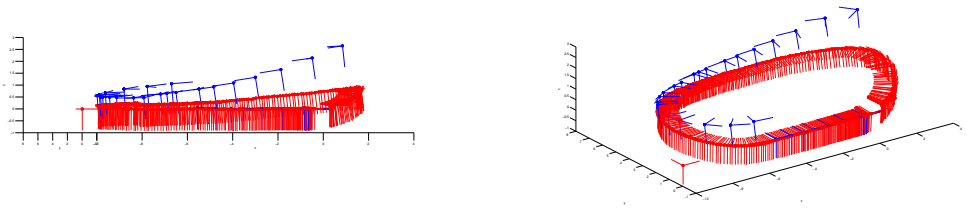


FIGURE 3.10 – La trajectoire estimée par ORBSLAM (bleu) sur les images RS aux côtés de la vérité terrain (rouge). Une dérive intervient sur l'axe z dès la première rotation.

Comme le montre les images 3.9 et 3.10, les translations affectent relativement peu le SLAM sur les images OD, étant donné qu'elles sont relativement lentes et engendrent peu de distorsions visibles dans les images. En revanche, dès lors que les distorsions deviennent non négligeables dans les images, induites ici par les deux principaux mouvements de rotation de la trajectoire,

l'estimation dérive car le modèle ne s'adapte pas aux observations. En observant les résultats obtenus sur les données synthétiques, la modélisation de la caméra OD s'avère nécessaire dès qu'une trajectoire comprend des rotations relativement rapides. La suite des travaux se concentre sur des séquences d'images réelles acquises à partir de caméras OD et OG, afin de confirmer que les observations obtenues sur les données synthétiques se reproduisent pour des trajectoires réelles.

3.2.3 Expériences sur données réelles

3.2.3.1 Protocole expérimental

Similairement à [Hedborg 2012], un dispositif a été mis en place regroupant deux caméras ueye, l'une à obturateur déroulant et l'autre à obturateur global, ainsi qu'une centrale inertielle.

Le choix d'une caméra OD ueye a été fait dans le but d'obtenir une comparaison plus rigoureuse des effets de distorsions OD, étant donné que cette caméra possède une optique similaire à la caméra ueye OG. La synchronisation est également beaucoup plus simple à réaliser entre ces deux caméras plutôt qu'entre une caméra équipant un téléphone portable et une caméra ueye.

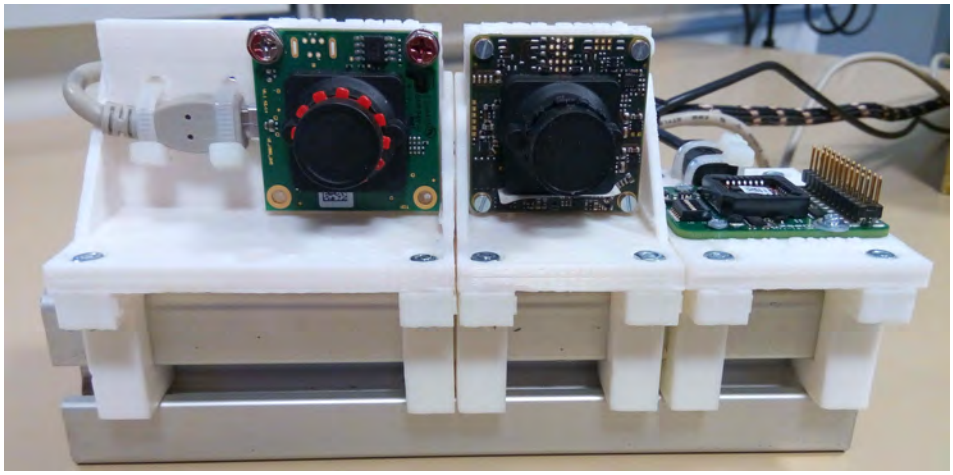


FIGURE 3.11 – Le dispositif permettant d'acquérir des séquences de scènes d'intérieurs. Les deux caméras (OD à gauche et OG à droite) ont des points de vue suffisamment proches pour que leurs images soient comparables. La centrale inertielle à gauche, qui peut être utilisée pour la synchronisation.

L'objectif est de démontrer l'impact de la non-modélisation des caméras OD sur les algorithmes de SLAM développés pour des modèles de caméras OG. Il n'est pas possible d'utiliser la trajectoire estimée à partir des images OG en tant que vérité terrain comme décrit dans [Hedborg 2012]. La vérité terrain est donc obtenue par un système de capture de mouvements. Des mar-

queurs infrarouges sont accolés au dispositif qui peut alors être suivi par les caméras de capture de mouvements (MOCAP VICON). La trajectoire ainsi obtenue est échantillonnée à 200hz, soit plus de 6 fois la fréquence des caméras (30hz). L'algorithme sur lequel sont effectués les tests est le SLAM open-source ORBSLAM, choisi pour les raisons évoquées précédemment. ORBSLAM intègre une relocalisation par sac-de-mots visuels très rapide et robuste en cas de perte du suivi, ce qui permet, sur certaines séquences à forte dynamique, de pouvoir comparer plus longuement les trajectoires même à la suite d'un échec du suivi.

3.2.3.2 Recalage de trajectoire

Les différentes trajectoires obtenues disposent d'un référentiel spatial et d'une échelle métrique propre à chacune. La trajectoire obtenue par MOCAP offre une trajectoire localisée dans un repère à une échelle réelle donnée en mètre. L'échelle pour les trajectoires estimées par le SLAM est obtenue en fonction de la distance médiane entre les points 3D initialement reconstruits et la position de la première caméra. Le repère spatial de ces trajectoires est aligné sur la première pose caméra estimée par l'algorithme. Le SLAM ne s'initialise pas nécessairement sur les mêmes images pour les deux caméras, étant donné les différences de points de vue et de conception des caméras.

Dans le but de comparer les trajectoires, il est donc nécessaire de les recaler dans l'espace. Les données sont horodatées dans un référentiel commun par des méthodes similaires à [Hedborg 2012] pour simplifier l'association des données.

Les trajectoires sont toutes considérées similaires à une rotation, une translation et à un facteur d'échelle près. Il est possible de chercher à estimer consécutivement ces paramètres, mais il est plus commun de directement estimer une similitude $\mathbf{S}_{w,c} \in \mathbb{S}im3$ (voir fig. 3.12).

Le repère de la trajectoire estimée par les caméras MOCAP sera considéré comme le repère monde. Il est donc nécessaire de définir deux matrices de transformations $\mathbf{S}_{OG,w}, \mathbf{S}_{OD,w} \in \mathbb{S}im(3)$ intégrant un terme d'échelle s. $\mathbf{S}_{OG,w}$ correspond à la transformation entre le repère monde et le repère de la trajectoire de la caméra OG, et $\mathbf{S}_{OD,w}$ représente la transformation entre le repère monde et le repère de la trajectoire estimée avec une caméra OD tel que :

$$\mathbf{S_{OG|OD,w}} = \begin{pmatrix} \mathbf{R}_{OG|OD} & \mathbf{a}_{OG|OD} \\ \mathbf{0} & s^{-1} \end{pmatrix} \tag{3.3}$$

Dans le but d'obtenir une estimée finale fidèle de ces transformations, il est primordial de les optimiser selon un critère défini à minimiser ou maximiser (erreur, distance, vraisemblance). Le critère de l'erreur sur la translation $\mathbf{a} \in \mathbb{R}^3$ est choisi pour sa simplicité de représentation géométrique et de mise en place. Dans le but de superposer les trajectoires afin de les comparer, la position (i.e. la translation) des caméras estimées doit être alignée.

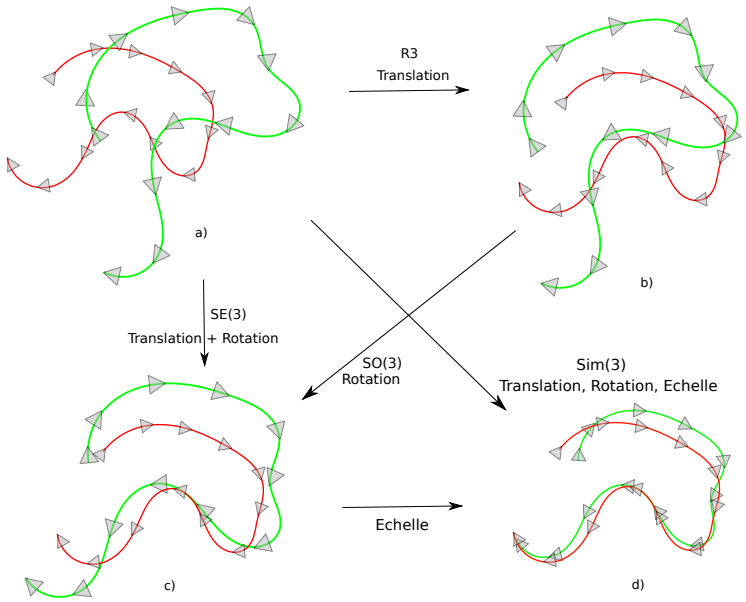


FIGURE 3.12 – Les trajectoires sont recalées spatialement, en estimant une translation, une rotation et une échelle par minimisation de l'erreur RMSE. Il est possible d'optimiser séquentiellement, en passant dans l'ordre de la figure a vers d, ou alors directement en optimisant sur $\mathbb{S}im3$ le groupe des similitudes (Translation, Rotation, Échelle)

Sur les séquences réalisées, l'utilisation d'une erreur de pose ($\mathbb{SE}(3)$) entraîne une convergence de l'optimisation vers des solutions erronées. Ces aspects sont visibles figure 3.13 qui illustre les composantes des trajectoires OD (bleu) et OG (vert) en comparaison de la vérité terrain (rouge) après un recalage minimisant la translation à gauche et un recalage minimisant la pose (sur $\mathbb{SE}(3)$) à droite.

Les données doivent donc être correctement associées afin de pouvoir calculer ce critère d'évaluation. Pour réaliser cette association, deux solutions sont présentées ci dessous.

Une première solution consiste à associer les données caméras aux données MOCAP les plus proches temporellement. Étant donné que la fréquence d'acquisition des données de la MOCAP est 6 fois supérieure à celle des caméras, l'erreur sur l'horodatage peut être considérée négligeable. Les n données de poses caméras estimées par le SLAM sont donc associées à n données MOCAP.

Ainsi, une fois l'association réalisée entre les données MOCAP et les données caméras, une erreur peut être minimisée entre les trajectoires. L'erreur sur la translation, est définie par un vecteur $\mathbf{Err}_{\mathbf{a},OG} \in \mathbb{R}^3$ pour l'OG et $\mathbf{Err}_{\mathbf{a},OD} \in \mathbb{R}^3$ pour l'OD :

$$\mathbf{Err}_{\mathbf{a},OD|OG} = \sum_{i=0}^{n} (\mathbf{a}_{MOCAP,i} - \mathbf{S}_{OG|OD}\mathbf{a}_{OD|OG,i})^2 \tag{3.4}$$

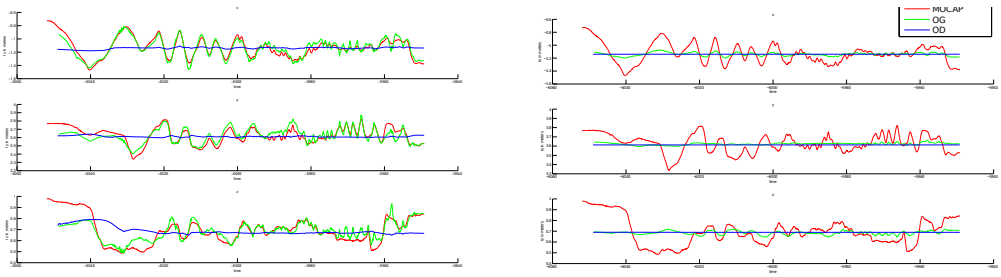


FIGURE 3.13 – Comparaison des composantes en translation des trajectoires OD (bleu) et OG (vert) recalées après optimisation, en minimisant l'erreur de translation (gauche), et l'erreur sur la pose (droite). La vérité terrain obtenue par MOCAP est affichée en rouge.

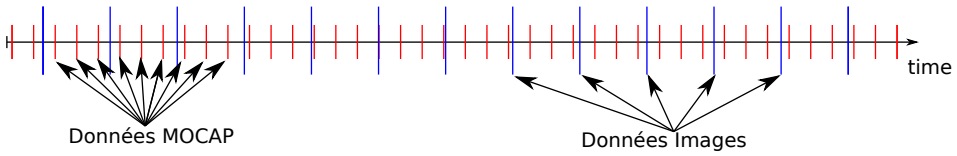


FIGURE 3.14 – Les données MOCAP (rouge) ont une fréquence différente de celle des caméras (bleu). Il est possible d'associer les poses caméras aux poses MOCAP selon leurs proximités temporelles.

L'erreur sur la pose complète de la caméra est définie par un vecteur d'erreur $\mathbf{Err}_{\mathbf{\Omega},OD|OG} \in \mathfrak{se}(3)$:

$$\mathbf{Err}_{\mathbf{\Omega},OD|OG} = \sum_{i=0}^{n} log(\mathbf{T}_{MOCAP,i}^{-1} \mathbf{S}_{OG|OD} \mathbf{T}_{OD|OG,i}) \tag{3.5}$$

Ainsi la similitude recherchée $\mathbf{S}^*$ est estimée par un processus de minimisation itératif de l'erreur par l'algorithme de Levenberg-Marquardt :

$$\mathbf{S}^* = \arg\min_{\hat{\mathbf{S}}} Err_{\mathbf{a}_{OD|OG}|\mathbf{\Omega}_{OD|OG}} \tag{3.6}$$

Nous proposons une autre solution qui consiste à interpoler la trajectoire MOCAP aux horodatages des poses caméras, afin d'obtenir des correspondances au temps exact des poses caméras. Les données de la trajectoire MOCAP sont disponibles à une fréquence bien plus élevée que celle de la caméra. En ajustant une trajectoire à ces données en utilisant autant de PC que de données, l'interpolation est considérée suffisamment précise et fidèle.

Pour obtenir une donnée MOCAP à la date d'exposition d'une image caméra, il est nécessaire d'interpoler la trajectoire MOCAP par les méthodes décrites dans le chapitre 2.

Ainsi, le processus itératif de recalage par optimisation consiste à minimiser une erreur entre une pose MOCAP $\mathbf{T}_{MOCAP}(t_{OG|OD})$ interpolée au temps

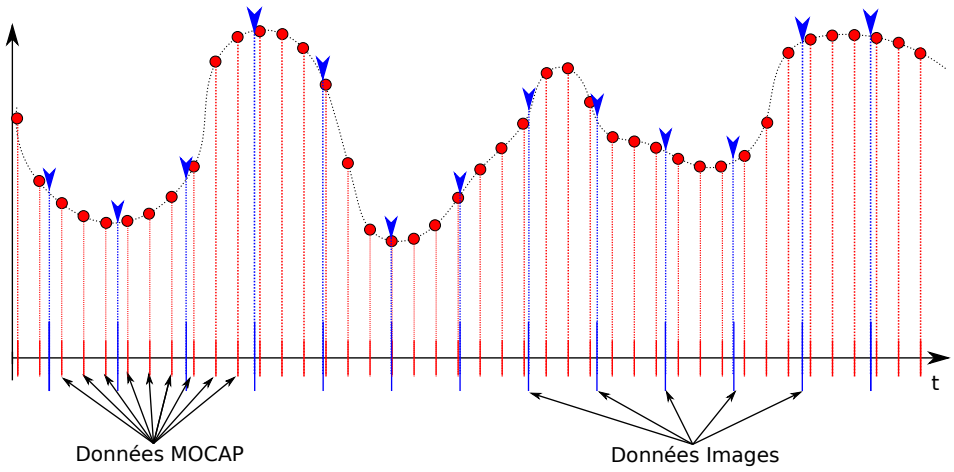


FIGURE 3.15 – Les données MOCAP (en rouge) sont interpolées aux horodatages des poses caméras. Les poses MOCAP sont utilisées comme poses de contrôle pour l'interpolation. La haute fréquence des données permet une interpolation fiable de la pose.

$t_{OG|OD}$ et une pose caméra $\mathbf{T}_{OG|OD}$ horodatée en $t_{OG|OD}$. L'erreur se définie donc par :

$$\mathbf{Err}_{\Omega,OG|OD} = log(\mathbf{T}^{-1}_{MOCAP}(t_{OG|OD})\mathbf{T}_{OG|OD}) \tag{3.7}$$

La définition en temps continu de la trajectoire permet d'éviter l'association de données en utilisant directement la pose interpolée à l'horodatage de la pose caméra. Pour le cas de nos séquences, les deux méthodes donnent des résultats très similaires. L'apport en précision de l'interpolation n'est pas significatif dans le cas des mouvements de caméras tenues à la main. Les mouvements étant globalement trop lents pour que l'erreur de position à quelques centièmes de secondes près soit significatif. Il est en revanche certain que cette méthode est applicable et utile dans des cas de mouvements plus rapides.

3.2.4 Résultats du SLAM sur séquences réelles OG et OD

Dans le contexte de la réalité augmentée pour l'aménagement d'intérieur, les mouvements de la caméra tenue par un utilisateur ont des caractéristiques sensiblement différentes de ceux d'une caméra apposée à un robot d'exploration par exemple.

Dans le cas de l'aménagement d'intérieur, l'utilisateur va essayer de visualiser un meuble virtuel dans une pièce sous divers angles. Ainsi, les mouvements de la caméra sont généralement bornés dans une zone restreinte. En pratique, la caméra observe quasiment continuellement la même portion de l'environnement où se trouve le meuble incrusté, ce qui empêche généralement une dérive de l'échelle au cours de la trajectoire.

Plusieurs séquences sont réalisées ayant des mouvements de différents types. Toutes ces séquences commencent par des mouvements en translation relativement lents afin de permettre au SLAM d'initialiser une première carte à partir d'appariements entre 2 images-clés. Les distorsions induites par les caméras OD sont alors négligeables et l'initialisation par des méthodes développées pour les caméras OG deviennent applicables. Ceci permet au SLAM appliqué sur les images OD de partir sur une base de points reconstruits fiable et d'observer la dérive au cours du temps induite par l'erreur de modélisation.

3.2.4.1 Séquences translation

Les séquences en translation réalisées comprennent des mouvements amples et rapides en translation et des mouvements lents et souples en rotation. Sur ces séquences, les mouvements en translation occasionnent peu de déformation induites par la caméra OD sur les images.

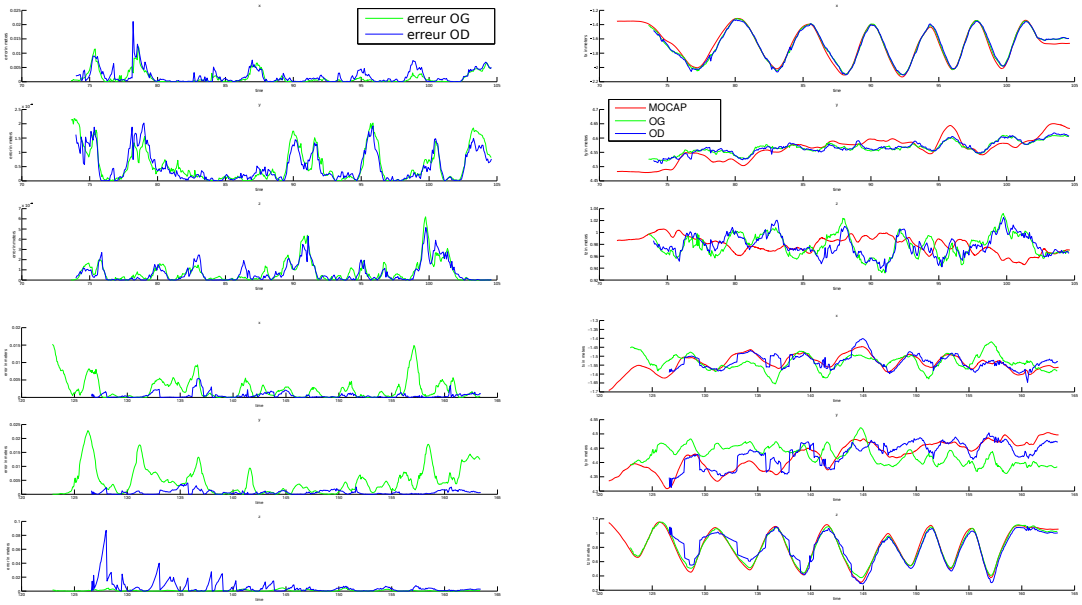


FIGURE 3.16 – Comparaison des composantes (droite) des trajectoires OD (bleu), OG (vert) et MOCAP (rouge) recalées. Les figures de gauche illustrent l'erreur en fonction du temps associée à la trajectoire OD (bleu) et OG (vert).

Comme l'illustre la 3.16, l'estimée de la trajectoire fournie par un SLAM utilisant un modèle de caméra OG sur les images OD est comparable à celle sur les images OG. Les erreurs sur l'estimée de la position des caméras (à gauche figure 3.16) sont sensiblement équivalentes. Ce résultat s'explique par le fait que les translations ont un impact plus faible que les rotations sur le mouvement pixellique entre les images. Ainsi, pour une caméra tenue à la main, la vitesse du mouvement en translation reste généralement trop faible

pour que les distorsions induites par les caméras OD soient influentes. En revanche, la caméra étant un capteur projectif, une courte translation de la caméra face à un objet très proche de celle-ci entraînera tout de même de fortes distorsions, mais ce cas n'est pas représenté dans ces séquences.

L'impact fort de la rotation est la raison pour laquelle les auteurs de PTAM [Klein 2009] considèrent uniquement la vitesse angulaire pour corriger la position des amers, négligeant la translation.

3.2.4.2 Séquences rotation

Les séquences en rotation sont réalisées avec des caméras suivant des mouvements brusques et rapides en rotation. Ces mouvements provoquent des distorsions même à faible vitesse angulaire, du fait de l'aspect projectif des caméras. De plus, dans le contexte d'une caméra tenue à la main, les trajectoires contiennent généralement de rapides rotations au cours de l'acquisition d'une image, entraînant des déformations visibles dans les images.

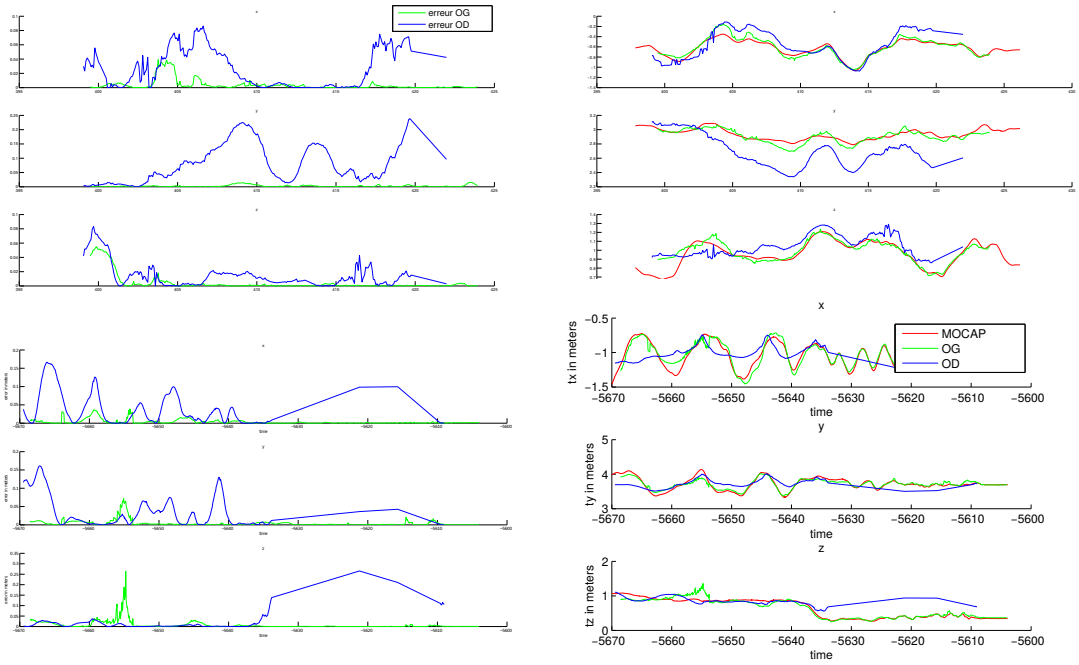


FIGURE 3.17 – Comparaison des composantes (droite) des trajectoires OD (bleu), OG (vert) et MOCAP (rouge) recalées. Les figures de gauche illustrent l'erreur en fonction du temps associée à la trajectoire OD (bleu) et OG (vert).

La figure 3.17 illustre bien la dérive rapide de l'estimée de la trajectoire du SLAM sur les images OD pour des mouvements rapides en rotation. Les composantes en translation et l'erreur associée de deux trajectoires sont présentées. L'erreur atteint plus de 25cm sur une trajectoire d'environ 2m (haut), avant de diminuer à nouveau du fait d'une ré-observation de la scène, et de

		Erreur Moyenne OG			Erreur Moyenne OD		
Type Seq	Num Seq	x	y	z	x	y	z
1	4	0.204	0.280	0.439	1.956	0.963	1.437
	5	0.181	0.097	0.120	0.213	0.124	0.110
	6	0.146	0.254	0.052	0.137	0.262	0.045
	7	0.343	0.403	0.759	3.620	3.361	1.249
	8	0.074	0.151	0.059	0.074	0.100	0.048
	9	0.290	0.616	0.354	0.275	0.485	0.233
	10	0.189	0.243	0.192	0.175	0.238	0.375
2	12	0.358	0.334	0.162	0.296	0.309	0.141
	13	0.209	0.146	0.377	0.240	0.202	0.416
	14	0.534	0.168	0.569	0.429	0.169	0.241
	15	0.131	0.054	0.076	0.191	0.049	0.069
	16	0.230	0.467	0.073	0.054	0.069	0.560
	17	0.039	0.036	0.036	0.035	0.121	0.031
	19	0.248	0.418	0.181	0.366	0.467	0.180
	20	0.372	0.258	0.224	0.384	0.259	0.256

TABLE 3.1 – Tableau de l'erreur moyenne sur les composantes en translation pour un SLAM exécuté sur des images OG (gauche) et OD (droite). Les plus faibles erreurs pour chaque séquence sont surlignées en bleues, et les erreurs significatives sont surlignées en rouge.

continuer dans ce cycle de dérives et de ré-observations.

Les 6 graphiques en bas de la figure 3.17 montre une trajectoire comprenant de rapides mouvements de rotation en fin de trajectoire. Le suivi de points est perdu sur la quasi-totalité de la seconde moitié de la trajectoire, à partir du moment où les rotations rapides commencent. Les rotations rapides provoquent sur la plupart des séquences réalisées soit une dérive, soit des pertes de suivi.

Cependant, bien que les pertes du suivi soient fréquentes, le SLAM se relocalise dès que le mouvement est moins brusque. Le SLAM sur les images OG parvient à suivre le mouvement mais entraîne une dérive à cause de la dégradation de l'information de l'image pour les mouvements rapides (flou de bouger, erreur d'appariements, etc). Cette dérive corrompt la carte, et cause donc une dérive globale du SLAM. La perte du suivi sur les images OD suffisamment tôt permet d'éviter cette dérive. Une analyse des résultats et du nombre de pertes du suivi est apportée par la suite.

3.2.4.3 Analyse des résultats

Le tableau 3.1 illustre l'erreur moyenne sur les composantes en translation. Les séquences 4 à 10 incluent des rotations rapides de la caméra, alors que les séquences 12 à 20 comportent des translations rapides de la caméra. Les

numéros n'apparaissant pas sont les séquences sur lesquelles le SLAM échoue.

Au premier abord, il semblerait que les différences ne soient pas significatives. Les très larges erreurs, surlignées en rouge et obtenues sur des séquences intégrant des mouvements à forte rotation, apparaissent uniquement pour les trajectoires OD.

Pour les séquences en translation, les différences sont généralement très faibles et peu significatives. Comme énoncé précédemment, les images issues d'une caméra OD pour des mouvements lents sont sensiblement similaires de celles obtenues avec une caméra OG. Les erreurs peuvent généralement s'apparenter à la légère différence de points de vue entre les caméras.

Lors des mouvements rapides, le suivi de points d'intérêts du SLAM échoue momentanément sur les images OD avant que la carte n'ait le temps d'être corrompue. Le SLAM relocalise ensuite la caméra lorsque le mouvement redevient moins brusque. Sur les images OG, le SLAM parvient à suivre des points, mais le mouvement rapide entraîne une dérive locale et un pic d'erreur de l'estimation de pose auquel le SLAM OD échappe. Ce mécanisme de perte de suivi fonctionne comme un garde fou qui empêche la corruption de la carte. C'est pourquoi l'erreur moyenne d'estimation de trajectoire pour les images OD est parfois plus faible que pour les images OG.

La perte de suivi momentanée peut être préférable à une corruption de la carte suivant les applications recherchées. La plupart de nos séquences réobservent très régulièrement ou en continu une partie de la scène. Ce cas particulier avantage de ce fait la perte de localisation momentanée. Ce n'est cependant pas le cas de nombreuses applications où la perte du suivi peut être préjudiciable.

Ces pertes du suivi sont observables sur les graphiques des trajectoires, lorsque la courbe forme localement une droite à cause de l'absence de données, mais sont difficiles à discerner. Le tableau 3.2 montre le nombre de pertes de suivi plus ou moins longues (plus de 5 frames à droite, plus de 10 frames à gauche) pour chaque séquence. Le SLAM est régulièrement perdu sur les images OD. En mettant en corrélation ces résultats avec le tableau précédent, il apparaît clairement que la plupart des séquences où l'erreur est plus élevée pour l'OG sont celles où le suivi est régulièrement perdu pour les images OD. C'est le cas des séquences 6, 8, 9, 12 et 14.

La séquence 16 est un cas particulier. Le mouvement y est très dynamique au départ de la séquence. Le SLAM sur les images OG s'initialise plus tôt que sur les images OD, au cours d'un mouvement rapide, qui produit une estimée initiale corrompue. Sur les images OD, le SLAM ne parvient pas à s'initialiser aussi tôt, et s'initialise donc lors d'un mouvement plus calme, ce qui permet d'éviter de commencer avec une carte corrompue.

Ainsi la majeure partie des trajectoires estimées fournie des résultats sensiblement similaires à l'exception des mouvements à forte dynamique. Ces pics

	nb Track. Fail>10 frames		nb Track. Fail>5 frames	
Num Seq	GS	RS	GS	RS
4	0	6	0	13
5	0	2	0	11
6	0	2	3	6
7	0	3	0	3
8	1	3	3	8
9	0	9	2	19
10	0	0	0	10
12	0	1	0	7
13	0	0	0	0
14	1	6	1	9
15	0	0	0	0
16	0	0	0	0
17	0	0	0	0
19	0	0	0	7
20	0	0	0	9

TABLE 3.2 – Tableau représentant le nombre de pertes de suivi pour chaque séquence. Le tableau de gauche illustre le nombre de pertes de suivi durant plus de 10 frames, et à droite pendant plus de 5 frames.

d'erreurs sont cependant lissés dans la moyenne. La seule différence est lorsque la carte est corrompue dès l'initialisation par un mouvement trop rapide effectué au cours de celle-ci. Ces résultats sont attendus et confirment l'impact des images acquises par des caméras OD sur les algorithmes de SLAM.

La section suivante décrit les modèles de caméras OD existants dans la littérature, qui permettent de tirer parti des avantages offerts par leurs conceptions plutôt que de tenter de corriger les images ou d'ignorer les distorsions.

3.3 Modèles existants de caméras OD

3.3.1 Modèles de mouvement pour caméra OD

Plusieurs modèles géométriques de caméras OD sont utilisés dans la littérature, préconisant différentes approximations pour décrire le mouvement de la caméra au cours de l'acquisition d'une image. [Meingast 2005] propose différents modèles de caméras admettant des mouvements à vitesse constante durant l'exposition de l'image et avance également une généralisation aux mouvements plus complexes. [Klein 2009] furent parmi les premiers à tenir compte de l'OD pour adapter PTAM sur smartphone ; dans leur approche, la vitesse angulaire de la caméra (supposée constante dans une image) est estimée afin de rectifier les mesures, les rendant utilisables avec un simple modèle de caméra OG. [Magerand 2010, Magerand 2012, Magerand 2014] utilise dif-

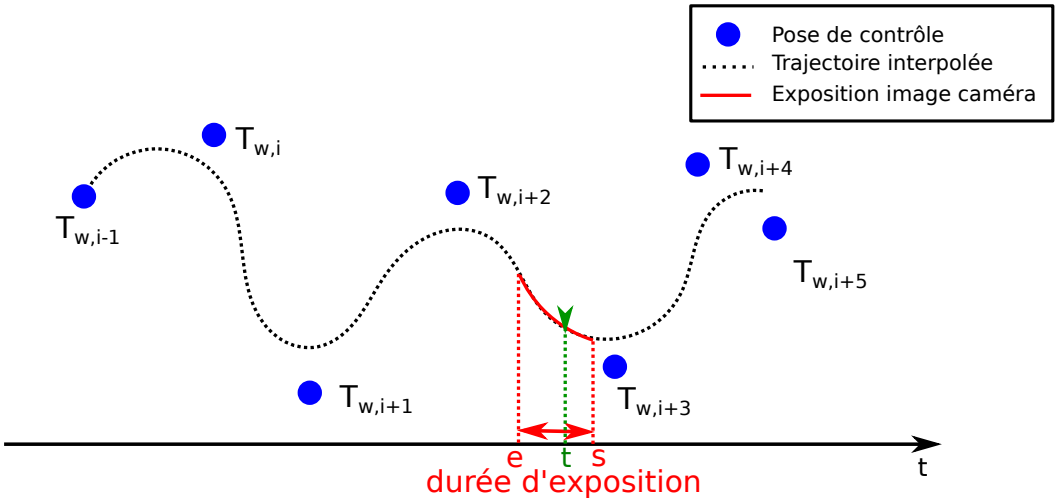


FIGURE 3.18 – Une composante de trajectoire interpolée (pointillés noirs) à partir de PC (bleu) en fonction du temps. La trajectoire interpolée au cours de l'acquisition d'une image OD est affichée en rouge. e et s correspondent aux dates de départ et de fin d'exposition de l'image et t (vert) correspond à la date d'interpolation, et donc à la date d'exposition d'une ligne de l'image pour laquelle une estimée de la pose est recherchée.

férents modèles de caméras OD en temps continu, mais uniquement pour le cas de l'estimation de déplacement d'objets connus observés par une caméra fixe. Les modèles en temps continu suppose que la trajectoire de la caméra est définie de manière continue en fonction du temps, à l'inverse des modèles discret où la trajectoire de la caméra est résumée par des poses discrètes dans l'espace (pour chaque image ou chaque image-clé). [Forssén 2010] [Hedborg 2011] [Hedborg 2012] proposent un ajustement de faisceaux OD utilisant des modèles d'interpolation linéaire indépendant pour la rotation (SLERP) et la translation. [Albl 2015] présente une estimation de poses caméras OD par solution non itérative, utilisant une approximation linéaire de l'orientation de la caméra (double linéarisation). Dans des travaux postérieurs[Albl 2016], le précédent modèle est modifié en tenant compte d'une connaissance a priori de la verticale, nécessitant uniquement cinq correspondances afin d'estimer la pose caméra.

[Dai 2016] présente une généralisation de la géométrie épipolaire aux modèles de caméras OD. Une description formelle de plusieurs modèles de caméras est également fournie. Un des modèles de caméras OD le plus simple est le modèle linéaire, qui considère que la caméra est en pure translation avec une vitesse linéaire constante $\mathbf{v} \in \mathbb{R}3$ durant l'acquisition d'une image. Soit $\mathbf{R}_0$ et $\mathbf{a}_0$ respectivement l'orientation et la position de la caméra au début de l'exposition de l'image, la pose caméra $\mathbf{T}_{wc_v}$ à la ligne v est :

$$\mathbf{T_{wc_v}} = \begin{pmatrix} \mathbf{R}_0 & \mathbf{a}_0 + v\mathbf{v} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \tag{3.8}$$

$\mathbf{v}$ est exprimée comme une vitesse en coordonnées image. Un modèle de caméra OD, légèrement plus complexe, inclut au modèle linéaire une rotation à vitesse angulaire constante, avec $\omega \in \mathbb{R}3$ la vitesse angulaire constante :

$$\mathbf{R}_v = (\mathbf{I}_3 + sin(v\omega)[n]_\times + (1 - cos(v\omega))[n]_\times)\mathbf{R}_0, \tag{3.9}$$

$$\mathbf{a}_v = \mathbf{a}_0 + v\mathbf{v} \tag{3.10}$$

Ces approximations, bien que généralement suffisantes, ne permettent pas de modéliser les vitesses et les accélérations non constantes. [Steven Lovegrove 2013] propose une interpolation sur $\mathbb{SE}3$ garantissant une continuité C^2 de la trajectoire caméra. Ce modèle d'interpolation permet l'intégration de données issues de multiples capteurs, et offre une modélisation fine de la caméra pour des variations de vitesses et d'accélérations au sein de l'image. Ce modèle utilise les B-Splines cubiques cumulatives détaillées dans le chapitre 2 pour obtenir une trajectoire de la caméra continu en fonction du temps t. Il est alors possible d'interpoler la pose caméra $\mathbf{T_{w,c}}$ pour chaque ligne de l'image et pouvoir ainsi projeter un point de la scène sur une ligne de l'image. La figure 3.18 illustre le principe de l'interpolation de la trajectoire (pointillés noirs) en fonction des PC (bleu). La trajectoire caméra durant l'exposition de l'image (rouge) peut ainsi être estimée à partir des PC $\mathbf{T}_{w,i+1\cdots i+4}$. La formulation d'interpolation de l'eq. 2.68 peut être adapté pour les données de la figure 3.18, la pose caméra $\mathbf{T}_{w,c}(t)$ interpolée au temps t est alors :

$$\mathbf{T}_{w,c}(t) = \mathbf{T}_{w,i+1} \prod_{j=1}^{3} \exp(\tilde{B}(t)_j \mathbf{\Omega}_{i+1+j}) \tag{3.11}$$

C'est ce dernier modèle d'interpolation qui est utilisé dans cette thèse afin d'avoir une représentation en temps continu de la trajectoire caméra. Ce modèle sera développé et enrichi dans le chapitre suivant.

Une estimée de la pose caméra est disponible pour chaque ligne de l'image, il est désormais possible de projeter un point 3D de la scène dans l'image.

3.3.2 Modèle de projection pour caméra OD

Le modèle de projection d'une caméra OD est défini de la manière suivante. Soit $\mathbf{P_w}$ un point 3D défini en coordonnées homogènes dans un repère monde $\mathbf{w}$. Soit $\mathbf{T_{c,w}} = \mathbf{T_{w,c}}^{-1}$ la matrice de transformation depuis le repère monde $\mathbf{w}$ vers le repère camera $\mathbf{c}$. Soit $\mathbf{K}$ la matrice des paramètres intrinsèques de la camera et $\pi(.)$ la projection perspective (établissant une correspondance de l'espace projectif $\mathbb{P}^2$ vers $\mathbb{R}^2$). Le modèle sténopé projette $\mathbf{P_w}$ dans le plan

image en $\begin{bmatrix} p_u & p_v \end{bmatrix}^T$ par l'opération suivante :

$$\begin{bmatrix} p_u \\ p_v \end{bmatrix} := \pi([\mathbf{K}|\mathbf{0}]\mathbf{T}_{\mathbf{c},\mathbf{w}}\mathbf{P}_{\mathbf{w}}) \tag{3.12}$$

Afin de modéliser la variabilité de la pose au cours du temps, $\mathbf{T}_{\mathbf{c},\mathbf{w}}$ devient dépendante au temps t, $\mathbf{T}_{\mathbf{c},\mathbf{w}}(t)$ (voir fig. 3.18). La spline étant définie par eq.(3.11) dans le repère de la spline $\mathbf{s}$, qui peut différer du repère camera $\mathbf{c}$, une transformation $\mathbf{T}_{\mathbf{c},\mathbf{s}}$ constante dans le temps est éventuellement requise afin d'obtenir $\mathbf{T}_{\mathbf{c},\mathbf{w}}(t)$:

$$\mathbf{T}_{\mathbf{c},\mathbf{w}}(t) = \mathbf{T}_{\mathbf{c},\mathbf{s}}\mathbf{T}_{\mathbf{s},\mathbf{w}}(t) \tag{3.13}$$

La projection à la date t est obtenue par :

$$\begin{bmatrix} p_u(t) \\ p_v(t) \end{bmatrix} := \pi([\mathbf{K}|\mathbf{0}]\mathbf{T}_{\mathbf{c},\mathbf{w}}(t)\mathbf{P}) = \omega(\mathbf{P}, \mathbf{T}_{\mathbf{c},\mathbf{w}}(t)) \tag{3.14}$$

Cependant ce modèle ne représente pas qu'à un temps donné correspond une unique exposition de ligne. Ainsi la projection $\begin{bmatrix} p_u(t_1) & p_v(t_1) \end{bmatrix}^T$ de $\mathbf{P}$ au temps $t = t_1$ sera réellement obtenue uniquement si la ligne $p_v(t_1)$ est exposée au temps t_1.

[Steven Lovegrove 2013] exprime l'exposition de la ligne au temps t comme une fonction de e, s, t, h. Ici, s correspond au temps d'exposition de la première ligne de l'image, e au temps à la fin de l'acquisition de l'image, et h à la hauteur de l'image en pixels tels que :

$$p_v(t) = h\frac{(t-s)}{(e-s)} \tag{3.15}$$

La(es) projection(s) $\begin{bmatrix} p_u(t) & p_v(t) \end{bmatrix}^T$ observée(s) par la caméra RS est(sont) obtenue(s) par intersection des courbes correspondant aux équations eq.(3.14) et eq.(3.15). Déterminer t est un problème d'optimisation qui se résout itérativement en utilisant un développement limité du premier ordre au voisinage du temps t :

$$p_v(t+\delta t) = h\frac{(t+\delta t-s)}{(e-s)} \tag{3.16}$$

$$\begin{bmatrix} p_u(t+\delta t) \\ p_v(t+\delta t) \end{bmatrix} = \omega(\mathbf{P}, \mathbf{T}_{c,w}(t)) + \delta t\frac{d\omega(\mathbf{P}, \mathbf{T}_{c,w}(t))}{dt} \tag{3.17}$$

En ne conservant que la partie concernant l'ordonné p_v dans cette équation, on obtient :

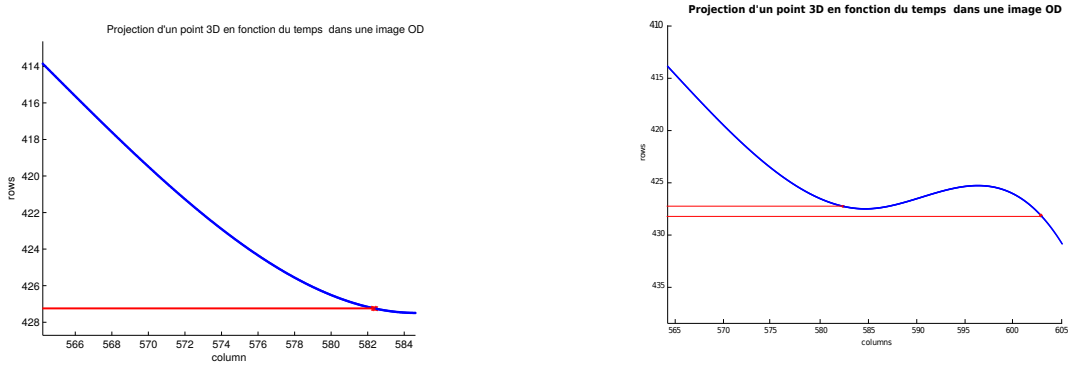


FIGURE 3.19 – Évolution de la projection d'un point 3D dans l'image au cours du temps de l'exposition de cette image, dans le cas d'un mouvement lent (gauche) et dans le cas d'un mouvement rapide avec changement de direction (droite). Dans le cas d'un mouvement rapide, un même point de la scène peut ainsi être projeté plusieurs fois dans l'image.

$$p_v(t+\delta t) = p_v(t) + \delta t \frac{d\omega(\mathbf{P}, \mathbf{T}_{c,w}(t))}{dt} \tag{3.18}$$

En réarrangeant ces équations et en résolvant pour δt :

$$\delta_t = -\frac{ht + s(p_v(t) - h) - ep_v(t)}{h + (s-e)\frac{d\omega_v(\mathbf{P}, \mathbf{T}_{c,w}(t))}{dt}} \tag{3.19}$$

En affectant à t un incrément tel que $t_{i+1} = t_i + \delta_t$ et en itérant, t_i converge généralement en approximativement 3 ou 4 itérations (voir fig. 3.20). Une fois t déterminé, la projection correspondante est obtenue en utilisant eq. 3.14. Pour des mouvements lents, la valeur de t peut être initialisée au temps correspondant à l'exposition de la ligne du milieu de l'image, pour lequel l'algorithme itératif aura de grande chance de converger. Cependant, pour des mouvements à forte dynamique, la valeur initiale doit être choisie avec précaution, étant donné que plusieurs projections sont possibles. Il est nécessaire de partir d'une valeur initiale proche de celle souhaitée, par exemple en prenant la ligne correspondant au temps trouvé dans l'image précédente. Une fois le temps t obtenu, $\mathbf{T}_{c,w}(t)$ s'obtient alors par interpolation en fonction de PC précédemment définies.

3.4 Conclusion

Le principe de fonctionnement des caméras OD a été détaillé dans ce chapitre, et une méthode d'étalonnage de ces caméras en utilisant les altérations photométriques dans les images a été exposée.

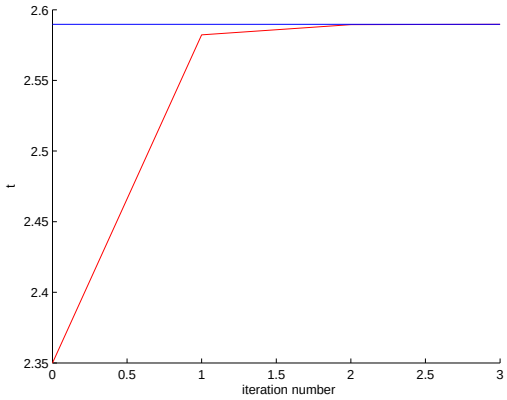


FIGURE 3.20 – Illustration de la convergence rapide de l'optimisation de la date t en fonction du nombre d'itérations. La ligne bleue indique la valeur trouvée en utilisant une méthode naïve de parcours exhaustif des lignes de l'image afin de trouver l'intersection entre la projection du point et l'exposition de la ligne. Sur le plupart des tests effectués, 3 itérations suffisent pour converger vers une valeur satisfaisante.

Un protocole expérimental utilisant des données réelles et synthétiques permet d'exposer l'influence négative des caméras OD sur les algorithmes de SLAM, entraînant une dérive de la trajectoire et une corruption de la carte lorsque ces caméras ne sont pas correctement modélisées. Ces aspects préjudiciables sont, en revanche, uniquement observables pour les cas où la caméra subit des rotations rapides. Ces mouvements sont généralement présents dans les trajectoires d'une caméra tenue à la main par un opérateur. Dans le cas de trajectoires où la rotation reste faible et pour des translations assez fortes, l'utilisation ou non d'une caméra OD n'affecte pas le processus du SLAM. Une solution possible est d'inciter l'utilisateur à ne pas exécuter de mouvements brusques, ou bien de désactiver le suivi afin de ne pas corrompre la carte et la trajectoire en cas de vitesse angulaire trop forte. Ce principe est applicable uniquement dans les cas où une perte du suivi durant plusieurs dizaines d'images voir plusieurs secondes n'est pas critique.

Finalement, un modèle de caméra OD de l'état de l'art est présenté, qui suppose une pose caméra différente pour chaque ligne de l'image. Les différentes poses caméras associées aux lignes de l'image sont interpolées par B-Spline cubique.

Le chapitre suivant s'attelle à améliorer ce modèle de caméra OD par interpolation, notamment en relaxant la contrainte de distribution temporelle uniforme des poses de contrôle.

Chapitre 4

Modèle d'interpolation à distribution temporelle non uniforme pour caméras OD

Sommaire

4.1 B-Splines non uniformes 77
4.1.1 Formulation matricielle des B-Splines cubiques non-uniformes 77
4.2 DTNU et génération des poses de contrôle 82
4.2.1 Méthode inertielle 84
4.2.2 Méthode par analyse de l'erreur 85
4.3 Adaption d'algorithmes d'optimisation pour modèle DTNU 87
4.3.1 PnP 87
4.3.2 Ajustement de faisceaux 87
4.3.3 Optimisation de trajectoires 88
4.3.4 Optimisation spatio-temporelle des poses de contrôle 88
4.3.5 Représentation par graphe 89
4.4 Résultats 90
4.4.1 Tests sur données synthétiques 90
4.4.2 Tests sur données réelles 94
4.5 Discussion 97
4.5.1 Utilisation de primitives segments 97
4.5.2 Coût calculatoire 98
4.6 Conclusion 98

Dans ce chapitre est fourni un modèle de caméra OD adaptable aux algorithmes de SLAM. Comme énoncé précédemment, différentes méthodes d'interpolation existent([Hedborg 2012],[Steven Lovegrove 2013]) pour la modélisation des caméras OD, mais elles se basent généralement sur une distribution uniforme des PC dans le temps. En effet, dans [Steven Lovegrove 2013], l'algorithme développé par les auteurs initialise une nouvelle PC chaque dixième de seconde. Cette distribution fonctionne bien même pour des mouvements relativement rapides du fait la fréquence élevée des PC dans le temps(environ 3 images entre chaque initialisation de PC).

Cette distribution temporelle uniforme (DTU) des PC possède des avantages, notamment la possibilité d'utiliser la formulation commune des B-Spline

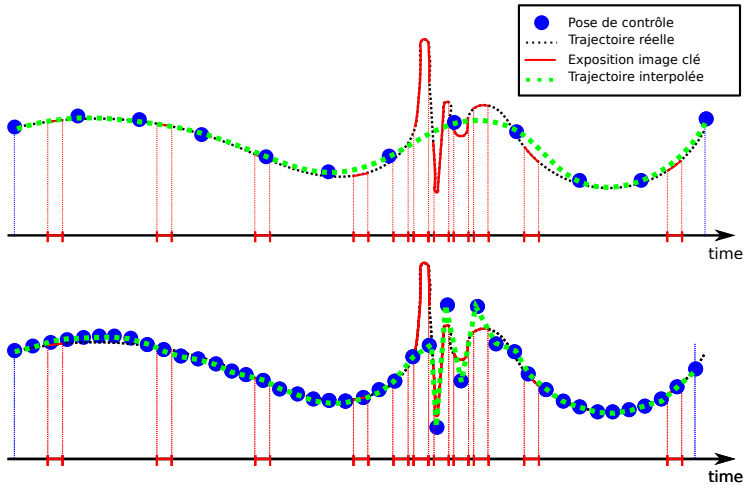


FIGURE 4.1 – Exemple de l'interpolation d'une composante d'une même trajectoire caméra (pointillés noirs) en fonction du temps pour une distribution uniforme à haute fréquence (bas) et basse fréquence (haut) de PC. La trajectoire intègre des mouvements à faible et haute dynamique. Pour une haute fréquence des PC, la trajectoire interpolée (pointillé vert) parvient à bien modéliser la trajectoire, bien qu'une partie soit lissée. Un sur-échantillonnage pour les parties à faible dynamique de la trajectoire est observable. Pour une fréquence d'échantillonnage plus faible des PC, l'interpolation lisse totalement la partie à haute dynamique, qui implique une mauvaise estimation de la pose caméra pour les lignes des images clé exposées durant cette période (rouge).

décrite au chapitre 2. En revanche, deux inconvénients sont présents dans l'utilisation d'une DTU :

— Dans le cas d'une absence de mouvement ou d'un mouvement lent, une DTU des PC avec une fréquence d'échantillonnage trop élevée implique une redondance de PC pour modéliser ces parties de la trajectoire, occasionnant un surcoût calculatoire inutile.

— Dans le cas d'un mouvement à forte dynamique, si la fréquence d'échantillonnage des PC n'est pas adaptée, l'information sera perdue ou perturbera la modélisation qui lissera la trajectoire. Augmenter la fréquence d'échantillonnage des PC pour modéliser cette partie de la trajectoire ramène au premier problème pour les parties à faible dynamique de la trajectoire.

Ces deux cas sont illustrés figure 4.1. Il est difficile de prévoir une fréquence d'échantillonnage des PC qui s'adapte à plusieurs situations. En pratique, cela équivaut à avoir d'un côté un lissage sur une partie de la trajectoire interpolée, et de l'autre une redondance d'information.

Nous proposons un modèle de B-Spline ayant une distribution temporelle non uniforme (DTNU) des PC pour l'interpolation de la trajectoire caméra. La relaxation de la contrainte de DTU permet d'allouer les PC au moment où

elles sont nécessaires. L'information peut alors être adaptée à la dynamique de la trajectoire. Cependant, la formulation classique des B-Splines (voir chapitre 2) ne permet pas une répartition non uniforme des poses de contrôle. Il faut donc utiliser une formulation légèrement différente des B-Splines pour des intervalles temporels non uniformes [Kim 1995].

4.1 B-Splines non uniformes

Par abus de langage, nous appellerons B-Splines non uniformes les B-Splines dont les points de contrôles sont distribués de manière non uniforme dans le temps. Pour ce cas particulier, les fonctions de bases sont définies par une relation de récurrence légèrement différente de De Boor ([de Boor 1972]), donnée dans [Kim 1995]. Pour une spline de degré k, la fonction de base $B_{i,k}(t)$ associée à un point P_i horodaté en t_i s'exprime par :

$$B_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(t) \tag{4.1}$$

Avec la condition d'arrêt définie par :

$$B_{i,0} = \begin{cases} 1 & \text{si} \quad t_i < t < t_{i+1} \\ 0 & \text{sinon} \end{cases} \tag{4.2}$$

La récurrence est appliquée en partant de la valeur de k correspondant au degré du polynôme d'interpolation souhaité. Cette formulation récursive n'est cependant pas adaptée pour le calcul numérique, et il est préférable d'utiliser une formulation matricielle.

4.1.1 Formulation matricielle des B-Splines cubiques non-uniformes

Nous regroupons et détaillons ici les travaux de [Yang 2009, Kim 1995, Steven Lovegrove 2013] en les exprimant sous des notations communes. Si on se restreint à l'utilisation de B-Spline cubiques, et en utilisant la définition récursive des fonctions de base, il est possible de calculer successivement ces fonctions de base pour le degré recherché.

Pour les B-Splines cubiques qui nous intéressent, la formule de récurrence s'applique jusqu'au degré 3, on obtient alors :

$$B_{i,1}(t) = \frac{t - t_i}{t_{i+1} - t_i} B_{i,0}(t) + \frac{t_{i+2} - t}{t_{i+2} - t_{i+1}} B_{i+1,0}(t) \tag{4.3}$$

$$B_{i,2}(t) = \frac{t - t_i}{t_{i+2} - t_i} B_{i,1}(t) + \frac{t_{i+3} - t}{t_{i+3} - t_{i+1}} B_{i+1,1}(t) \tag{4.4}$$

$$B_{i,2}(t) = \frac{t-t_i}{t_{i+2}-t_i}\left(\frac{t-t_i}{t_{i+1}-t_i}B_{i,0}(t) + \frac{t_{i+2}-t}{t_{i+2}-t_{i+1}}B_{i+1,0}(t)\right) + \frac{t_{i+3}-t}{t_{i+3}-t_{i+1}}\left(\frac{t-t_{i+1}}{t_{i+2}-t_{i+1}}B_{i+1,0}(t) + \frac{t_{i+3}-t}{t_{i+3}-t_{i+2}}B_{i+2,0}(t)\right) \tag{4.5}$$

$$B_{i,2}(t) = \overbrace{\frac{(t-t_i)^2}{(t_{i+2}-t_i)(t_{i+1}-t_i)}}^{X_i} B_{i,0}(t) + \overbrace{\left(\frac{(t-t_i)(t_{i+2}-t)}{(t_{i+2}-t_i)(t_{i+2}-t_{i+1})} + \frac{(t_{i+3}-t)(t-t_{i+1})}{(t_{i+3}-t_{i+1})(t_{i+2}-t_{i+1})}\right)}^{Y_i} B_{i+1,0}(t) + \overbrace{\frac{(t_{i+3}-t)^2}{(t_{i+3}-t_{i+1})(t_{i+3}-t_{i+2})}}^{Z_i} B_{i+2,0}(t) \tag{4.6}$$

Les fonctions de base pour les bases de degrés 3 ($k = 4$) sont définies par :

$$B_{i,3}(t) = \frac{t-t_i}{t_{i+3}-t_i}B_{i,2}(t) + \frac{t_{i+4}-t}{t_{i+4}-t_{i+1}}B_{i+1,2}(t) \tag{4.7}$$

Utilisant les définitions de l'équation 4.6 :

$$B_{i,3}(t) = \frac{t-t_i}{t_{i+3}-t_i}\left(X_i B_{i,0}(t) + Y_i B_{i+1}(t) + Z_i B_{i+2,0}(t)\right) + \frac{t_{i+4}-t}{t_{i+4}-t_{i+1}}\left(X_{i+1}B_{i+1,0}(t) + Y_{i+1}B_{i+2,0}(t) + Z_{i+1}B_{i+3,0}(t)\right) \tag{4.8}$$

Sachant que $B_{i,0}$ est non nulle lorsque $t_i < t < t_{i+1}$, les fonctions de base sont non nulles sur uniquement 4 intervalles de temps, de t_i à t_{i+4}. On obtient donc une formulation relative au temps t :

$$B_{i,3}(t) = \begin{cases} \frac{(t-t_i)}{(t_{i+3}-t_i)}X_i & \text{si} \quad t_i < t < t_{i+1} \\ \frac{t-t_i}{t_{i+3}-t_i}Y_i + \frac{t_{i+4}-t}{t_{i+4}-t_{i+1}}X_{i+1} & \text{si} \quad t_{i+1} < t < t_{i+2} \\ \frac{t-t_i}{t_{i+3}-t_i}Z_i + \frac{t_{i+4}-t}{t_{i+4}-t_{i+1}}Y_{i+1} & \text{si} \quad t_{i+2} < t < t_{i+3} \\ \frac{t_{i+4}-t}{t_{i+4}-t_{i+1}}Z_{i+1} & \text{si} \quad t_{i+3} < t < t_{i+4} \end{cases} \tag{4.9}$$

En utilisant les notations issues de [Yang 2009] $u_x = t_{i+x}$ et $u_{xy} = t_{i+x} - t_{i+y}$ il est possible de simplifier l'équation 4.9 :

$$B_{i,3}(t) = \begin{cases} \frac{(t-u_0)^3}{u_{30}u_{20}u_{10}} & \text{si} \quad t_i < t < t_{i+1} \\ \frac{(t-u_0)^2(u_2-t)}{u_{30}u_{20}u_{21}} + \frac{(t-u_0)(u_3-t)(t-u_1)}{u_{30}u_{31}u_{21}} + \frac{(u_4-t)(t-u_1)^2}{u_{41}u_{30}u_{20}} & \text{si} \quad u_1 < t < t_{i+2} \\ \frac{(t-u_0)(u_3-t)^2}{u_{30}u_{31}u_{32}} + \frac{(u_4-t)(t-u_1)(u_3-t)}{u_{41}u_{31}u_{32}} + \frac{(u_4-t)^2(t-u_2)}{u_{41}u_{42}u_{32}} & \text{si} \quad t_{i+2} < t < u_3 \\ \frac{(u_4-t)^3}{u_{41}u_{42}u_{43}} & \text{si} \quad t_{i+3} < t < t_{i+4} \end{cases} \tag{4.10}$$

L'intervalle de définition du segment d'interpolation considéré est $[t_{i+3}, t_{i+4}]$. [Yang 2009] introduit une nouvelle notation des fonctions de base $B(3, i, j; t)$ pour $t_{i+j} < t < t_{i+j+1}$avec $j = (0, 1, \cdots, k)$. A partir cette formulation, les fonctions de base non nulles sur le tronçon d'interpolation $[t_{i+3}, t_{i+4}]$ sont donc :

— $B(3, i+3, 0; t)$
— $B(3, i+2, 1; t)$
— $B(3, i+1, 2; t)$
— $B(3, i, 3; t)$

En observant la formule de ces 4 fonctions de base, on constate que la formulation non uniforme utilise un plus grand nombre de dates de points de contrôle pour garantir une interpolation C2. En effet la fonction de base $B(3, i+3, 0; t)$ utilise les horodatages t_{i+3} à t_{i+6}, et la fonction de base $B(3, i, 3; t)$ utilise les horodatages t_{i+1} à t_{i+4}. Les horodatages de 6 PC $t_{i+1\cdots i+6}$ et les poses des 4 PC $\mathbf{T}_{w,i+1\cdots i+4}$ sont utilisés pour l'interpolation dans l'intervalle $[t_{i+3}, t_{i+4}]$.

Malgré ces différences, la propriété canonique des fonctions de base des B-Splines est toujours respectée dans le cas non-uniforme :

$$B(3, i, 3; t) + B(3, i+1, 2; 3) + B(3, i+2, 1; 3) + B(3, i+3, 0; 3) = 1 \tag{4.11}$$

A partir de ces formulations, il est possible d'utiliser à nouveau l'expression du temps intermédiaire entre les deux points de contrôle $\mathbf{P_{i+3}}$ et $\mathbf{P_{i+4}}$ du segment considéré :

$$u(t) = \frac{t - t_{i+3}}{t_{i+4} - t_{i+3}} \tag{4.12}$$

Ainsi si l'on souhaite obtenir une formulation matricielle similaire à celle des B-Splines uniformes ;

$$\mathbf{P}_w(t) = \begin{pmatrix} 1 & u(t) & u(t)^2 & u(t)^3 \end{pmatrix} \mathbf{M}^T \begin{pmatrix} \mathbf{P}_{i+1} \\ \mathbf{P}_{i+2} \\ \mathbf{P}_{i+3} \\ \mathbf{P}_{i+4} \end{pmatrix}, \mathbf{M} = \begin{pmatrix} M_{00} & M_{01} & M_{02} & M_{03} \\ M_{10} & M_{11} & M_{12} & M_{13} \\ M_{20} & M_{21} & M_{22} & M_{23} \\ M_{30} & M_{31} & M_{32} & M_{33} \end{pmatrix} \tag{4.13}$$

il est nécessaire de factoriser les équations 4.10 par $u(t)$. Les détails de la factorisation sont donnés dans[Yang 2009]. Les coefficients de la matrice $\mathbf{M}$ sont définis par :

$$\begin{aligned}
M_{00} &= \frac{u_{43}^2}{(u_{42}u_{41})} \\
M_{01} &= -3M_{00} \\
M_{02} &= 3M_{00} \\
M_{03} &= -M_{00} \\
M_{20} &= \frac{u_{32}^2}{(u_{52}u_{42})} \\
M_{21} &= 3u_{32}\frac{u_{43}}{(u_{52}u_{42})} \\
M_{22} &= \frac{3u_{43}}{(u_{52}u_{42})} \\
M_{23} &= -u_{43}^2\left[\frac{1}{(u_{52}u_{42})} + \frac{1}{(u_{63}u_{53})} + \frac{1}{(u_{53}u_{52})}\right] \\
M_{30} &= M_{31} = M_{32} = 0 \\
M_{33} &= \frac{u_{43}^2}{(u_{63}u_{53})}
\end{aligned} \tag{4.14}$$

et d'après les propriétés canoniques (voir eq. 4.11) des Bsplines :

$$\begin{aligned}
M_{10} &= 1 - M_{00} - M_{20} \\
M_{11} &= -M_{01} - M_{21} - M_{31} = 3M_{00} - M_{21} \\
M_{22} &= -M_{12} - M_{22} - M_{32} = -M_{00} - M_{22} \\
M_{13} &= -M_{03} - M_{23} - M_{33} = -M_{00} - M_{23} - M_{33}
\end{aligned} \tag{4.15}$$

4.1.1.1 Forme cumulative pour les Bsplines cubiques non-uniformes

Il est désormais possible d'obtenir une forme matricielle cumulative pour les B-Splines non uniformes à partir de la matrice des coefficients $\mathbf{M}$ de l'équation 4.13. Comme énoncé dans le chapitre 2, le calcul des coefficients $C_{i,j}$ cumulatifs à partir de la matrice des coefficients $\mathbf{M}$ s'effectue de la manière suivante :

$$C_{ij} = \sum_{l=i}^{k} M_{l,j} \tag{4.16}$$

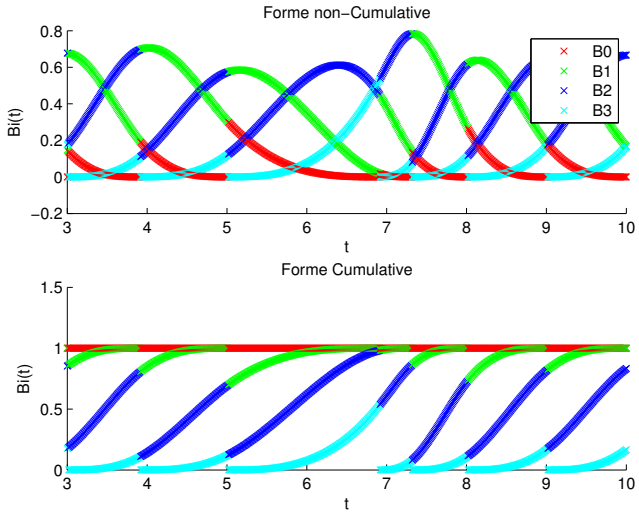


FIGURE 4.2 – Cette figure décrit l'évolution de la valeur des $k = 4$ fonctions de base non-uniformes cumulatives $\tilde{B}_{0..k}(t)$(haut) et non cumulatives $B_{0..k}(t)$(bas) en fonction du temps pour une DTNU des PC. A chaque fonction de base correspond une couleur (rouge,vert,bleue,cyan), illustrant l'influence d'une PC(bas) ou d'une variation de PC(haut) sur la trajectoire interpolée au cours du temps.

Alors la matrice cumulative est de la forme :

$$\mathbf{C} = \begin{pmatrix} C_{00} & C_{01} & C_{02} & C_{03} \\ C_{10} & C_{11} & C_{12} & C_{13} \\ C_{20} & C_{21} & C_{22} & C_{23} \\ C_{30} & C_{31} & C_{32} & C_{33} \end{pmatrix} \tag{4.17}$$

Pour simplifier les calculs, remarquons que :

$$\begin{aligned}
C_{00} &= M_{00} + (1 - M_{00} - M_{20}) + M_{20} = 1 \\
C_{01} &= M_{01} + M_{11} + M_{21} + M_{31} = \overbrace{-3M_{00} + 3M_{00}}^{=0} \overbrace{-M_{21} + M_{21}}^{=0} + \overbrace{M_{31}}^{=0} \\
C_{02} &= M_{02} + M_{12} + M_{22} + M_{32} = \overbrace{3M_{00} - 3M_{00}}^{=0} \overbrace{-M_{22} + M_{22}}^{=0} + \overbrace{M_{32}}^{=0} \\
C_{03} &= M_{03} + M_{13} + M_{23} + M_{33} = \overbrace{-M_{00} + M_{00}}^{=0} \overbrace{-M_{23} - M_{33} + M_{23} + M_{33}}^{=0} \\
C_{30} &= C_{31} = C_{32} = M_{30} = M_{31} = M_{32} = 0
\end{aligned} \tag{4.18}$$

On obtient alors une forme convenable avec une complexité calculatoire

équivalente à la formulation non-cumulative :

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ M_{10} + M_{20} & M_{11} + M_{21} & M_{12} + M_{22} & M_{13} + M_{23} + M_{33} \\ M_{20} & M_{21} & M_{22} & M_{23} + M_{33} \\ 0 & 0 & 0 & M_{33} \end{pmatrix} \tag{4.19}$$

Ces coefficients, à l'inverse du cas uniforme, sont dépendants des horodatages des PC, impliquant un coût calculatoire supplémentaire dû à la nécessité de recalculer la matrice de coefficients pour chaque quartet de poses. Le vecteur des fonctions de base non nulles $\tilde{B}(t)$ s'exprime par :

$$\tilde{\mathbf{B}}(t) = \mathbf{C} \begin{bmatrix} 1 \\ u(t) \\ u(t)^2 \\ u(t)^3 \end{bmatrix} \tag{4.20}$$

La figure 4.2 illustre l'évolution de la valeur des 4 fonctions de base cumulatives (bas) et non cumulatives (haut) au cours du temps.

La matrice cumulative des coefficients C ne dépendant pas de t, il est possible de retrouver les dérivées des fonctions de base en fonction de t à l'aide d'une formulation similaire à celle des B-Splines uniformes :

$$\dot{\tilde{\mathbf{B}}}(t) = \frac{1}{\Delta t_{i+2,i+3}} \mathbf{C} \begin{bmatrix} 0 \\ 1 \\ 2u(t) \\ 3u(t)^2 \end{bmatrix}, \ddot{\tilde{\mathbf{B}}}(t) = \frac{1}{\Delta t_{i+2,i+3}^2} \mathbf{C} \begin{bmatrix} 0 \\ 0 \\ 2 \\ 6u(t) \end{bmatrix} \tag{4.21}$$

Les dérivées première et seconde s'obtiennent en utilisant les équations 2.70 et 2.71 du chapitre 2.

4.2 DTNU et génération des poses de contrôle

La formulation non-uniforme des B-Splines autorise des intervalles temporels variables entre les PC. Il est nécessaire de définir un algorithme permettant de générer des PC de contrôle, de les déplacer temporellement ou de les supprimer. Nous proposons ici deux méthodes qui permettent la génération des PC et faisant intervenir des données inertielles ou une analyse de l'erreur de reprojection. La première est adaptée à des applications en ligne, les PC pouvant être générées en ligne en utilisant les données inertielles, alors que la seconde fait intervenir un processus itératif la rendant prédisposée pour une application hors ligne.

La DTNU des PC induit régulièrement la création de multiples PC au sein

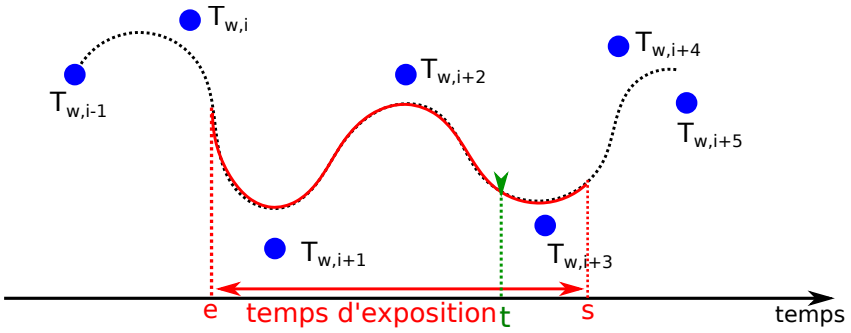


FIGURE 4.3 – Cette figure illustre la trajectoire interpolée à partir de PC (bleu) au cours de l'exposition d'une image (rouge). Le temps d'exposition de l'image comprend plusieurs intervalle temporelle de PC. Ainsi, dans cet exemple, plusieurs quartets/sextets de PC sont nécessaires pour interpoler les poses associées aux lignes d'une seule image.

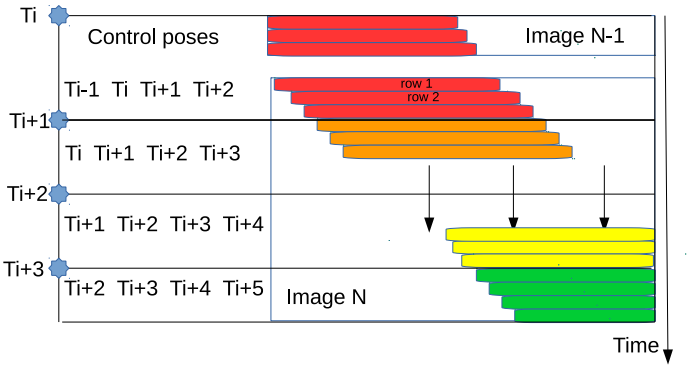


FIGURE 4.4 – Exemple de multiple PC au sein d'une image, et leur influence sur l'interpolation de pose pour différentes lignes de l'image. 3 PC sont horodatées entre le début et la fin de l'exposition de l'image. Ainsi le quartet de PC nécessaire à l'interpolation change 3 fois au cours de l'acquisition de l'image.

d'une même image, ce qui résulte en une multitude de quartets et sextets de poses nécessaires à l'interpolation pour différentes portions de l'image (voir figure 4.4 et 4.3). Ce cas est également possible pour des DTU si la fréquence d'échantillonnage des PC est supérieur à la fréquence image. La récupération du quartet/sextet de PC associé à une date t est effectuée pour chaque point d'intérêt dans chaque image. Il est donc nécessaire de prendre certaines précautions afin d'éviter une perte de temps précieuse dans cette opération, notamment en s'abstrayant d'une recherche globale sur toutes les PC. Ceci est accompli en conservant chronologiquement les indices des PC et en utilisant une table de hachage pour un accès rapide aux PC souhaitées.

4.2.1 Méthode inertielle

Nous proposons d'analyser les données d'accélérations et de vitesses angulaires fournies par une centrale inertielle afin de déterminer quand l'ajout d'une PC est nécessaire. Deux seuils, un pour la vitesse angulaire et un pour la vitesse linéaire sont mis en place afin d'estimer quand la dynamique du mouvement est trop forte en comparaison de la fréquence initiale des PC et le contexte. La valeur de ces seuils est définie arbitrairement et nécessite d'être adaptée selon les applications.

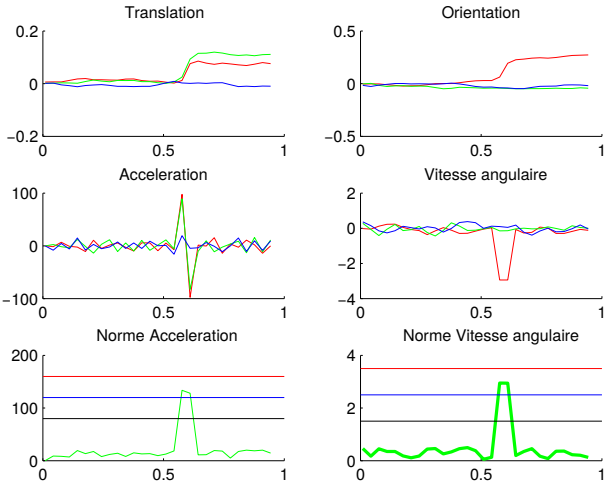


FIGURE 4.5 – Cette figure illustre des données inertielles interpolées à partir de données d'une trajectoire simulées. A gauche les composantes en translation(haut) et accélération(milieu) x(rouge) y(vert) et z(bleu), à droite les composantes en rotation(haut) et vitesse angulaire(milieu). Les deux courbes vertes du bas illustrent la norme de l'accélération en cm/s^2 (à gauche) et la norme de la vitesse angulaire en rad/s(à droite). Différents niveaux de seuil(rouge, bleu, noir) sont fixés afin de générer une ou plusieurs PC quand ils sont dépassés.

La figure 4.5 montre une trajectoire générée ainsi que les données inertielles associées. Les données inertielles sont générées par interpolation en utilisant les équations 2.70 et 2.71. Les différents seuils (noir, bleu et rouge) sont associés dans une table de correspondance à un nombre de PC à générer.

Afin de tester cette méthode, 30 PC sont générées synthétiquement suivant une trajectoire globalement linéaire et intégrant une ou plusieurs impulsions en translation. Ces PC sont initialement disposées dans le temps à intervalle constant $\Delta_t = 0.33s$, et sont ensuite affectées d'un bruit gaussien spatial. Une trajectoire caméra ainsi que des données inertielles sont interpolées à des fréquences de 30Hz et 100Hz respectivement. De nouvelles PC sont générées à intervalles temporels plus grand $\Delta_t = 0.1s$ et initialisées sur la trajectoire caméra de manière similaire à [Steven Lovegrove 2013]. Les données IMU sont

ensuite analysées et une ou plusieurs PC sont ajoutées entre les intervalles de PC où les seuils établis sont dépassés. Un nuage de points 3D synthétique est généré. Ces points 3D sont projetés dans les images à partir de poses caméra interpolées avec les PC non bruitées. Ces observations sont ensuite affectées d'un bruit gaussien pixellique. En considérant le nuage de points comme connu a priori, le problème du PnP est résolu afin de raffiner les PC. L'optimisation est réalisée en minimisant l'erreur entre les observations et la projection des points 3D par le modèle de caméra OD décrit précédemment. Les détails sur la résolution du PnP sont données dans la section suivante. L'association entre les observations et les points 3D est effectuée grâce à la synthèse et ne requiert donc pas de mise en correspondance. La vérité terrain est obtenue en interpolant la trajectoire caméra à partir des 30 PC initiales non bruitées.

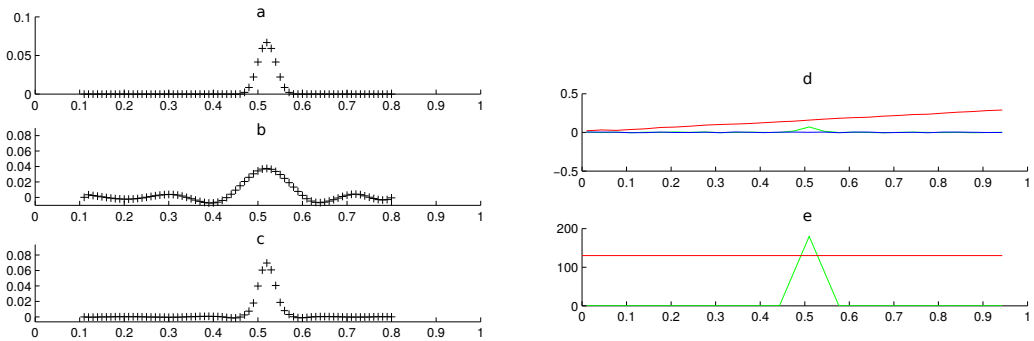


FIGURE 4.6 – a) illustre la trajectoire vérité terrain (noir), b) la trajectoire interpolée(noir) après optimisation des PC sans ajout de PC, c) la trajectoire caméra interpolée(noir) après optimisation des PC incluant celle ajoutée par notre méthode, d) la trajectoire en x (rouge) y (vert) z (bleu), e) la norme de l'accélération (vert) au côté du seuil défini (rouge) .

L'optimisation est effectué sur les PC bruitées avant l'ajout de la PC, puis après la génération de la PC. La figure 4.6 illustre les résultats obtenus après optimisation. L'ajout d'une seule PC permet une bien meilleure modélisation de la trajectoire.

Le concept de cette méthode est donc démontré et tout laisse à penser qu'elle s'adapte aux données réelles. La détermination des valeurs des seuils pour le cas des données réelles reste problématique et doit être réalisée avec minutie. Afin de ne plus dépendre de seuils arbitraires, nous proposons une seconde méthode utilisant l'analyse temporelle de l'erreur de reprojection.

4.2.2 Méthode par analyse de l'erreur

La méthode de génération de PC par analyse de l'erreur consiste à trouver les zones temporelles où l'erreur après optimisation est la plus élevée. Cette

méthode fait intervenir un processus itératif d'estimation de la trajectoire en partant d'une DTU des PC. Différentes erreurs peuvent être utilisées, telles que l'erreur de reprojection où l'erreur entre 2 poses caméras. Nous décrivons ici un procédé analysant l'erreur de reprojection afin de valider la méthode.

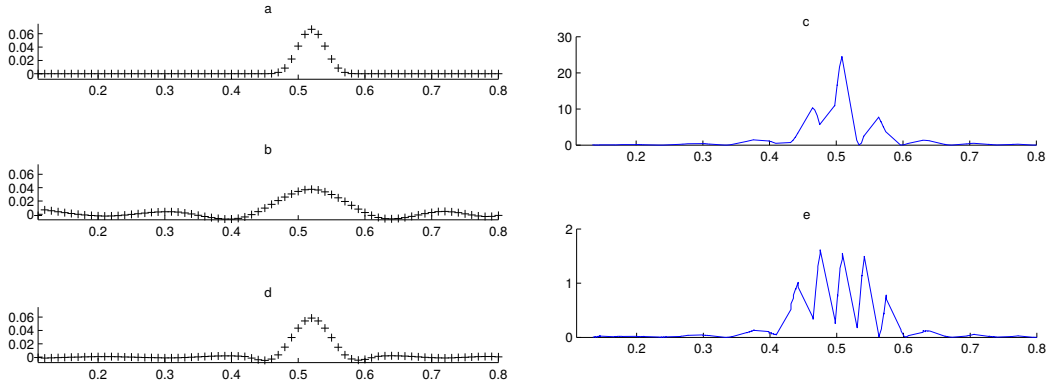


FIGURE 4.7 – Les graphiques à gauche illustrent la trajectoire vérité terrain (a), la trajectoire interpolée après optimisation sans ajout de PC (b) et avec ajout de PC par notre méthode (d). Les graphiques à droite montrent l'erreur de reprojection en fonction du temps sans (d) et avec (e) ajout de PC. On observe ainsi la réduction drastique de l'erreur de reprojection après un ajout d'une seule PC. Notons ici que l'échelle de l'erreur est divisée par 15 entre les figures c) et e).

En reprenant les données générées pour la méthode IMU, une première optimisation est effectuée sur les PC bruitées (sans ajout de PC). Les PC optimisées servent ensuite à calculer l'erreur de reprojection. Les erreurs de reprojection des images sont ensuite associées temporellement à des intervalles de Poses $\mathbf{T}_{w,i}, \mathbf{T}_{w,i+1}$ pour être sommées. Un point est ensuite ajouté soit dans l'intervalle temporel possédant l'erreur de reprojection maximale, soit dans tout les intervalles dont l'erreur dépasse un certain seuil. Cependant, nous préférerons l'approche où un seul point est ajouté, car elle permet d'éviter les problématiques liés au seuil (réglage, effet de seuil) et également d'avoir un aspect local de la modification qui rend le processus itératif plus robuste et plus facilement contrôlable. Seul le seuil associé à la condition d'arrêt de l'ajout de PC subsiste. Dans notre cas, ce seuil définit une erreur de reprojection globale minimale en dessous de laquelle les améliorations induites par l'ajout d'une PC seraient négligeables.

L'ajout d'un trop grand nombre de PC en une seule itération provoque une complexification du problème en créant des minimas locaux vers lesquels l'optimisation converge. Ceci vient principalement du fait que les PC ajoutées sont initialisées comme étant sur la trajectoire, sur un tronçon où l'erreur est très élevée.

La figure 4.7 met en lumière l'apport de l'ajout d'une PC de contrôle là

où l'erreur est la plus élevée. L'erreur de reprojection finale est drastiquement réduite, ce qui se traduit par une meilleur modélisation de la trajectoire interpolée. Les détails concernant l'optimisation sont données dans la suite.

4.3 Adaption d'algorithmes d'optimisation pour modèle DTNU

4.3.1 PnP

L'algorithme du PnP (Perspective-N-Points) permet d'estimer les paramètres extrinsèques d'une ou plusieurs caméras à partir de n correspondances entre des points 3D connus $\mathbf{P}_k$ de la scène et leurs observations 2D $\mathbf{p}_{i,k}$ dans les images.

A partir de ces correspondances et d'une estimée initiale de la pose caméra $\mathbf{T}_{w,c}$ peut être calculée une erreur de reprojection fonction du point k et de l'image i :

$$\mathbf{Err}(i,k) = \mathbf{p}_{i,k} - \pi([K|0]\mathbf{T_{w,c}}^{-1}\mathbf{P_k}) \tag{4.22}$$

En adaptant la formulation du PnP aux caméras OD, on obtient une formulation de l'erreur dépendante de la date d'exposition de la ligne sur laquelle est projeté le point :

$$\mathbf{Err}(i,k) = \mathbf{p}_{i,k} - \pi([K|0]\mathbf{T_{w,c}}(t)^{-1}\mathbf{P_k}) \tag{4.23}$$

Dans le cas du PnP OD, l'ensemble des paramètres θ à optimiser est un ensemble de PC et non plus directement des poses caméras. L'ensemble des paramètres raffinés $\hat{\theta}$ est obtenu en minimisant la somme des carrés des erreurs de reprojection :

$$\hat{\theta} = \arg\min_{\theta} \sum_i \sum_k \mathbf{Err}(i,k)^2 \tag{4.24}$$

4.3.2 Ajustement de faisceaux

L'ajustement de faisceaux est une méthode d'optimisation qui, à l'instar du PnP, permet de raffiner un ensemble de paramètres en minimisant une fonction de coût. A la différence du PnP, l'ajustement de faisceaux intègre également les points 3D de la carte dans les paramètres à optimiser. Les paramètres de l'ajustement de faisceaux sont donc définis par $\theta_b = [\theta, \mathbf{P}_0 \cdots \mathbf{P}_M]$, et sont raffinés par minimisation de l'erreur de reprojection.

La minimisation est effectuée par l'algorithme de Levenberg-Marquardt. Les paramètres de θ_b peuvent être initialisés à partir d'un SLAM/PnP utilisant

un modèle de caméra OG. Les PC contenues dans θ sont alors initialisées comme étant sur la trajectoire interpolée.

4.3.3 Optimisation de trajectoires

Le PnP et l'ajustement de faisceaux sont des méthodes de minimisation d'une erreur de reprojection $\mathbf{Err} \in \mathbb{R}2$ car elles supposent des données images en entrée. D'autres métriques peuvent être utilisées dans le cas de données de poses caméra en entrée. Il est possible d'affiner l'interpolation d'une trajectoire en minimisant une erreur $\mathbf{Err}_T(t_j) \in \mathbb{SE}(3)$ entre les poses interpolées $\mathbf{T}_{w,c}(t_j)$ et les poses caméras de référence $\mathbf{T}_{GT}(t_j)$ horodatées en t_j.

$$\mathbf{Err}_T(t_j) = log(\mathbf{T}_{GT}(t_j)^{-1}\mathbf{T}_{w,c}(t_j)) \tag{4.25}$$

A l'instar de l'ajustement de faisceaux et du PnP, la minimisation s'effectue par Levenberg-Marquardt, et permet de raffiner l'ensemble des paramètres des PC.

4.3.4 Optimisation spatio-temporelle des poses de contrôle

L'utilisation d'un modèle de B-Spline non-uniforme permet d'optimiser la composante temporelle des PC. Les horodatages des PC peuvent alors être ajoutés aux vecteurs des paramètres à optimiser, en conservant la même fonction de coût que précédemment. La datation des PC intervient uniquement dans le calcul de la matrice des coefficients M (eq. 4.13).

Cependant, l'ajout de ces paramètres doit être réalisé avec précaution, la modification de l'horodatage des PC par une mise à jour d'une itération de l'optimisation devant respecter un certain nombre de contraintes. Les intervalles temporels doivent rester positifs, et l'ensemble des dates d'exposition des lignes des images doit rester compris dans ces intervalles.

En pratique, l'optimisation simultanée des PC et de leurs horodatages converge régulièrement vers des solution erronée locale. Pour répondre à ce problème, nous proposons d'alterner des phases d'optimisation spatiale et temporelle des PC.

Ainsi, après une première optimisation spatiale, l'optimisation temporelle permet de sortir du minimum local et de converger vers une solution proche. Il est ensuite possible d'appliquer une optimisation uniquement temporelle ou spatio-temporelle de manière à affiner la trajectoire.

Les algorithmes présentés nécessitent une étape de minimisation des résidus. Différentes approches sont possibles pour représenter ce problème. Nous détaillons par la suite une représentation par graphe de l'optimisation pour notre modèle de caméra OD par B-Splines non-uniformes.

4.3.5 Représentation par graphe

La méthode par graphe présentée au chapitre 2 est étendue afin d'intégrer notre modèle de caméras OD à DTNU des PC. Nous utilisons pour cela une version modifiée de g2o [Kuemmerle 2011] afin de satisfaire nos contraintes. En effet, dans le cas d'une paramétrisation en temps continu de la trajectoire caméra, la représentation par graphe du problème diffère du modèle classique présenté dans le chapitre 2.

Notre cas d'interpolation cubique présume l'utilisation de 4 PC pour l'interpolation de la pose de caméra. Dans le graphe, les nœuds représentent les PC et non directement les poses associées aux images clés. La représentation des amers par des nœuds reste inchangée. En revanche, l'observation d'un amer sur une ligne de l'image contraint 4 PC. Une arrête symbolisant une contrainte d'observation doit donc désormais relier un point de la carte à 4 PC (voir fig 4.8). L'outil logiciel g2o permet par le biais des "multi-arêtes" de représenter aisément ces arêtes qui relient plus de deux nœuds entre eux.

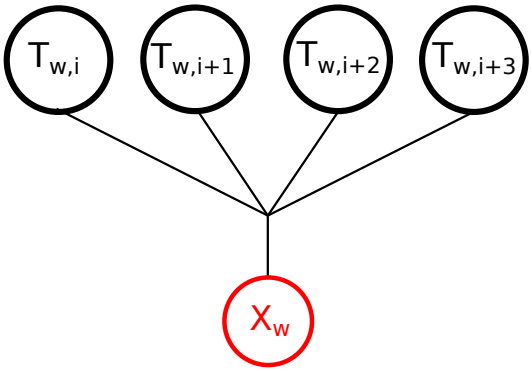


FIGURE 4.8 – Exemple de graphe traduisant l'observation d'un amer X_w sur une ligne d'image exposée à un temps $t_{i+2} < t < t_{i+3}$. La pose associée à cette ligne est contrainte par 4 PC $\mathbf{T}_{w,i..i+3}$.

Pour la cas de l'optimisation spatio-temporelle, il est nécessaire d'ajouter les 6 horodatages des PC comme étant contraints par l'observation d'un point de la carte. Une multi-arête doit alors relier les paramètres spatiaux de 4 PC et les paramètres temporels de 6 PC. La figure 4.9 illustre bien la complexité induite par l'ajout des horodatages.

Le découplage des optimisations spatiale et temporelle peut être obtenue en fixant les nœuds associés aux horodatages ou aux PC lors des phases d'optimisation.

A partir de ces différentes arêtes, il est possible de construire un graphe pour les problèmes d'ajustement de faisceaux et d'optimisation temporelle. L'outil logiciel d'optimisation de graphes g2o fourni automatiquement les dérivées numériques, en calculant les jacobiennes par différences finies. Ce pro-

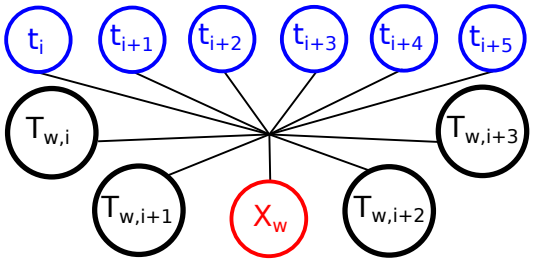


FIGURE 4.9 – Exemple de graphe traduisant l'observation d'un amer X_w dans une image où la pose est contrainte par 4 PC $\mathbf{T}_{w,i..i+3}$ et 6 horodatages de PC $t_{i..i+6}$.

cessus est relativement coûteux, puisqu'il oblige à évaluer la fonction de coût en faisant varier infinitésimalement chacun des paramètres à optimiser. Le calcul des dérivées analytiques est cependant très complexe dans les modèles de caméra OD. Nous choisissons pour prototyper et valider nos hypothèses d'utiliser dans un premier temps les dérivées numériques, le calcul des dérivées analytiques étant abordé en annexe.

4.4 Résultats

4.4.1 Tests sur données synthétiques

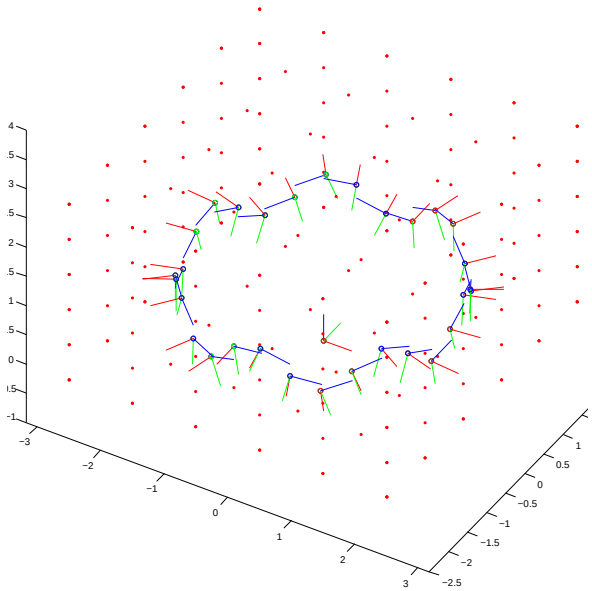


FIGURE 4.10 – Exemple de PC qui suivent une trajectoire circulaire où la composante en translation z suit une sinusoïde et l'axe z de la caméra est aligné avec la tangente à la courbe. Le nuage de points est affiché en rouge.

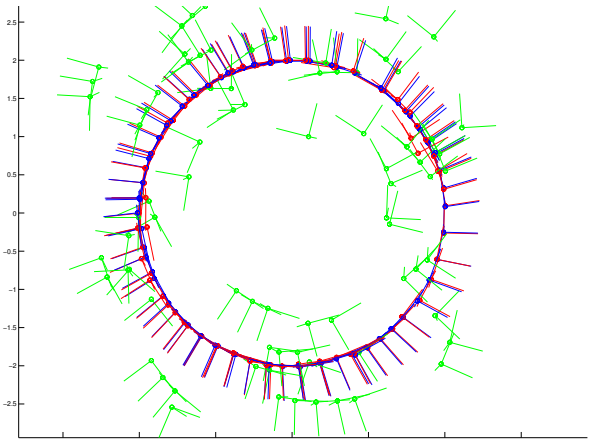


FIGURE 4.11 – Les poses de caméras interpolées à partir des PC initiales (vert), des PC optimisées (bleu) et de la vérité terrain (rouge).

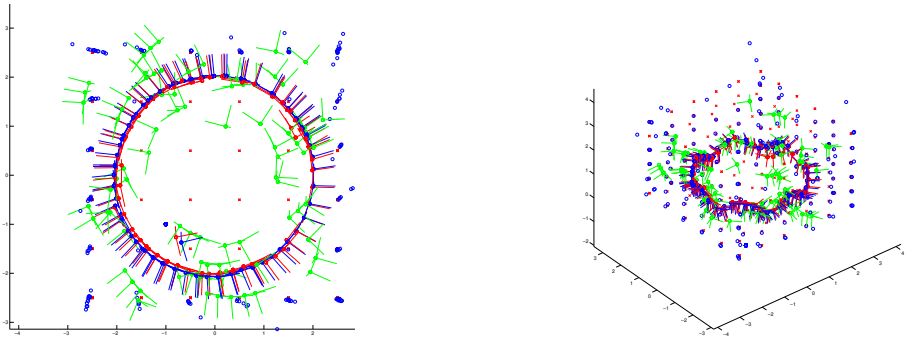


FIGURE 4.12 – Vues de dessus (gauche) et de côté (droite) des résultats de l'optimisation non-uniforme. Les poses de caméras interpolées à partir des PC initiales (vert), des PC optimisées (bleu) et à partir de la vérité terrain (en rouge) sont affichées aux côtés des points optimisés de la carte.

Nous choisissons ici de travailler sur des données simulées afin de se soustraire aux problématiques de traitement d'image et d'association de données. Les expérimentations se concentrent sur le comportement de notre modèle pour les problèmes du PnP et de l'ajustement de faisceaux.

Dans un premier temps, des PC sont générées suivant différents types de mouvements (linéaires, circulaires ou aléatoires). De plus, un nuage de points 3D représentant un modèle géométrique d'environnement simple est créé(voir fig. 4.10).

Les PC sont initialisées au niveau des poses interpolées (rappelons que la trajectoire ne passe généralement pas par les PC). Les PC et les points 3D sont ensuite affectés d'un bruit gaussien spatial. Les intervalles temporels entre les PC sont générés aléatoirement afin d'obtenir une DTNU permettant de tester et valider notre modèle.

Nous choisissons arbitrairement la fréquence de la caméra ($fps = 3$), ainsi

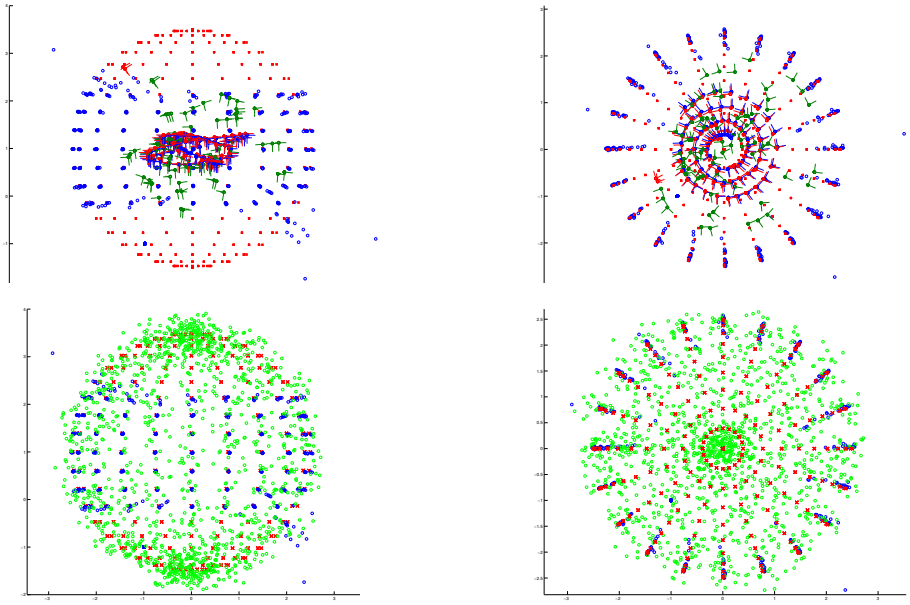


FIGURE 4.13 – Résultats des poses caméra et du nuage de points après optimisation sur une autre séquence. Les figures du haut illustrent l'ensemble des poses caméras avant optimisation (vert), après optimisation (bleu) et la vérité terrain (rouge). Les figures du bas illustrent le nuage de point avant optimisation (vert) et après optimisation (bleu) ainsi que la vérité terrain (rouge).

que les temps de départ et de fin auxquels la caméra acquière des images. Nous interpolons à l'aide de ces paramètres les poses de caméras et nous projetons les points dans les images avec un modèle similaire à celui décrit dans [Vandeportaele 2017] et [Steven Lovegrove 2013]. Ces données sont également bruitées, puis utilisées comme mesures dans un schéma d'optimisation. Les données sont générées sous MATLAB.

Dans le cas du PnP, nous utilisons des poses interpolées à partir des PC bruitées comme estimées initiales des PC. Une centaine d'images interpolées observant un nuage de points non bruités sont générées à partir de 60 PC bruitées. Le PnP est résolu sur l'ensemble des images afin de raffiner les PC. La résolution est effectuée par notre optimiseur développé en C++ et basé sur g2o [Kuemmerle 2011].

Sur la figure 4.11 les poses de caméras initiales (en vert) convergent après optimisation (en bleu) vers la vérité terrain (en rouge). Cinq itérations d'optimisation suffisent à obtenir des poses de caméras fidèles à la vérité terrain malgré des poses initiales très bruitées.

Pour tester l'ajustement de faisceaux, nous utilisons les mêmes trajectoires simulées que pour le PnP, en intégrant cette fois les points bruités à l'initialisation qui seront raffinés par l'optimisation. L'ajustement de faisceaux est également résolu par notre optimiseur utilisant g2o [Kuemmerle 2011]. Les

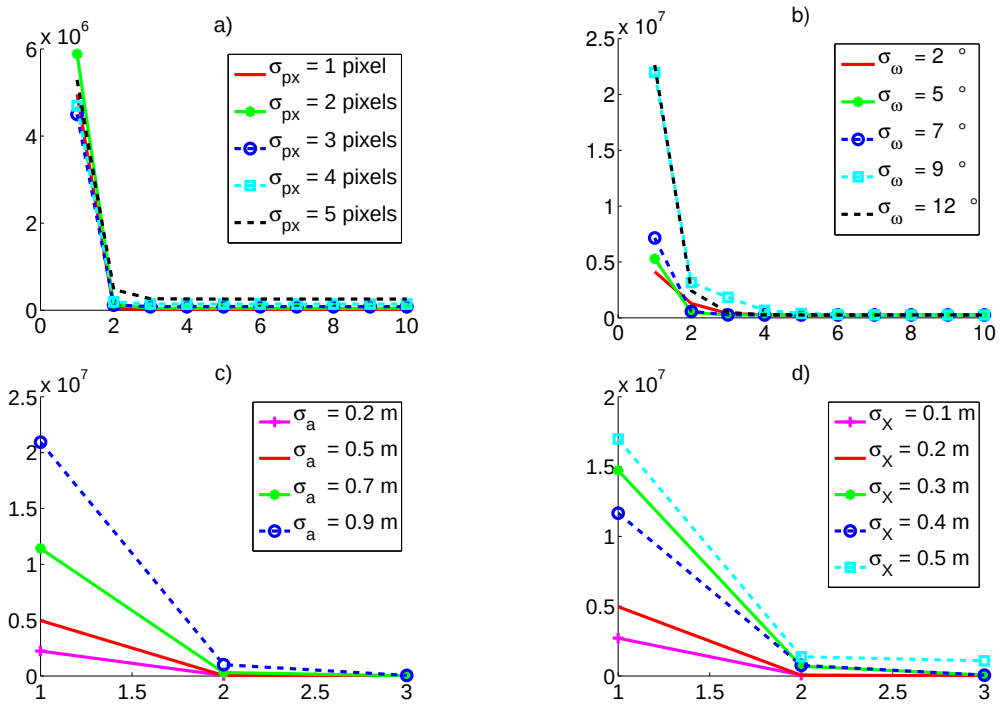


FIGURE 4.14 – Evolution de l'erreur au cours des itérations de l'optimisation suivant différents niveaux de bruits. Détails dans le texte.

résultats obtenus après optimisation sont exposés dans la figure 4.13 où l'on observe la convergence des points et des caméras optimisés (en bleu) vers la vérité terrain (en rouge). La colonne de droite montre uniquement les nuages de points bruités initiaux (en vert), optimisés (en bleu) et de la vérité terrain (en rouge). La colonne de droite montre uniquement les nuages de points bruités initiaux (en vert), optimisés (en bleu) et de la vérité terrain (en rouge).

L'ensemble des bruits ajoutés sur les données en entrée sont décrits par une loi normale centrée en 0. Les écarts-types choisis pour réaliser les figures précédentes étaient de $\sigma_{px} = 2px$ pour le bruit pixellique, de $\sigma_\omega = 5°$ pour le bruit sur l'orientation des PC selon chacun des axes, de $\sigma_{\mathbf{a}} = 0.5m$ pour le bruit sur la position des PC, et enfin $\sigma_{\mathbf{X}} = 0.2m$ pour le bruit sur les points 3D. La figure 4.14 expose la convergence de l'optimisation en faisant varier indépendamment ces niveaux de bruit sur les données initiales. Les figures montrent l'évolution de l'erreur en fonction des itérations de l'optimisation suivant le bruit sur les observations (a), sur les rotations (b) et translations (c) des PC, et sur le nuage de points (d). L'optimisation est robuste à différents niveaux de bruit, et converge vers des solutions satisfaisantes en très peu d'itérations (entre trois et cinq itérations).

Au delà des valeurs maximales pour les écarts-types données dans la figure, l'ajustement de faisceaux converge généralement vers des solutions erronées.

4.4.2 Tests sur données réelles

Afin de motiver notre méthode ajoutant la composante temporelle des PC à l'optimisation, nous avons dans un premier temps cherché à adapter une trajectoire interpolée sur la trajectoire réelle d'une caméra tenue à la main. Les données de cette trajectoire servant de vérité terrain sont obtenues par un système de capture de mouvements (MOCAP VICON) fournissant une trajectoire échantillonnée à une fréquence de 200hz. Diverses séquences ont été réalisées en suivant le mouvement d'une caméra tenue à la main, avec des dynamiques différentes en rotation et translation.

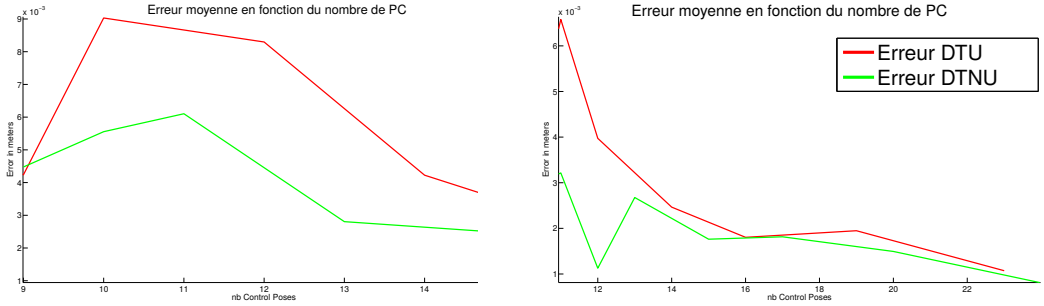


FIGURE 4.15 – L'erreur moyenne sur la translation en fonction du nombre de PC avec une distribution temporelle uniforme (rouge) et non-uniforme (vert).

Nous tentons de faire correspondre aux trajectoires réelles une trajectoire interpolée avec une DTU et une DTNU des PC et suivant différentes démarches d'optimisation de ces PC. Les PC sont initialisées sur la trajectoire MOCAP à des temps uniformément répartis. L'erreur $\mathbf{Err}_T \in \mathbb{R}6$ entre les poses interpolées et la vérité terrain de l'équation 4.25 est minimisée pour l'ensemble des horodatages t_j de la trajectoire MOCAP :

L'optimisation est réalisée par l'algorithme Levenberg-Marquardt avec un calcul des jacobiennes aux différences finies. Cette optimisation permet de raffiner spatialement les PC. Comme décrit dans [Vandeportaele 2017], une PC est ensuite ajoutée entre les deux PC où l'erreur après optimisation est la plus élevée.

La figure 4.15 illustre l'erreur moyenne en translation obtenue après optimisation spatiale en fonction du nombre de PC, pour une DTU (rouge) et pour une DTNU (vert). Comme attendu, à nombre de PC égal, il est possible de modéliser plus fidèlement une trajectoire avec une distribution non uniforme des PC qu'avec une distribution uniforme. Cette méthode ajoute des PC à un temps arbitraire entre les horodatages des deux PC environnantes. La distribution temporelle peut alors être affinée en optimisant les horodatages des PC.

L'optimisation offre dans la majorité des cas un gain substantiel qui se traduit par une diminution significative de l'erreur. Cependant de meilleurs résultats sont obtenus en suivant un schéma d'optimisation alterné. Une pre-

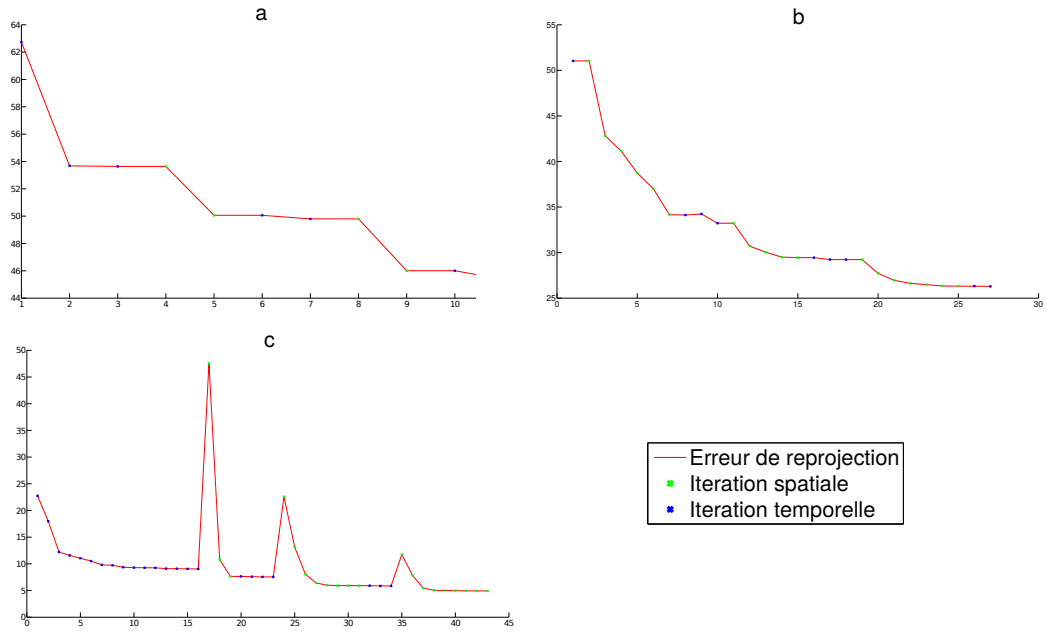


FIGURE 4.16 – Évolution de l'erreur sur la translation (rouge) en fonction des itérations de l'optimisation spatiale (vert) et temporelle (bleu). L'optimisation suit un schéma d'optimisation temporelle puis spatiale(haut) ou un schéma avec ajout de PC(bas).

mière optimisation spatiale est réalisée jusqu'à convergence. On répète ensuite n fois les trois opérations suivantes :

- Optimisation temporelle
- Ajout d'une PC quand l'erreur est la plus élevée
- Optimisation spatiale

Des vue comparatives des trajectoires interpolées avant et après $n = 3$ cycles d'optimisation uniforme spatiale (en vert) et d'optimisation spatio-temporelle sont montrées dans les figures 4.17 et 4.18. La vérité terrain y est affichée en rouge. La répartition temporelle et spatiale des PC s'adapte automatiquement à la dynamique de la trajectoire, ce qui est observable notamment sur les composantes en rotation des trajectoires de la figure 4.18). Les PC spatio-temporellement optimisées (bleu) se concentrent temporellement au moment des oscillations, ce qui permet leur modélisation là où l'interpolation uniforme échoue et lisse la trajectoire.

Pour une optimisation globale, il est approprié de privilégier dans un premier temps une optimisation spatiale des PC, pour ensuite réaliser des cycles d'optimisation spatio-temporelle afin de raffiner la trajectoire interpolée.

La figure 4.16 montre l'évolution de l'erreur de reprojection au fur et à mesure des itérations de l'optimisation. Les courbes du haut illustrent une optimisation saptio-temporelle sans ajout de PC intermédiaires. Les différents cycles d'optimisation spatiales (vert) et temporelle (bleu) sont distinguables.

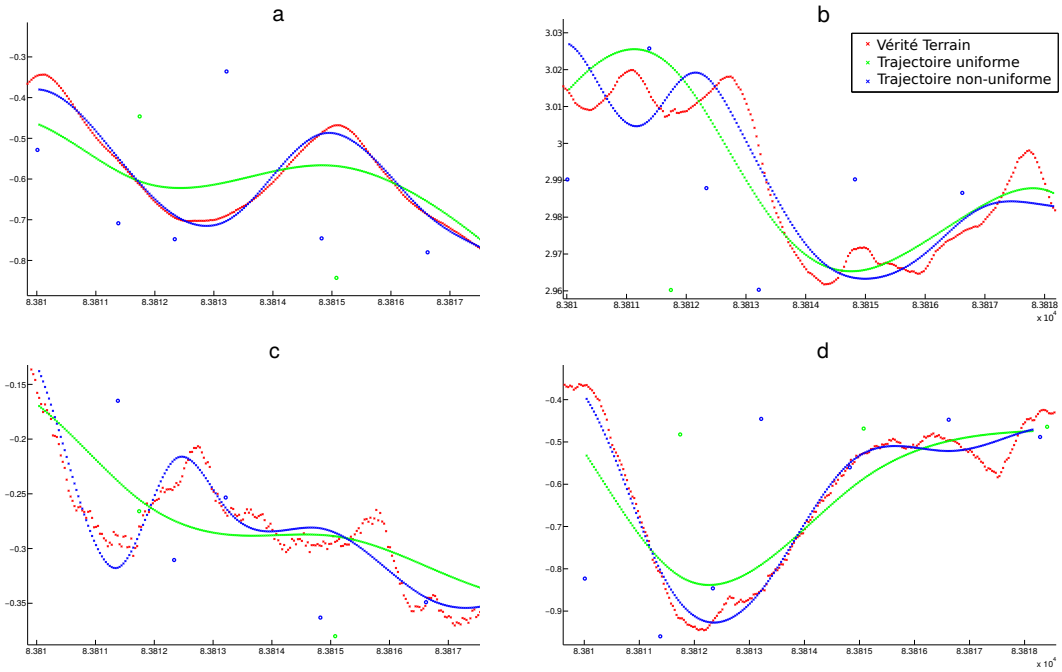


FIGURE 4.17 – Les composantes (X, Y) en translation (Haut) et en rotation (Bas) des trajectoires interpolées sur la séquence 1.

La figure du bas montre le comportement du cycle d'optimisation proposée avec ajout de PC. L'augmentation momentanée de l'erreur est engendrée par l'ajout d'une nouvelle PC initialisée sur la courbe qui est rapidement optimisée spatialement. Il est important de noter que ces optimisations sont réalisées après la convergence d'une optimisation spatiale globale, et l'optimisation temporelle permet alors de sortir du minimum local atteint.

Cette expérience démontre qu'il est possible d'adapter la disposition spatio-temporelle des PC à la dynamique d'une trajectoire réelle par le biais d'un processus d'optimisation. Le modèle d'interpolation cubique permet une modélisation fidèle de la trajectoire. La fonction de coût utilisée ici est relative à une vérité terrain qui n'est pas accessible dans le cas d'un processus de SLAM. Des tests approfondis doivent être menés en intégrant l'optimisation temporelle dans un ajustement de faisceaux. Passer de la minimisation d'une erreur sur $\mathfrak{se}(3)$ à une distance sur le plan image pour l'erreur de reprojection peut éventuellement poser problème pour des configurations dégénérées qu'il serait nécessaire d'évaluer.

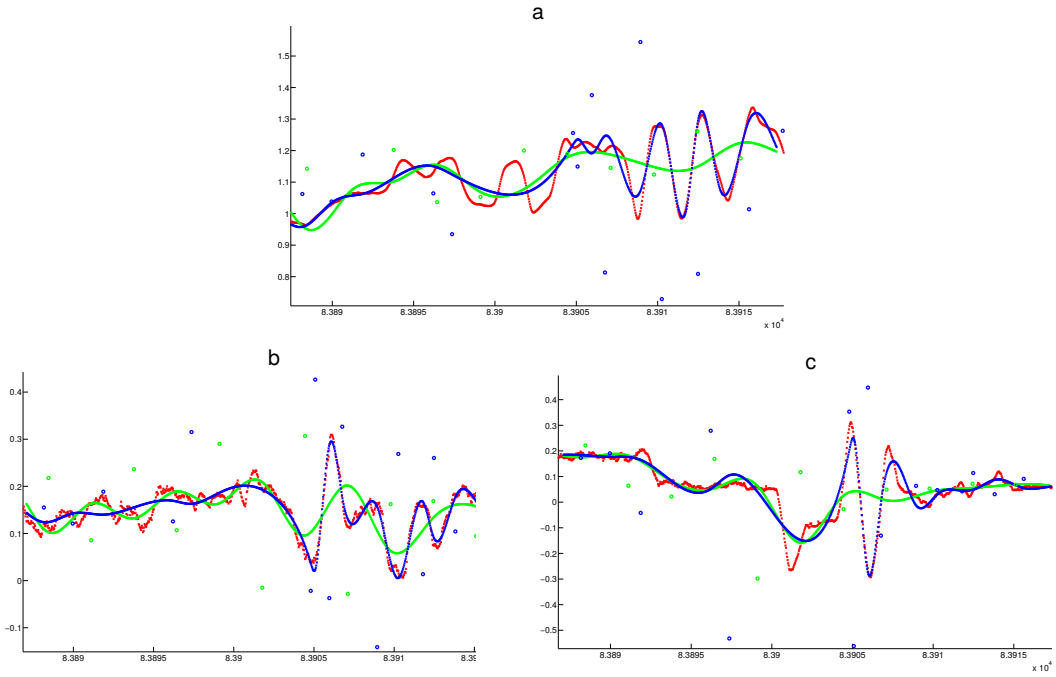


FIGURE 4.18 – La composante (Z) en translation (Haut) et les composantes (X, Y) en rotation (Bas) des trajectoires interpolées sur la séquence 4.

4.5 Discussion

4.5.1 Utilisation de primitives segments

La méthode d'optimisation par mesure de vraisemblance est uniquement réalisé sur des observations de points. Il est possible d'enrichir le modèle d'environnement par des primitives géométriques de plus haut niveau, et plus particulièrement les lignes. En effet dans le contexte d'intérieur, les environnements sont généralement peu texturées et le suivi de point peut très facilement défaillir lors de l'exploration d'une pièce. En revanche, de nombreuses lignes sont généralement visibles et suivables dans ces environnements. Ces méthodes ont été très étudiées dans la littérature, mais leur adaptation pour des modèles de caméra OD n'a pas été portée à notre connaissance.

Pour projeter un segment de droite, il est possible d'interpoler la position des extrémités de ce segment dans l'image à partir de nos modèles de projection OD. Les poses caméras peuvent ensuite être interpolées pour l'ensemble ou une sous partie des lignes comprises entre les projections des extrémités. La figure 4.19 illustre la projection d'une grille observée par une caméra en mouvement avec des durée d'exposition différentes.

Il est éventuellement possible d'utiliser un modèle OG pour la projection des extrémités et obtenir la ligne de départ et de fin dans l'image. Étant donné que pour la mise en correspondance, un échantillonnage le long du segment va

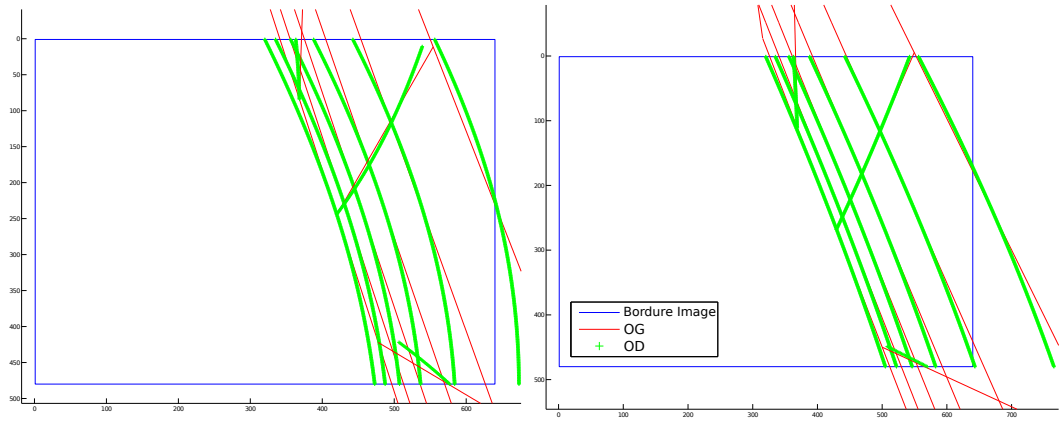


FIGURE 4.19 – Projection d'une grille observée par une caméra en mouvement à l'aide d'un modèle de projection OD(vert) et OG(rouge). A gauche la durée d'exposition de l'image est de 32ms à gauche et 10ms à droite. Ces temps d'exposition correspondent à des temps d'exposition communs respectivement en intérieur et en extérieur.

être effectué, la projection sur l'ensemble des lignes n'est pas nécessairement utile. De plus, la longueur des segments dans l'image n'est généralement pas invariante selon les points de vue.

Cette approximation est donc généralement suffisante si le segment projeté dans l'image n'est pas trop horizontal.

4.5.2 Coût calculatoire

Le coût calculatoire de la projection OD d'un amer pour une formulation en temps continu est plus élevée que le modèle OG car il fait intervenir plusieurs PC et suppose un plus grand nombre d'opérations. Cependant, dans le cas d'une optimisation globale, nous pensons que la définition en temps continu avec une DTNU des PC peut diminuer le coût calculatoire pour les applications où la trajectoire est partiellement lisse. C'est particulièrement le cas des longues trajectoires accomplies par des véhicules non-holonomes comme les voitures sur autoroutes. En effet ces trajectoires peuvent être globalement représentées par très peu de PC, et l'évaluation des jacobiennes est alors peu coûteuse. De plus, pour ce type de trajectoire, le coût mémoire s'en retrouve largement diminué.

4.6 Conclusion

Nous avons proposé dans cette section un modèle de B-Spline cumulative non uniforme permettant de conserver la continuité C2 de la trajectoire. Ce modèle a été testé pour ajuster des trajectoires interpolées sur des trajectoires

réelles avec différentes démarches d'optimisation. Nous avons montré l'apport de l'optimisation des horodatages des PC pour l'interpolation de trajectoires réelles. Nous avons également proposé un ajustement de faisceaux utilisant le modèle de B-Spline non uniforme et exposé son comportement sur des données simulées suivant différents niveaux de bruit.

Le chapitre suivant aborde les problèmes des méthodes existantes de reconstruction sur les scènes d'intérieurs, et différentes approches sont apportées afin de retrouver la structure des pièces considérées.

Chapitre 5

Reconstruction en milieux intérieurs

Sommaire

5.1 Méthodes de reconstruction dense par vision 102
5.1.1 Reconstruction à partir d'images clés issues d'un SLAM . . . 102
5.1.2 Limitation de la méthode 103
5.1.3 Méthodes de reconstruction en intérieur 104
5.2 Modèles de pièces 105
5.2.1 Modèle parallélépipédique 105
5.2.2 Modèle plan du sol 106
5.2.3 Définition des prédictions et des observations 109
5.2.4 Points 3D 109
5.2.5 Segments et droites 3D 110
5.3 Détection automatique des observations dans l'image 112
5.3.1 Détection des plans principaux 112
5.3.2 Estimation de la structure de la pièce 114
5.3.3 Détection automatique des coins 116
5.4 Optimisation de la structure 119
5.4.1 Données en entrées 120
5.4.2 Optimisation du modèle parallélépipédique 121
5.4.3 Optimisation du modèle "plan du sol" rectiligne 122
5.4.4 Optimisation du modèle "plan du sol" libre 123
5.4.5 Discussion 126
5.5 Perspectives 126
5.5.1 Augmentation du nombre d'images 126
5.5.2 Augmentation du nombre de segments 127
5.5.3 Appariement de segments 3D-2D 128
5.5.4 Apprentissage automatique 130
5.6 Conclusion 130

Les travaux présentés dans les sections précédentes portent sur la modélisation des caméras OD pour éviter les impacts de l'obturateur déroulant dans les fonctions de localisation et reconstruction simultanées, notamment lorsque la caméra est déplacée rapidement par un opérateur. Ces modèles permettent de représenter plus fidèlement la géométrie de la caméra, mais

induisent cependant un coût calculatoire non négligeable. Bien que l'utilisation d'une distribution non uniforme des PC permettent de réduire le coût calculatoire dans certaines conditions, le problème d'une reconstruction dense ou semi-dense sans accélération matérielle est difficile[Kim 2016]. De plus, nos méthodes reposent sur l'utilisation d'appariement de points d'intérêts entre les images : un ensemble épars d'amers 3D ponctuels est généralement suffisant d'une part pour localiser la caméra (SLAM) ou pour estimer ses déplacements (odométrie visuelle) et d'autre part pour reconstruire partiellement l'environnement en intérieur. Cette représentation n'est pas suffisante pour reconstruire la structure complète de la pièce, notamment quand les méthodes de mise en correspondance échouent (murs uniformes ou portant des motifs répétitifs).

Certaines méthodes dédiées à la reconstruction en intérieur ont fait leur apparition [Gallup 2007],[Mičušík 2010] afin de répondre aux problèmes qu'ont les méthodes de reconstruction basées vision face à ces environnements. Nous proposons ici une méthode de reconstruction d'un environnement d'intérieur utilisant des modèles géométriques simples de pièces basées sur des a-priori (hypothèse monde Manhattan) et faisant interagir l'utilisateur.

5.1 Méthodes de reconstruction dense par vision

La reconstruction de l'environnement est un domaine très étudié dans la littérature, pour laquelle différentes méthodes ont été développées suivant différents cadres et contraintes. Dans le contexte monoculaire dans un environnement inconnu, les méthodes les plus communes sont les méthodes qui exploitent l'appariement de patchs d'image, la triangulation et l'ajustement de faisceaux [Furukawa 2010b].

5.1.1 Reconstruction à partir d'images clés issues d'un SLAM

Compte tenu des contraintes apportées par notre contexte d'application sur plateformes mobiles où la puissance de calcul disponible est limitée, les méthodes de reconstruction dense, généralement très coûteuses, n'apparaissent pas comme un choix viable en l'état.

Il est en revanche possible de délocaliser la reconstruction dense sur un serveur, et de réaliser uniquement la localisation en temps réel sur le dispositif mobile. L'application se chargeant ainsi d'envoyer un sous-ensemble d'images (les images clés) associées aux poses de la caméra, au serveur sur lequel seront exécutées la reconstruction dense et les autres opérations coûteuses. Cette méthode a notamment été exploitée par [LocherMichael 2016] afin de reconstruire des environnements extérieurs à partir d'une tablette.

Pour évaluer la reconstruction dense à partir d'images-clés issues d'un SLAM, un jeu de données d'images acquises par une caméra évoluant en milieu intérieur a été créé. La méthode ORB-SLAM est ensuite utilisée afin d'obtenir un ensemble d'images-clés, chacune associée à une pose caméra estimée par suivi de points d'intérêts à partir de ces images.

Ces image-clés associées aux poses de la caméra sont ensuite injectées en entrée dans VSFM, un logiciel libre de $Structure - from - Motion$ utilisant les méthodes CMVS (Clustered View for Multi-View Stereo)[Furukawa 2010a] et PMVS (Patch-based Mutli-View Stereo)[Furukawa 2010b] pour fournir une reconstruction dense.

5.1.1.1 PMVS et CMVS

Ces méthodes de reconstruction reposent sur un processus comprenant trois principales étapes également présentes dans le SLAM. Premièrement, des patchs d'image sont mis en correspondance entre les différentes images, en utilisant des méthodes plus coûteuses mais plus robustes que celles du SLAM, la contrainte sur le temps de calcul étant moindre. Les correspondances sont aussi calculées exhaustivement entre chaque paire d'images possible sans contrainte de voisinage, afin d'éventuellement repérer les fermetures de boucles. Ces correspondances sont ensuite utilisées afin d'estimer les positions relatives des caméras et reconstruire l'environnement par triangulation et estimation de la matrice fondamentale. Finalement, un ajustement de faisceaux global optimise la cohérence du modèle avec les observations. Contrairement au SLAM, l'ensemble des données en entrée sont considérées disponibles dès le départ, et sont donc traitées en lot et non à la volée.

Pour modéliser des environnements très larges, les problèmes d'optimisation deviennent trop coûteux à résoudre en mémoire et en temps de calcul. CMVS [Furukawa 2010a] propose de subdiviser le problème et de gérer la reconstruction par sous-ensemble d'images. Ces méthodes sont adaptées à des scènes texturées pour lesquels la mise en correspondance de patchs entre les images est aisément réalisable : les patchs doivent être observés depuis plusieurs points de vue, dissociables et invariants. La figure 5.1 illustre un intérieur reconstruit en utilisant VSFM sur un ensemble d'images-clés issues d'un SLAM.

5.1.2 Limitation de la méthode

Dans les milieux intérieurs, les plans principaux des pièces sont usuellement de couleur unie et possèdent donc très peu d'informations texturales à mettre en correspondance. Il est également commun d'avoir des motifs texturaux répétitifs tels les carrelages au sol. Le premier point reste toutefois

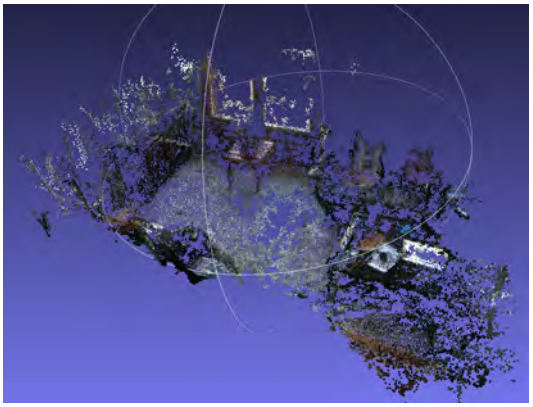


FIGURE 5.1 – Reconstruction dense par VSFM d'un salon en utilisant des images clés issues d'un SLAM. La structure de la pièce et particulièrement les murs ne sont pas reconstruits.

le plus problématique dans des contextes de reconstruction comme illustré dans la figure 5.1. La méthode parvient à reconstruire le plan du sol et la plupart des objets présents dans la pièce, qui sont généralement suffisamment texturés pour corréler des patchs entre les images. En revanche, les murs qui sont blancs unis et donc très peu texturés, ne sont pas reconstruits.

Par définition, ces méthodes nécessitent un recouvrement entre les images afin de pouvoir optimiser les poses de la caméra et trianguler les points 3D. Il est donc indispensable d'observer à partir de plusieurs points de vue différents une même partie de l'environnement. Cependant augmenter le nombre d'images implique d'augmenter le nombre d'opérations à réaliser, ce qui affecte le temps de calcul et la mémoire utilisée. De plus, obtenir une reconstruction satisfaisante pour estimer la structure de la pièce engage l'utilisateur à réaliser un jeu de données complet et complexe, selon un protocole auquel un usager lambda aura du mal à se conformer.

Ces problèmes sont connus et étudiés dans la littérature sur la reconstruction multi-vues, et différentes méthodes ont été proposées afin de parvenir à surmonter ces difficultés.

5.1.3 Méthodes de reconstruction en intérieur

Afin de simplifier le problème de la reconstruction, certaines hypothèses sont émises sur la structure de la pièce. L'hypothèse Manhattan suppose un environnement multi-planaire, composé d'un sol et d'un plafond parallèle, et de murs perpendiculaires à ces derniers et perpendiculaires entre eux.

En utilisant cette hypothèse, [Lee 2009] présente des travaux de reconstruction dense à partir d'une seule image, en considérant l'environnement comme purement multi-planaire et en utilisant une détection de points de fuite

[Rother 2002] afin d'estimer l'orientation relative entre les directions principales de la pièce et de la caméra. Des détails concernant cette méthode sont apportés dans la section suivante.

[Flint 2010a] ajoute à ces travaux une composante multi-vues, et présente une méthode d'estimation des points de fuite multi-vues ainsi qu'un extracteur de lignes dans l'image suivant les directions de fuite. Cette approche est améliorée [Flint 2010b] en utilisant une recherche exhaustive et déterministe des intersections des murs dans les images. Le temps de calcul est minimisé en exploitant des méthodes de programmation dynamique et le calcul d'intégrale d'images. Finalement , [Flint 2011] propose un modèle probabiliste qui incorpore des éléments monoculaires, stéréos et 3D.

[Mičušík 2010] propose d'utiliser une segmentation en superpixels (SLIC) pour grouper des pixels partageant des propriétés similaires, et d'utiliser des méthodes de balayage plans multi-vues [Gallup 2007] sur ces superpixels afin de déduire la profondeur à laquelle se trouve le plan. Les directions de balayage sont déduites à partir d'une estimation des points de fuite.

Similairement, [Concha Belenguer 2015] propose avec DPPTAM d'utiliser une segmentation en superpixels afin de reconstruire les plans non texturés dans l'image. Des points 3D reconstruits par un SLAM semi-dense (LSD-SLAM) sont associés aux contours de ces superpixels dans l'image. Un plan 3D est alors estimé à partir des points 3D reconstruits associés au contour sur lequel le superpixel peut ensuite être projeté.

Nous proposons dans la section suivante différents modèles de pièces permettant de représenter la structure 3D en utilisant différents a priori.

5.2 Modèles de pièces

Les méthodes de reconstruction multi-vues classiques sont capables de fournir un nuage de points dense des objets texturés présents dans la scène, mais en général, elles échouent à reconstruire un nuage de points dont la structure de la pièce serait extraite. Avant de proposer une méthode permettant de reconstruire cette structure de la pièce, différents modèles de structure de pièces 3D adaptés à notre contexte sont détaillés.

5.2.1 Modèle parallélépipédique

Un premier modèle simpliste de pièce d'intérieur peut être caractérisé comme un parallélépipède rectangle : 4 murs orthogonaux ou parallèles entre eux, un sol et un plafond orthogonaux aux murs.

Une pièce parallélépipédique $\mathbf{M}_p$ est alors définie par une longueur l, une largeur w, une hauteur h, un angle de rotation θ par rapport au repère monde, ainsi qu'un point 2D $\mathbf{p}_0 = [x_0, y_0]^T$ définissant la position du centre de la face

du sol par rapport au repère monde. Le plan du sol de la pièce est supposé aligné avec le repère monde. La pièce parallélépipédique $\mathbf{M}_p$ est définie par :

$$\mathbf{M}_p = (l, w, h, x_0, y_0, \theta) \tag{5.1}$$

Il est possible de relâcher la contrainte d'alignement du plan du sol au repère monde, en remplaçant le paramètre de rotation 2D dans le modèle par un vecteur de dimension 3, $log(\mathbf{R}) = [\omega_x, \omega_y, \omega_z]^T$ exprimant une rotation 3D (paramétrisation de Rodrigues).

$$\mathbf{M}_p = (l, w, h, x_0, y_0, \omega_x, \omega_y, \omega_z) \tag{5.2}$$

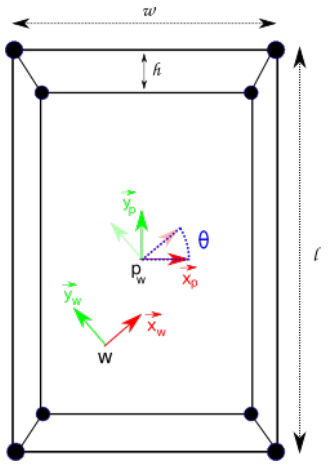


FIGURE 5.2 – Le modèle de pièce parallélépipédique est entièrement caractérisé par 6 paramètres représentés ici.

5.2.2 Modèle plan du sol

Afin de représenter des pièces possédant des structures plus complexes, un modèle, très utilisé dans la littérature, est celui du plan du sol. Ce modèle suppose une pièce dont les murs sont perpendiculaires au sol et au plafond, et ne possédant pas d'autre plan horizontal. Il est ainsi possible de paramétrer la pièce par une unique hauteur ainsi qu'un ensemble de paramètres définissant le plan du sol.

5.2.2.1 Polygone rectiligne

Un modèle plan du sol respectant l'hypothèse de Manhattan est celui du polygone rectiligne. Pour une pièce possédant un nombre n de coins, il est possible de la modéliser à l'aide d'un polygone rectiligne (2D) et d'un scalaire exprimant la hauteur.

Ce polygone rectiligne peut être représenté par un point 2D $\mathbf{p_0} = [x_0, y_0]^T$ correspondant à un premier point de référence, un vecteur $\mathbf{d}_{xy}$ de n scalaires réels correspondant à un déplacement en x ou en y dans le repère du polygone, et enfin un angle de rotation θ correspondant à l'angle entre l'axe x du repère monde et l'axe défini par les deux premiers points du polygone (voir fig 5.3). Le modèle possède donc $n + 2$ degrés de liberté :

$$\mathbf{M}_{pr} = (x_0, y_0, x_1, y_2, x_3, \ldots y_{n-2}, \theta, h) = (\mathbf{p_0}, \mathbf{d}_{xy}, \theta, h) \tag{5.3}$$

Le choix de l'alignement de la première arête du polygone le long de l'axe x est arbitraire. Il en découle que les indices pairs correspondent à un déplacement en x dans le vecteur de scalaires $\mathbf{d}_{xy}$, alors que les indices impairs correspondent à un déplacement le long de l'axe y. Un point d'indice $k \neq (0||n-1)$ dans le repère du polygone s'obtient de la manière suivante :

$$\begin{cases} \mathbf{p}_k = (\mathbf{d}_{xy}(k), \mathbf{d}_{xy}(k-1)) & \text{si k mod 2=0} \\ \mathbf{p}_k = (\mathbf{d}_{xy}(k-1), \mathbf{d}_{xy}(k)) & \text{sinon} \end{cases} \tag{5.4}$$

Un alignement de la première arête le long de l'axe y entraînerait une inversion de la correspondance des indices pairs et impairs. Un point $\mathbf{p}_k$ quelconque du polygone est reprojeté dans le repère monde à l'aide d'une simple rotation :

$$\mathbf{p}_w = \mathbf{R}(\theta) * \mathbf{p}_k \tag{5.5}$$

Le dernier point est calculé en prenant pour coordonnée y celle du point précédent et pour coordonnée x celle du premier point. Ce modèle est une généralisation du modèle de la pièce parallélépipédique aux modèles polyédriques rectilignes. En effet, pour $n = 4$ points, on obtient :

$$\mathbf{M}_{pr} = (x_0, y_0, x_1, y_2, h, \theta) \leftrightarrow (x_1, y_2, h, x_0, y_0, \theta) \tag{5.6}$$

où $x_1 = l$ et $y_2 = w$ (ou inversement). La paramétrisation du parallélépipède obtenu est la même que celle obtenue précédemment (avec 6 degrés de liberté).

5.2.2.2 Points 2D et hauteur

Il est possible d'obtenir un modèle plus permissif que les précédents en relâchant la contrainte d'orthogonalité des murs.

Si cette contrainte est relâchée, le modèle ne peut plus être simplifié en une suite de chemin suivant deux directions principales. Il est donc nécessaire d'augmenter les degrés de liberté du modèle en représentant chacun des coins de la pièce par des points 2D libres sur le plan du sol. Ce modèle de pièce plus

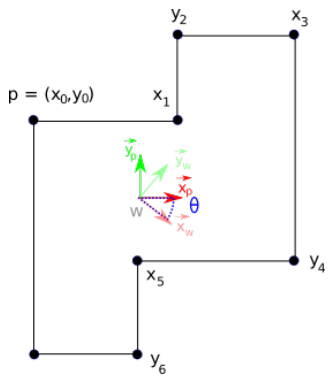


FIGURE 5.3 – Le modèle rectiligne est une extension du modèle parallélépipédique. En dehors du premier point, uniquement une coordonnée x ou y est nécessaire pour exprimer chaque autre point.

libre $\mathbf{M}_l$ est défini par :

$$\mathbf{M}_l = (\mathbf{p}_1, \mathbf{p}_2, \dots \mathbf{p}_n, h) \quad \text{avec} \quad \mathbf{p}_k = (x_k, y_k) \tag{5.7}$$

Nous avons pour ce modèle $2n + 1$ degrés de liberté. Ce modèle à l'avantage d'exprimer directement les points dans le repère monde, évitant ainsi les opérations de changement de repère.

La contrainte d'orthogonalité des murs peut être conservée en introduisant un terme de régularisation permettant de discriminer un angle qui ne serait pas droit (voir section suivante).

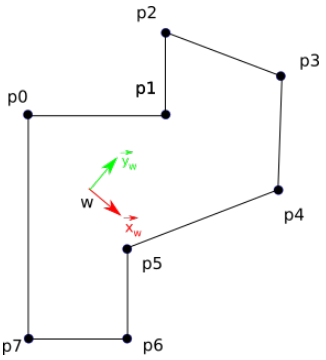


FIGURE 5.4 – Ce modèle libre permet de représenter un plus grand panel de pièces, où les points sont exprimés directement dans le repère monde. Il est possible d'y intégrer un terme de régularisation permettant de contraindre l'orthogonalité des coins souhaités.

5.2.3 Définition des prédictions et des observations

Ces modèles de pièces à divers degrés de liberté peuvent être utilisés dans un processus itératif d'optimisation. Le modèle spécifie les paramètres à optimiser selon un critère afin qu'il respecte au mieux les observations. Il est donc nécessaire de définir une fonction de coût à minimiser et le type d'observation à utiliser. Les modèles présentés précédemment permettent de définir des plans 3D, des segments 3D ainsi que des points 3D pouvant être reprojetés dans l'image et fournir des prédictions pour l'optimisation. Il reste donc à définir quelles prédictions utiliser à partir des observations obtenues dans les images.

5.2.4 Points 3D

Une métrique communément utilisée et présentée précédemment est l'erreur de reprojection. Elle nécessite pour être calculée d'acquérir les points 2D qui correspondent aux observations des coins de la pièce dans l'image (aussi bien ceux du sol que ceux du plafond), ainsi que les coins 3D associés du modèle reprojetés dans l'image. Ces reprojections s'obtiennent différemment selon le modèle choisi.

Pour le modèle parallélépipédique, il faut tout d'abord obtenir les points 3D dans le repère du parallélépipède :

$$\mathbf{M}_{0,p} = (\frac{-w}{2}, \frac{l}{2}, 0)^T \quad \mathbf{M}_{4,p} = (\frac{-w}{2}, \frac{l}{2}, h)^T \tag{5.8}$$

$$\mathbf{M}_{1,p} = (\frac{w}{2}, \frac{l}{2}, 0)^T \quad \mathbf{M}_{5,p} = (\frac{w}{2}, \frac{l}{2}, h)^T \tag{5.9}$$

$$\mathbf{M}_{2,p} = (\frac{w}{2}, \frac{-l}{2}, 0)^T \quad \mathbf{M}_{6,p} = (\frac{w}{2}, \frac{-l}{2}, h)^T \tag{5.10}$$

$$\mathbf{M}_{3,p} = (\frac{-w}{2}, \frac{-l}{2}, 0)^T \quad \mathbf{M}_{7,p} = (\frac{-w}{2}, \frac{-l}{2}, h)^T \tag{5.11}$$

Les points 3D sont ensuite projetés dans le repère monde à l'aide de la transformation du repère parallélépipédique p vers le repère monde w, $\mathbf{T}_{w,p}$:

$$\mathbf{T}_{w,p} = \begin{pmatrix} cos(\theta) & -sin(\theta) & 0 & x_0 \\ sin(\theta) & cos(\theta) & 0 & y_0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{5.12}$$

$$\mathbf{M}_{k,w} = \mathbf{T}_{w,p}\mathbf{M}_{k,p} \tag{5.13}$$

Ces points peuvent ensuite être projetés dans l'image à l'aide du modèle sténopée ou OD, connaissant la matrice des paramètres intrinsèques $\mathbf{K}$ de la caméra ainsi que la matrice de transformation du repère monde vers le repère

caméra $\mathbf{T}_{c,w} = (\mathbf{R}|\mathbf{a})$ (ou $\mathbf{T}_{c,w}(t)$) :

$$\lambda \mathbf{m}_k = \lambda \begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix} = \mathbf{K}(\mathbf{R}|\mathbf{a}) * \mathbf{M}_{k,w} \tag{5.14}$$

Comme énoncé précédemment, les points 3D associés au modèle de pièce rectiligne peuvent être récupérés à partir des paramètres du modèle et projetés dans le repère monde par simple rotation. La reprojection dans l'image peut donc simplement s'écrire :

$$\lambda \mathbf{m}_k = \lambda \begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix} = \mathbf{K}(\mathbf{R}|\mathbf{a}) * (\mathbf{R}(\theta) * \mathbf{M}_{k,w}) \tag{5.15}$$

Les points du modèle libre ont l'avantage d'être exprimés dans le repère monde. Il suffit alors uniquement de rajouter la composante en $z = (0|h)$ aux points 2D pour qu'ils puissent être projetés dans l'image en utilisant l'équation 5.14.

L'erreur de reprojection d'un point k du modèle dans une image i s'exprime par la distance au carré entre le pixel projeté (prédiction) $\hat{m}_{k,i}$ et le pixel observé $\mathbf{m}_{k,i}$:

$$err_{k,i} = (\hat{\mathbf{m}}_{k,i} - \mathbf{m}_{k,i})^2 \tag{5.16}$$

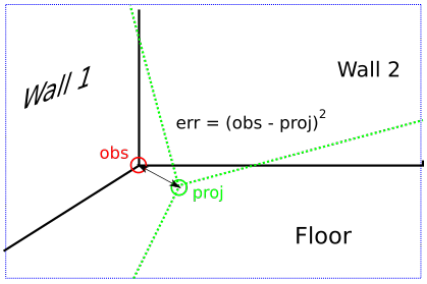


FIGURE 5.5 – Illustration de l'erreur de reprojection entre le point 2D du modèle reprojeté (vert) et l'observation (rouge). L'erreur de reprojection correspond à la distance euclidienne au carré.

5.2.5 Segments et droites 3D

L'utilisation des points 2D, pour représenter les coins possède l'avantage d'être simple à mettre en place, mais possède également certains inconvénients. Dans les milieux d'intérieurs, l'observation des coins n'est pas toujours possible, notamment à cause de l'ameublement de la pièce. Le mobilier régulièrement placé dans les coins d'une pièce rend impossible l'observation

du modèle, et donc son estimation. Il est possible d'utiliser des observations différentes en utilisant une métrique similaire (erreur de reprojection). Nous proposons d'utiliser les segments de droites en tant qu'observations, et des droites homogènes en tant que prédictions. Les observations de segments de droites correspondent à des intersections des plans principaux de la pièce, entre les murs et le sol ou les murs entre eux, par exemple.

Les points du modèle sont reprojetés dans l'image en utilisant les méthodes décrites précédemment. A partir de ces points 2D $\mathbf{m}_0, \mathbf{m}_1$ exprimés en coordonnées homogènes, une droite homogène correspondante est obtenue par produit vectoriel :

$$\mathbf{l} = (l_0, l_1, l_2)^T = \mathbf{m}_0 \cdot \mathbf{m}_1 = \begin{pmatrix} u_0 \\ v_0 \\ 1 \end{pmatrix} \times \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} \tag{5.17}$$

La normalisation de la droite homogène s'exprime alors par :

$$\mathbf{l}_n = (\frac{l_0}{\sqrt{l_0^2 + l_1^2}}, \frac{l_1}{\sqrt{l_0^2 + l_1^2}}, \frac{l_2}{\sqrt{l_0^2 + l_1^2}}) \tag{5.18}$$

La distance entre une droite homogène normalisée $\mathbf{l}_n$ et un point 2D en coordonnées homogènes $\mathbf{p}_h$ est définie par produit scalaire :

$$dist = \mathbf{l}_n \cdot \mathbf{p_h} = l_0 * u + l_1 * v + l_2 \tag{5.19}$$

L'erreur associée à un segment $\mathbf{f}_k$ est définie par la somme des distances entre ses extrémités $\mathbf{s}_f$ et $\mathbf{e}_f$ et sa droite homogène associée $\mathbf{l}_f$.

$$err_{i,k} = \mathbf{l}_f \cdot \mathbf{s}_f + \mathbf{l}_f \cdot \mathbf{e}_f \tag{5.20}$$

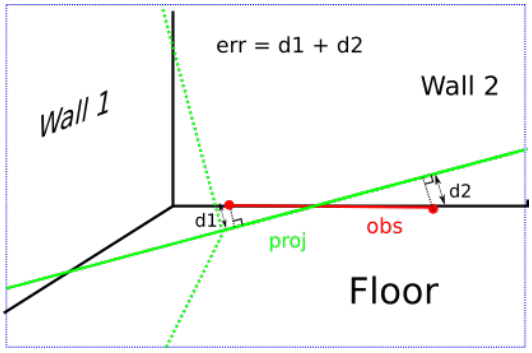


FIGURE 5.6 – Illustration de l'erreur de reprojection entre le segment observé (rouge) et la droite homogène reprojetée du modèle (vert). L'erreur de reprojection correspond à la somme des distances orthogonales entre les extrémités du segment et la droite homogène.

La fonction de coût associée au modèle est définie comme étant la somme des carrés des différences, et donc la somme des erreurs de reprojection $err_{i,k}$ associées aux segments observés $\mathbf{f}_{k,i}$ dans l'image i.

$$F(x) = \sum_{i,k} err_{i,k} \tag{5.21}$$

5.3 Détection automatique des observations dans l'image

Reconstruire un environnement d'intérieur à partir d'images monoculaires nécessite un modèle de la pièce ainsi que des observations associées. Les types d'observations pouvant être utilisés pour des modèles d'intérieurs ont été définis précédemment. Cependant, la manière d'obtenir ces observations n'a pas été précisée. Une première méthode simple, tenant compte de notre contexte dans lequel l'utilisateur peut interagir avec l'application, serait de demander à l'utilisateur de fournir ces informations.

Les méthodes pouvant être utilisées pour obtenir automatiquement ces observations sont détaillées par la suite.

5.3.1 Détection des plans principaux

Les observations précédemment décrites reposent sur les droites d'intersections entre les plans principaux de la pièce, où les coins représentent une intersection entre trois plans de la pièce. Ces observations reposent sur des intersections de plans. Une première route à suivre dans leur détection automatique est l'extraction des plans principaux dans l'image.

La détection de plans principaux, monoculaire ou multi-vues, est un problème qui peut être résolu de diverses manières suivant les données en entrée (RANSAC sur nuage de point, balayage plans, etc...). Cependant, leur détection à partir d'une image uniquement reste une tâche complexe, qui peut être partiellement résolue en milieu intérieur à l'aide des hypothèses de Manhattan.

5.3.1.1 Estimation des points de fuite

L'hypothèse de Manhattan sur les milieux intérieurs suppose que l'environnement soit aligné selon trois directions principales, orthogonales entre elles. Ces directions principales peuvent être représentées par une matrice de rotation 3D $\mathbf{R}_v$, et être estimées à partir des lignes de fuite dans l'image.

En effet, la caméra étant un appareil projectif, des droites parallèles dans le monde se projettent dans l'image en des droites se croisant en un point de fuite. Il est possible que le point de fuite dans l'image soit situé à l'infini si la caméra est fronto-parallèle à une direction principale.

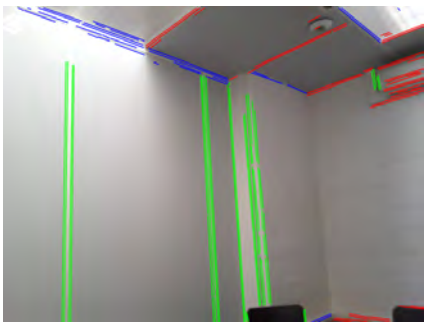



FIGURE 5.7 – Détection de points de fuite sur des images d'intérieur acquises par une plateforme mobile. Les trois directions principales sont affichées en rouge, en vert et en bleu.

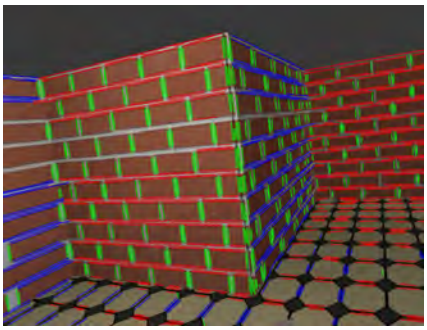
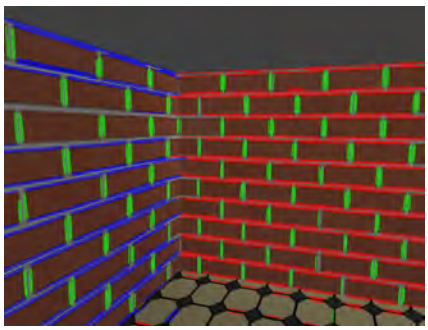



FIGURE 5.8 – Détection de points de fuite sur des images de synthèses générées par Blender.

Afin de détecter les points de fuite, les segments de droites doivent être extraits de l'image. LSD [von Gioi 2010] permet d'obtenir un plus grand nombre de segments que les plus anciens algorithmes [Canny 1987, Duda 1972]. Cette méthode est choisie afin de disposer d'un maximum de segments pour l'estimation des points de fuite.

Nous utilisons notre libre implémentation de la méthode robuste monoculaire de [Rother 2002] afin d'estimer ces points de fuite. Les points de fuite peuvent être utilisés afin d'estimer la focale pixellique de la caméra. Les figures 5.8 et 5.7 illustrent les résultats de notre implémentation sur des images synthétiques (5.8) et sur des images réelles (5.7).

5.3.1.2 Estimation de carte d'orientation

Les points de fuite encodent l'orientation relative entre la caméra et les directions principales de la pièce. [Lee 2009] utilise les lignes de fuite afin d'estimer l'orientation des pixels relativement aux directions principales. Cette carte d'orientation est calculée à l'aide d'un balayage des segments de droite

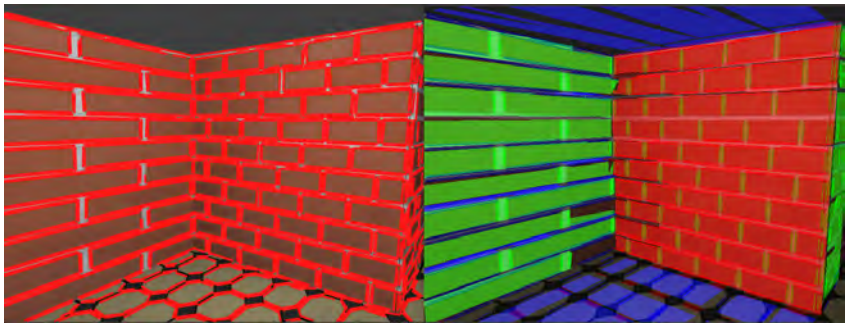


FIGURE 5.9 – Illustration des lignes détectés par LSD sur une image de synthèse (gauche), et de l'estimation de l'orientation des pixels de l'image (droite).

associés à un point de fuite vers les autres points de fuite (voir fig. 5.10).

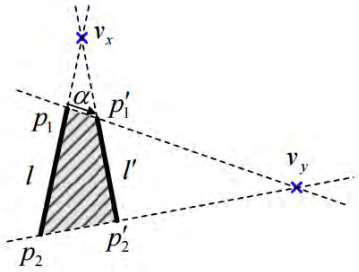


FIGURE 5.10 – Illustration de la méthode de balayage des segments de droite. Un segment de droite l associé à un point de fuite v_x est balayé vers un autre point de fuite, dans un sens puis dans l'autre. Les pixels compris dans la zone hachurée sont alors considérés comme étant possiblement orientés selon les deux directions principales associées à v_x et v_y. Image issue de [Lee 2009].

Ensuite, uniquement les cas possibles sont retenus afin de composer une carte d'orientation. Pour plus de détails, nous référons le lecteur à l'article de [Lee 2009]. Les figures 5.11 et 5.9 présentent les résultats de notre implémentation de cette méthode sur des images réelles et des images synthétiques.

5.3.2 Estimation de la structure de la pièce

La carte d'orientation permet d'avoir une estimée initiale sur l'orientation des pixels, qui peut être utilisée de différentes façons afin d'obtenir la structure de la pièce. [Lee 2009] et [Flint 2010a] l'utilisent comme donnée en entrée afin de calculer une fonction de coût permettant l'évaluation des hypothèses de modèles ou de structures de pièces (voir figure 5.16).

[Lee 2009] propose une approche de séparation et d'évaluation ($branch - and - bound$), qui consiste à tester toutes les combinaisons possibles de pièces

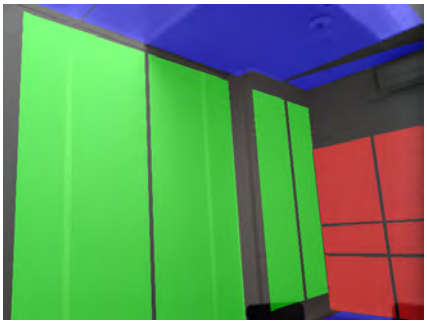
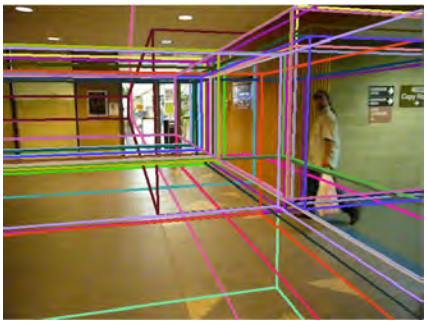


FIGURE 5.11 – La carte d'orientation estimée sur une image réelle. Bien qu'étant globalement correct, l'estimation échoue à estimer un des pans de la scène.

créées en intersectant les segments de fuite détectés dans l'image et en ne gardant que les combinaisons de pièces géométriquement correctes. Cette méthode est cependant très coûteuse, et possède une complexité en O(exp(n)), n étant le nombre de lignes de fuite.



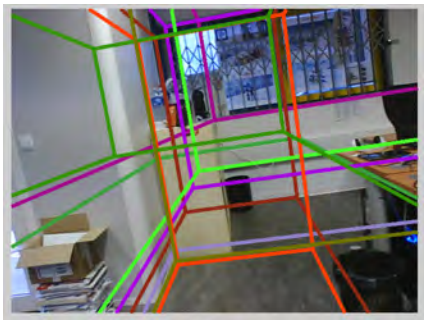


FIGURE 5.12 – Exemple des hypothèses de structures de pièces géométriquement valides obtenues par une approche exhaustive d'intersection des segments.

[Flint 2010b] propose une méthode d'estimation des intersections verticales entre les murs par balayage des colonnes de l'image. Pour ce faire, les images sont rectifiées par homologie pour que les lignes verticales soient alignées avec les colonnes de l'image. Cette méthode utilise en entrée la carte d'orientation afin d'évaluer les hypothèses générées. Chacune des hypothèses possibles, selon un certain nombre d'intersections (jusqu'à 5 dans l'article), est évaluée par un score de cohérence avec la carte d'orientation. Le calcul de ce score est également considérablement accéléré en pré-calculant les intégrales d'images de la carte d'orientation, permettant d'obtenir le nombre de pixels dans une zone rectangulaire en $O(1)$.

5.3.3 Détection automatique des coins

Nous proposons une méthode proche de celle décrite dans [Flint 2010b] et [Schwing 2013] pour la détection des intersections entre les plans principaux de la pièce. Cette méthode se base sur un balayage des lignes de fuite afin d'obtenir les intersections entre les plans les plus cohérentes avec la carte d'orientation et les segments 2D détectés dans l'image.

5.3.3.1 Balayage des lignes de fuite

Le but de cette étape est de balayer l'image selon une direction de fuite puis de calculer un score associé à cet angle représentant une droite d'intersection dans l'image (voir fig 5.13).

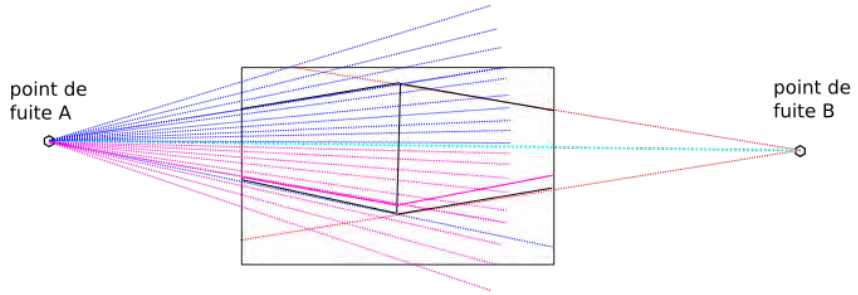


FIGURE 5.13 – Illustration du balayage des lignes de fuite associées au point de fuite A. Le balayage effectué au-dessus et en-dessous de l'horizon sont deux cas spécifiques. Le balayage au-dessus de l'horizon a pour but de trouver l'intersection avec le plafond, alors que le balayage en-dessous cherche l'intersection avec le sol.

Pour chaque point de fuite, les bordures correspondantes de l'image sont recherchées, et doivent être sélectionnées suivant la position du point de fuite dans le plan image (voir fig 5.14). L'angle entre les deux vecteurs formé par le point de fuite et les deux bordures de l'image correspond à l'angle de balayage α. L'image est ensuite balayée avec un pas angulaire $\gamma = \frac{\alpha}{nbStep}$ défini selon un nombre arbitraire de pas $nbStep$ choisi à l'avance.

La validité de l'intersection créée par chaque pas angulaire est ensuite évaluée par le biais d'un score qui définit la cohérence de cette hypothèse d'intersection aux observations disponibles.

5.3.3.2 Calcul du score

Le score choisi est une combinaison de deux scores : un score d'orientation et un score d'alignement.

Le score d'orientation est un score normalisé. Il est calculé en accumulant le nombre de pixels en accord avec la séparation de l'image créée par la droite balayée. La droite sépare l'image en deux parties A et B. Deux scores sont

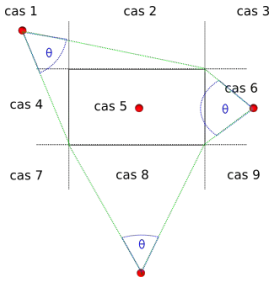


FIGURE 5.14 – Les différentes positions possibles d'un point de fuite, et les bordures sélectionnées associées. Ici uniquement trois cas sont illustrés, les autres étant des symétrie de l'un. Pour le cas où le point de fuite est dans l'image, il suffit de balayer sur 2π.

calculés en parallèle relativement à la carte d'orientation. Un premier score va s'incrémenter pour chaque pixel d'une orientation o_1 dans la zone A et chaque pixel d'orientation o_2 dans la zone B, et décroître pour chaque pixel d'orientation o_2 dans la zone A et chaque pixel d'orientation o_1 dans la zone B. Le second score est calculé de manière opposé. Le score d'orientation final est défini comme le score maximum de ces deux score :

$$score_1 = nb_pix(A, o_1) - nb_pix(B, o_2) \quad (5.22)$$
$$score_2 = nb_pix(A, o_2) - nb_pix(B, o_1) \quad (5.23)$$
$$score_o = max(score_1, score_2) \quad (5.24)$$

Le score d'orientation $score_o$ est ensuite normalisé en le divisant par le nombre de pixels ayant pour orientation o_1 et o_2.

Le score d'alignement est un score normalisé calculé en fonction du nombre de segments de fuite détectés dans l'image en accord avec la droite d'intersection. La distance entre les extrémités de chaque segment et la droite d'intersection est évaluée. Si la distance est inférieure à un certain seuil s_a, le score est incrémenté. Ce score d'alignement $score_a$ est ensuite normalisé en divisant par le nombre de segments associés à la direction de fuite.

Une combinaison linéaire de ces seuils peut être envisagée pour le calcul du score cumulé :

$$score = \omega score_o + \lambda score_a \quad (5.25)$$

Les poids ω et λ associés à ces scores doivent être choisis judicieusement selon le type de scène choisie. Deux approches peuvent être imaginées sur l'utilisation de ces scores. Soit ne sélectionner que les droites ayant les meilleurs scores pour chaque direction de fuite, soit sélectionner les droites ayant un score supérieur à un certain seuil.

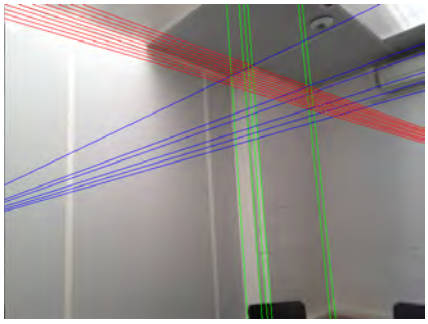


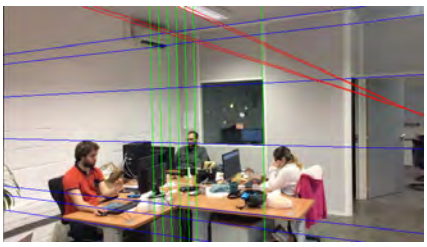
FIGURE 5.15 – Les hypothèses de droites intersectrices sélectionnées après le balayage. Ces droites sont celles ayant obtenues un score supérieur à un seuil pré-défini.

Deux scores sont conservés, le score cumulé et le score d'orientation, puis un seuil est défini pour chacun de ces scores au-dessus duquel une droite d'intersection est conservée. En effet, dans certains cas le score d'orientation permet de sélectionner des droites qui ne sont pas détectées par les méthodes de détection de droites dû à des gradients trop faibles (voir fig. 5.7 où la droite verticale principale n'est pas détectée par LSD).

Les résultats obtenus des lignes ayant un score suffisant sont présentés figure 5.15. La structure de la pièce est bien encodée dans les hypothèses résultantes.

5.3.3.3 Élimination des hypothèses faibles

Parmi les hypothèses de droites sélectionnées par les méthodes basées score, il est nécessaire de tester les triplets de droites $triplet = (l_{i,o1}, l_{j,o2}, l_{k,o3})$ pouvant fournir un coin de pièce valide. Les hypothèses qui ne respectent pas les contraintes géométriques énoncées dans [Lee 2009] peuvent d'ores-et-déjà être éliminées. Les triplets dont l'aire du triangle formé par l'intersection pair-à-pair de ces droites est trop étendue sont également éludés.



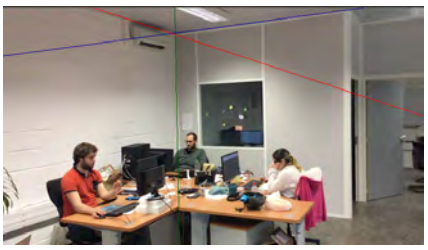


FIGURE 5.16 – Les droites candidates pour un coin de la pièce (gauche) et le triplet de droite ayant obtenu le meilleur score (droite).

Une fois le nombre d'hypothèses sélectionnées suffisamment restreint, une détection de points d'intérêts (Harris, GFFT ou FAST) est exécutée dans le

voisinage des hypothèses de coins. Uniquement les hypothèses possédant un point d'intérêt suffisamment proche sont validées.



FIGURE 5.17 – A gauche, une hypothèse respectant certaines conditions. A droite, les coins finaux détectés automatiquement dans l'image.

La figure 5.17 présente une hypothèse à gauche qui respecte les contraintes sur l'aire du triangle d'intersection, les contraintes géométriques, et les contraintes de cohérence avec la carte d'orientation. Cependant, elle ne respecte pas la contrainte imposant la proximité avec un point d'intérêt extrait dans l'image. Cette hypothèse n'est donc pas conservée. A droite, les 4 hypothèses de coins de la pièce en noir.

La méthode décrite ici donne des résultats satisfaisants dans le cas où les lignes de fuite sont suffisamment visibles et les coins ne sont pas obstrués. Ce n'est cependant pas toujours le cas dans notre contexte, particulièrement pour les cuisines où certains pans entiers de la structure de la pièce ne sont pas visibles ou identifiables. A l'instar des méthodes de [Lee 2009],[Flint 2010b] et [Schwing 2013], celle-ci reste sensible à l'estimation des points de fuite et de la carte d'orientation. C'est pourquoi nous choisissons d'inclure l'utilisateur pour confirmer ou fournir les observations dans le cas où leurs estimations échoueraient.

5.4 Optimisation de la structure

Dans les sections précédentes, les modèles de pièces, leurs fonctions de coût ainsi que les méthodes permettant d'obtenir les observations ont été présentés. Nous utilisons une représentation par graphe pour l'optimisation se basant sur g2o. Un nœud principal représente le modèle de pièce et ses paramètres à optimiser, et d'autres nœuds représentent les paramètres des caméras. Ce nœud principal est relié aux caméras par des arêtes représentant les contraintes d'observation. Une observation dans une image par une caméra d'un segment du modèle correspond à une arête.

L'optimisation a pour but d'optimiser les paramètres du modèle $\mathbf{M}$ en minimisant sa fonction de coût associée :

$$\mathbf{M}^* = argminF(\mathbf{M}) \tag{5.26}$$

L'optimisation est représentée par un graphe où chaque observation d'un segment ou d'un coin du modèle correspond à une arête reliant la pose de caméra associée et le modèle (voir fig. 5.18).

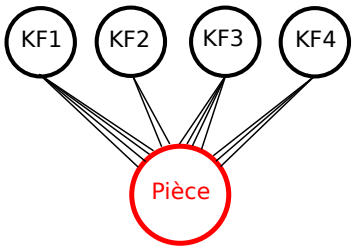


FIGURE 5.18 – Représentation par graphe de l'optimisation d'un modèle de pièce, ici une pièce parallélépipédique. Une arête relie la pièce à une des poses de la caméra pour chaque observation d'un segment ou d'un coin dans une image.

5.4.1 Données en entrées

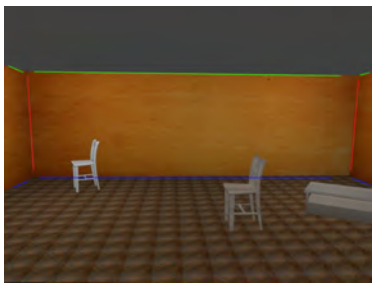


FIGURE 5.19 – Exemple de segments de droite que l'utilisateur peut fournir dans l'image. Ici toutes les lignes d'intersections ont été renseignées, cependant, il possible de fournir moins d'informations en entrée.

Afin de pouvoir reconstruire une pièce selon les modèles décrits précédemment, des données doivent être fournies en entrée pour pouvoir ensuite réaliser une optimisation à l'aide de ces données.

Pour une première version de la reconstruction de la structure de la pièce, nous choisissons un scénario simple où l'on dispose au minimum d'une image de chaque coin de la pièce et des paramètres caméras pour chaque image. L'utilisateur est donc invité à prendre des photos de chaque coin de la pièce, la pose caméra étant suivi à l'aide d'un algorithme de SLAM ou d'odométrie visuelle (OrbSLAM, Vuforia, ARKIT,ARCore). Les approches inertielles fournissant une échelle de la pièce sont privilégiées dans notre contexte d'ameublement où la métrique est importante. L'aspect modélisation de la caméra OD est dans

un premier temps éludé, étant donné que les mouvements de la caméra lors de l'acquisition des images de coins par l'utilisateur sont négligeables.

Il est ensuite nécessaire de disposer des observations des coins ou des droites d'intersections principales (voir fig 5.19). Ces observations peuvent être obtenues par le biais des méthodes automatiques détaillées précédemment ou fournies directement par l'utilisateur.

5.4.2 Optimisation du modèle parallélépipédique

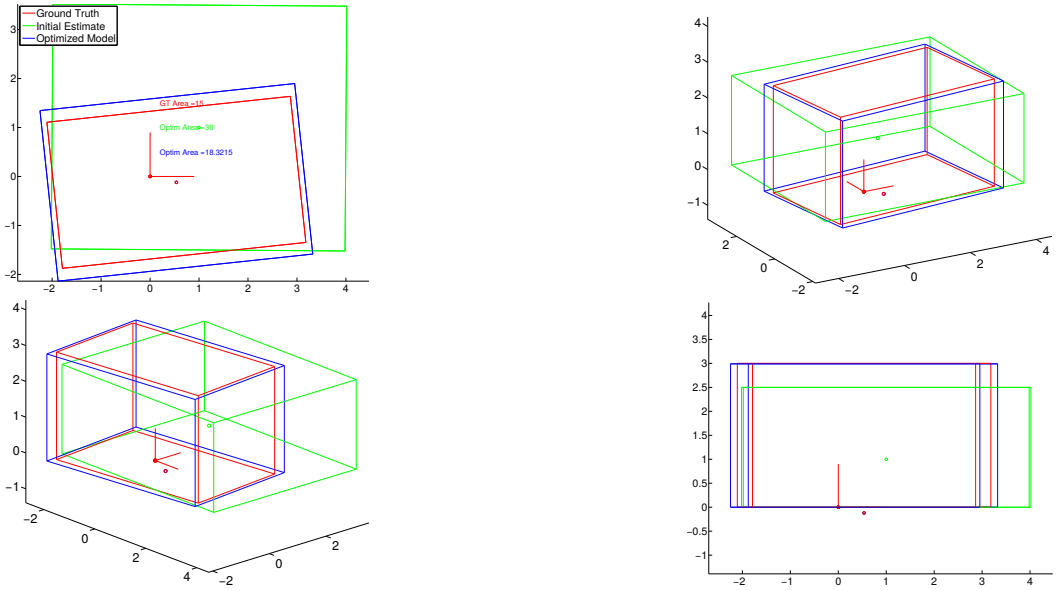


FIGURE 5.20 – Visualisation de la reconstruction en utilisant le modèle parallélépipédique. L'estimée initiale (vert) converge vers la vérité terrain (rouge) après optimisation (bleu).

Le modèle parallélépipédique est optimisé à partir d'une image par coin de la pièce, soit 4 images. Étant donné qu'il n'y a pas de recouvrement entre les images, et que la longueur et la largeur de la pièce ne sont pas observées, il est impossible d'optimiser conjointement les poses de la caméra et l'ensemble des paramètres du modèle. La largeur et la longueur sont donc totalement dépendantes de l'incertitude sur les poses de la caméra fournies en entrée. La figure 5.20 montre les résultats de l'optimisation du modèle sur des données réelles en fixant les poses de la caméra. Les poses fournies via un module SLAM exécuté indépendamment, permettent d'obtenir une estimée après optimisation qualitativement satisfaisante mais quantitativement peu fiable sur les paramètres non observés (4% d'erreur sur la longueur, 16% sur la largeur).

Cependant, si la longueur et la largeur sont observables dans les images, alors uniquement deux images (une observant la longueur, l'autre observant la



FIGURE 5.21 – Reconstruction de la structure à partir de 2 caméras observant l'ensemble des paramètres du modèle.

largeur) sont nécessaires pour optimiser simultanément le modèle et la position relative des caméras (voir fig. 5.21) .

5.4.3 Optimisation du modèle "plan du sol" rectiligne

Ce modèle est une généralisation du modèle parallélépipédique, et peut s'appliquer à des pièces à $n : n \mod 2 = 0$ coins. Disposant d'une image de chaque coin de la pièce et des observations fournies par l' utilisateur, le modèle rectiligne peut être optimisé en fixant les poses de la caméra.

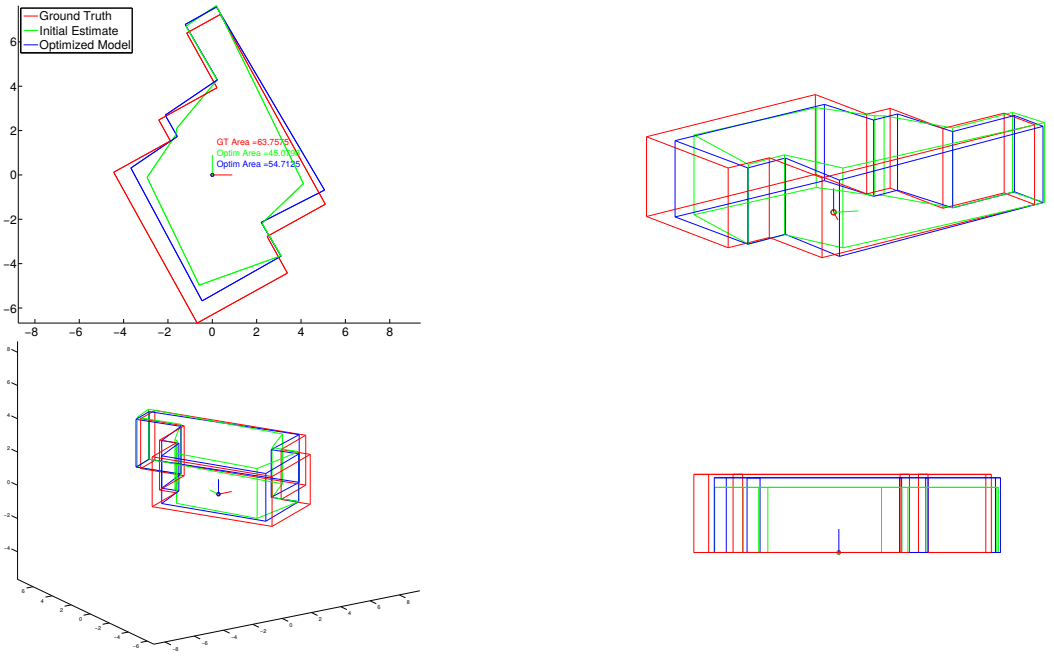


FIGURE 5.22 – Visualisation de la reconstruction en utilisant le modèle rectiligne. L'estimée initiale (vert) converge vers la vérité terrain (rouge) après optimisation (bleu).

La figure 5.22 illustre la convergence du modèle vers la vérité terrain. Certaines poses de la caméra sont mal estimées par le SLAM intégré dans le dispositif mobile, entraînant une mauvaise initialisation qui corrompt l'optimisation finale. L'erreur sur la surface au sol est de 14% après optimisation, contre 30% avant optimisation. Le résultat obtenu est cependant qualitativement satisfaisant, et peut être suffisant pour la gestion des collisions en réalité augmentée.

5.4.4 Optimisation du modèle "plan du sol" libre

Disposant de données en entrée similaires aux modèles précédents, l'optimisation sur le modèle libre offre des résultats sensiblement similaires au modèle rectiligne.

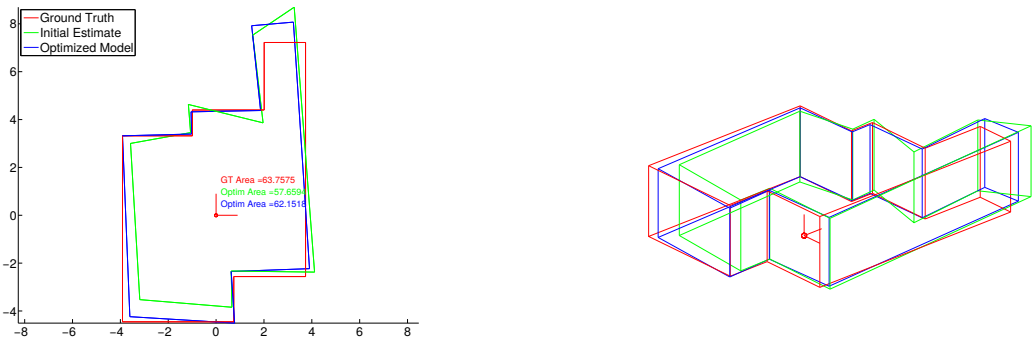


FIGURE 5.23 – Illustration de la converge de l'optimisation pour le modèle libre.

Sachant que le modèle à reconstruire respecte l'hypothèse d'un monde Manhattan, un terme de régularisation discriminant la non-orthogonalité des murs est ajouté. La fonction de coût est ainsi calculée non pas pour chaque segment, mais pour chaque paire de murs observés. Chaque paire de murs possède un segment d'intersection vertical, et chaque mur dispose d'un segment d'intersection avec le sol et d'un segment d'intersection avec le plafond, observé ou non.

Ainsi, pour une image observant un coin, l'erreur de reprojection associée à cette paire de murs est la somme des erreurs de reprojection associée aux segments d'intersections des murs. Cette erreur est ensuite multipliée par un terme de régularisation discriminant la non-orthogonalité entre ces murs. Soit $\vec{n}_{\Pi_{i-1}}$ la normale au plan Π_i associée au $i^{ème}$ mur, la fonction de coût devient :

$$F(x) = \sum_i (err_i * |(1 + (\vec{n}_{\Pi_{i-1}} \cdot \vec{n}_{\Pi_i}))| \tag{5.27}$$

Ce terme permet d'artificiellement forcer le modèle vers un modèle rectiligne, tout en gardant une certaine liberté si les données en entrée sont trop en contradiction avec l'orthogonalité des murs. Ce principe est illustré dans la fi-

gure 5.23. Les coins ne respectant pas l'orthogonalité sont ceux étant observés par une caméra lointaine, où l'erreur d'estimation de la pose caméra devient plus influente. Quelques exemples du modèle libre reprojeté dans l'image, avant et après optimisation sont illustrés figure 5.24.

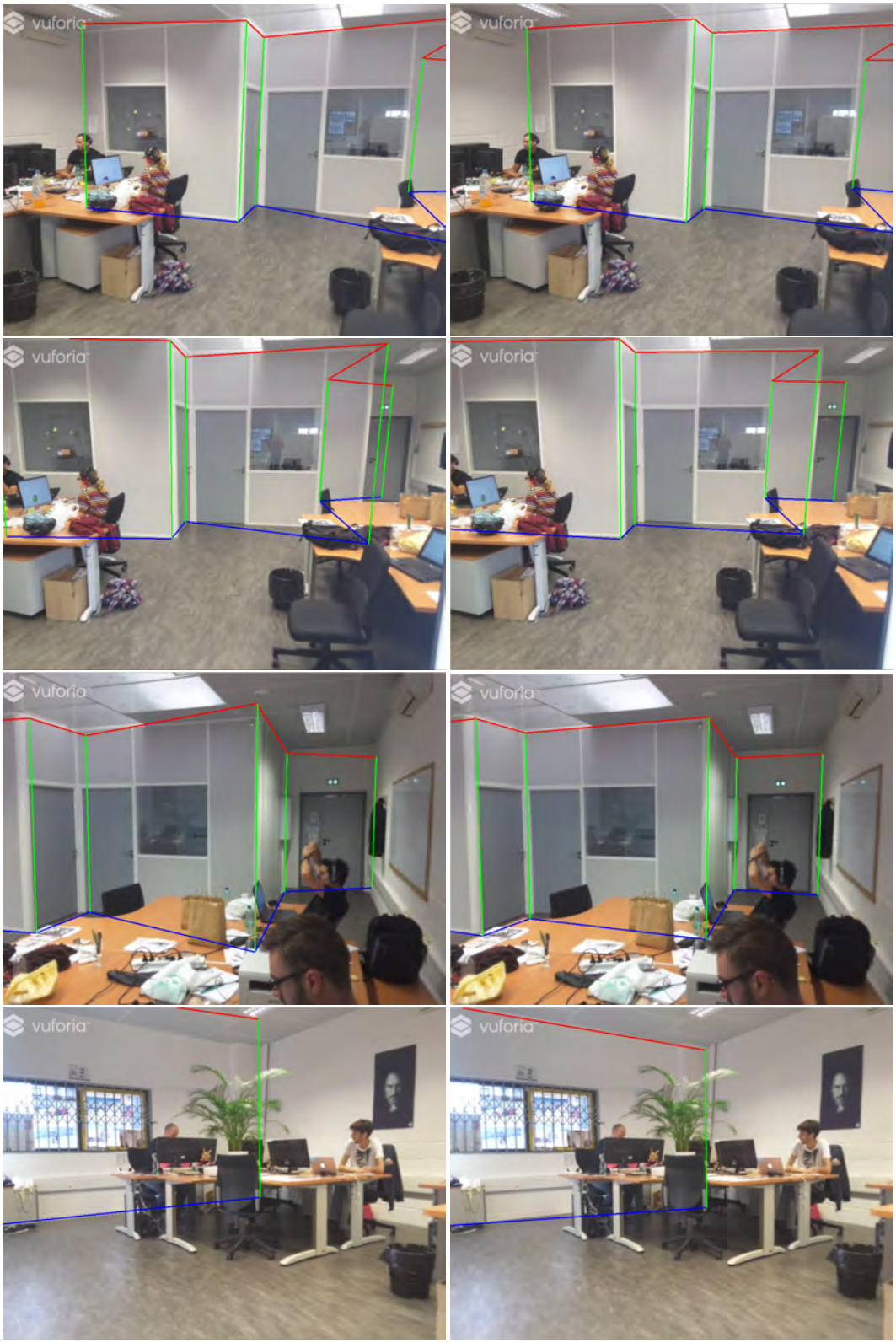


FIGURE 5.24 – Le modèle reprojeté avant optimisation (gauche) et après optimisation (droite).

5.4.5 Discussion

Nous avons montré dans cette section une méthode d'initialisation et d'optimisation de la structure avec un minimum de données en entrée (1 image par coin de la pièce, les poses de la caméra associées ainsi que les observations). Cette méthode offre des résultats satisfaisants mais reste cependant trop dépendante des estimations des poses de la caméra données en entrée. Excepté le cas où une observation de 2 coins est disponible dans chaque image, offrant un recouvrement entre les images, il est impossible d'optimiser les poses de la caméra. La reconstruction est ainsi décorrélée de la localisation, ce qui comme discuté dans l'introduction, n'est pas viable. En revanche, cette méthode offre tout de même une estimée de la reconstruction optimisée qui reste fidèle à la réalité. Nous proposons donc d'utiliser cette méthode pour obtenir l'estimée initiale de la structure, qui doit être enrichie en ajoutant des données supplémentaires en entrée.

5.5 Perspectives

Les modèles présentés sont très simplistes mais suffisent à représenter la majorité des pièces d'intérieur. En revanche, les données en entrée et les observations sont généralement insuffisantes pour optimiser simultanément la localisation des poses de la caméra et la reconstruction de la structure. Nous décidons donc d'augmenter le nombre de données en entrée en changeant légèrement l'acquisition du jeu de données, sans changer l'opération à réaliser par l'utilisateur.

5.5.1 Augmentation du nombre d'images

Indépendamment de l'utilisateur, des images associées aux poses de la caméra sont continuellement enregistrées à une fréquence donnée entre les prises de vue des coins de la pièce. Cette opération a pour but d'obtenir du recouvrement entre les différentes images acquises, et ainsi de pouvoir optimiser les positions relatives des caméras.

Pour le cas de prises de vue de chaque coin par l'utilisateur, le mouvement au moment de l'exposition est négligeable et la modélisation de la caméra par un modèle OG est une hypothèse valable. En revanche, pour le cas de l'enregistrement d'images à l'insu de l'utilisateur déplaçant la caméra, la modélisation fine du capteur est nécessaire. Les modèles de caméra sont généralement trop coûteux pour une adaptation temps réel sur téléphones et tablettes. En revanche, il est possible de réaliser un ajustement de faisceaux global OD a posteriori et de corriger ainsi les déviations induites par la modélisation OG. Cependant, une partie de la dérive de l'estimation des poses caméras est due au manque d'information texturale dans les images pour lesquels les algorithmes

de SLAM par points échouent. Nous proposons d'utiliser des primitives segments, mieux répartis dans les images, afin de pouvoir optimiser la position relative des caméras.

5.5.2 Augmentation du nombre de segments

La méthode de reconstruction, présentée précédemment, utilise uniquement les segments représentant la projection des intersections entre les plans de la pièce. Nous proposons d'utiliser l'ensemble des segments détectés par LSD [von Gioi 2010] dans les images.



FIGURE 5.25 – La reprojection du modèle de pièce (gauche) est utilisée afin de classifier les segments 2D en fonction de leur appartenance aux différents plans (droite).

L'estimation initiale de la pièce fournie par notre méthode permet la projection de l'ensemble de ces segments sur les plans du modèle. Le modèle est reconstruit à partir des images des 4 coins de la pièce. Le modèle étant optimisé pour correspondre au mieux aux observations de ces images, il est logique de projeter les segments de ces images sur le modèle. Cependant, il est nécessaire de trouver sur quel plan projeter chaque segment 2D. Nous proposons de projeter les murs du modèle dans l'image, et de classifier les segments 2D selon leur inclusion dans le polygone défini par la projection des murs dans l'image.

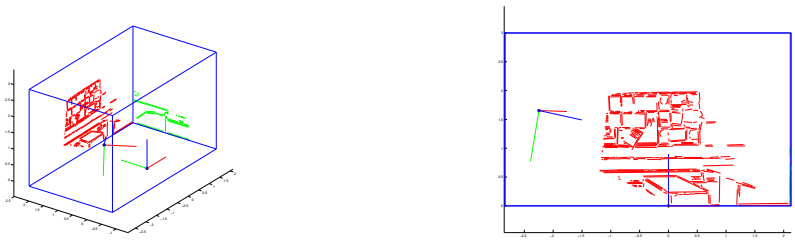


FIGURE 5.26 – Différentes vues 3D de la projection des segments de la première image de coin sur le modèle estimé d'une pièce parallélépipédique.

La figure 5.25 illustre la projection du modèle dans l'image et la segmentation des segments 2D selon leur appartenance au mur de gauche (rouge), au mur de droite (vert), ou aucun des deux (cyan). Un segment 2D n'appartient à aucun plans si ces extrémités sont situées sur 2 plans différents. Les segments sont ensuite projetés sur leur plan associé après segmentation. La figure 5.26 montre la projection des segments de la première image de coin sur le modèle estimé 3D de la pièce.



FIGURE 5.27 – Différentes vues 3D de la projection des segments des 4 images de coins sur le modèle estimé.

L'opération est répétée pour chaque image de coin, qui permet d'obtenir des estimées 3D de segments répartis sur l'ensemble des murs de la pièce (voir fig 5.27). Bien que certains segments détectés dans l'image n'appartiennent pas au plan du modèle, leur projection sur ce modèle permet d'obtenir une estimée initiale de la profondeur maximale de ces segments. Ils peuvent être filtrés ultérieurement, par exemple en observant si leur projection entre deux images n'obéit pas à une homographie.

5.5.3 Appariement de segments 3D-2D

Les segments doivent être appariés afin de pouvoir fournir des observations pour le processus d'optimisation. Deux méthodes sont possibles pour l'appariement de segments :

— L'appariement par descripteur binaire [Zhang 2013].
— Le suivi de segments [Comport 2006]

Les méthodes de suivi sont généralement plus rapides que les méthodes d'appariement mais très sensibles au mouvement pixellique entre les images. En temps normal, ces méthodes ne seraient pas conseillées pour une fréquence d'image aussi faible que celle choisie. Nous disposons cependant d'une estimée initiale 3D des segments, ce qui permet de projeter ces segments dans les images intermédiaires et d'obtenir une estimée 2D relativement fiable du segment (voir fig. 5.28). Les segments peuvent ensuite être appariés en utilisant la méthode décrite dans [Comport 2006].

Une fois un ensemble de segments appariés entre les images disponibles, il est possible de réaliser une optimisation globale des caméras, de la structure

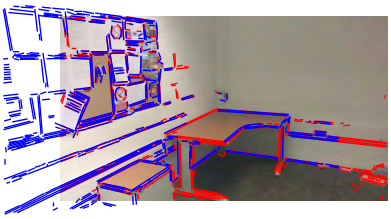


FIGURE 5.28 – Reprojection des segments reconstruits (bleu) dans une image aux côtés des segments 2D détectés dans cette image (rouge).

et des segments. Les segments ne sont en revanche pas entièrement libre dans l'espace, car supposés sur les plans du modèle. Nous proposons donc deux paramétrisations différentes pour ces segments, une paramétrisation dépendante du modèle, et une paramétrisation libre pour les segments filtrés qui n'appartiennent pas au modèle.

Le segment appartenant au modèle observé dans les images ne décrit pas l'observation d'un segment, mais une observation particulière du modèle. Le segment n'a donc pas d'existence propre et ne correspond qu'à une observation du modèle. Ainsi, dans la représentation par graphe, l'observation d'un segment se traduit par une multi-arête reliant le modèle à deux caméras ; la première étant celle de sa première observation, la deuxième étant celle de son observation courante. Un segment contraint donc le modèle, la pose caméra de sa première observation et la pose caméra de son observation courante (voir fig. 5.29).

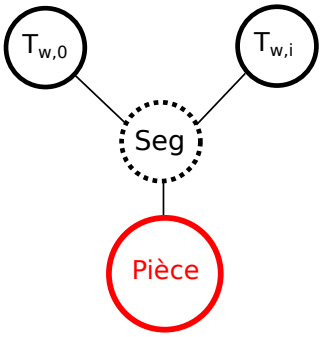


FIGURE 5.29 – L'observation d'un segment dans une image traduit l'observation du modèle par une caméra ayant pour transformation associée $\mathbf{T}_{w,i}$. Le segment étant paramétré par sa première observation, la transformation $\mathbf{T}_{w,0}$ est également contrainte.

5.5.4 Apprentissage automatique

De nombreux travaux récents s'appuyant sur l'apprentissage automatique, et notamment l'apprentissage profond, ont montré des résultats impressionnants pour la reconstruction en environnement intérieur. C'est notamment le cas des approches proposées pour l'estimation de carte de profondeur à partir d'une image monoculaire [Eigen 2014, Liu 2018] (utilisée dans CNN-SLAM [Tateno 2017]), ou pour l'estimation directe de la structure de la disposition de la pièce dans l'image [Lee 2017]. L'utilisation de ces méthodes pour obtenir les observations dans l'image pourrait offrir de meilleurs résultats pour notre application.

5.6 Conclusion

Ce chapitre illustre la difficulté de l'application de la reconstruction dans les environnements d'intérieur en contexte monoculaire multi-vues. Les méthodes adaptées à ces environnements reposent généralement sur l'hypothèse d'un monde Manhattan. Nous proposons une approche par optimisation, utilisant plusieurs modèles basés sur cette hypothèse, permettant de décrire de manière minimale une pièce pour faciliter une estimation de la structure par optimisation.

En s'appuyant sur des travaux utilisant des méthodes de reconstruction adaptées aux contraintes d'intérieur, nous avons proposé une approche de détection automatique des observations de ce modèle dans les images. Bien que ces méthodes offrent des résultats satisfaisants dans de nombreuses configurations, d'autres configurations communes de pièces telles que les cuisines font généralement échouer ces méthodes. La principale raison est que la structure de la pièce est tout simplement invisible dans les images, car elle est occultée par le mobilier. L'intervention d'un utilisateur pour donner ou confirmer les observations est généralement nécessaire.

Dans le contexte de reconstruction de la structure de la pièce en utilisant un appareil mobile tenu par un utilisateur, nous proposons une méthode simple d'acquisition d'un jeu de données minimal (1 image par coin de la pièce). Ce procédé permet d'obtenir aisément une estimée initiale fiable de la structure de la pièce. Cependant, les poses de la caméra fournies par une méthode SLAM à court terme (odométrie visuelle) intégrée sur la dispositif (typiquement ARkit ou ARcore) dérivent au cours du temps, particulièrement dans les environnements intérieurs, et notre procédé ne permet pas d'optimiser conjointement les poses de la caméra et le modèle.

Pour résoudre ce problème, un enrichissement des données en entrée est proposé, utilisant plus d'images intermédiaires entre les images de coins, et s'appuyant sur les segments extraits des images afin de pouvoir optimiser

conjointement les poses et le modèle.

Chapitre 6

Applications

Sommaire

6.1 Scénario monoculaire 133
6.1.1 Scénario feuille A4 134
6.1.2 Scénario image et centrale inertielle 135
6.2 Réalité augmentée sur plateforme Android 136
6.2.1 Performances 137
6.2.2 Réalité augmentée 138
6.3 Réalité diminuée et réalité altérée 139
6.3.1 Acquisition des images depuis différents points de vue 141
6.3.2 Acquisition de données haut niveau fournies par l'utilisateur . 142
6.3.3 Reconstruction du modèle de scène 143
6.3.4 Effacement 144
6.3.5 Limitations 144
6.4 Conclusion 145

Les chapitres précédents s'attellent à résoudre les problématiques liées au contexte dans lequel s'effectuent ces travaux, le développement d'un service de réalité augmentée sur tablette ou téléphone portable, pour assister un usager dans l'aménagement d'un environnement intérieur. Nous avons décrit nos contributions notamment sur la localisation d'une caméra avec gestion de l'obturateur déroulant aux chapitres 3 et 4, puis sur la modélisation d'environnements d'intérieur au chapitre 5. Ce chapitre a pour but de présenter les applications s'appuyant sur ces contributions.

6.1 Scénario monoculaire

Avant le commencement de nos travaux, Innersense s'appuyait sur une solution du marché fournie par VUFORIA[LLC 2015] utilisant un marqueur au sol pour l'initialisation du SLAM ; ce marqueur permettait de localiser la caméra F relativement à ce marqueur dans la scène, mais aussi d'obtenir l'échelle absolue de cette scène. Innersense a exploité cette solution dans ses premières réalisations en réalité augmentée, donc incruster dans la vidéo un objet virtuel là où a été positionné le marqueur dans la scène.

Cette solution, à base de marqueur de type QR-code, présentait plusieurs inconvénients (en particulier à partir du moment où VUFORIA est devenu une composante de QUALCOMM). Un premier besoin fort dans l'ameublement non supporté par cette solution, est de pouvoir essayer le meuble sur une simple photo, et non sur une vidéo. Ainsi, dans un premier temps, une application de localisation de la caméra depuis une unique image a été étudiée. Deux scenarii ont été proposés afin de répondre à cette demande :

— Un scénario de localisation par rapport à un marqueur connu placé dans l'environnement et visible dans l'image
— Un scénario de localisation par rapport au sol en utilisant des informations inertielles et un a priori sur la hauteur de la caméra.

6.1.1 Scénario feuille A4

Le premier scénario consiste à estimer la position relative de la caméra par rapport à un marqueur connu placé dans la scène. Ce marqueur est dans notre cas une feuille A4, choisie pour sa disponibilité chez l'ensemble des utilisateurs. Ainsi, l'utilisateur est invité à placer une feuille à la position où il souhaite visualiser son meuble, puis prendre une photo contenant ce marqueur.

Une première étape consiste à estimer où se trouve la feuille dans l'image. Différentes approches peuvent être utilisées, notamment des méthodes basées apprentissage. Cependant, bien que robustes, ces méthodes échouent régulièrement. Nous choisissons de demander à l'utilisateur de pointer la feuille sur l'image. La feuille peut ensuite être recherchée de manière plus robuste dans la région pointée par l'utilisateur, notamment en utilisant une segmentation image. Les coins du trapèze sont ensuite extraits, et l'association 2D-3D est réalisée connaissant les dimensions de la feuille A4.

A partir de ces associations, une homographie est estimée et les paramètres extrinsèques de la caméra en sont extraits [Hartley 2004]. Les paramètres intrinsèques de la caméra sont considérés connus et fournis directement par Android ou Ios. Ces paramètres extrinsèques et intrinsèques peuvent ensuite être utilisés afin de surimposer le modèle 3D d'un objet virtuel dans l'image (voir fig 6.1).

Cependant, l'association 2D-3D n'est pas toujours valide dans certaines configurations d'observation où la longueur de la feuille à l'image apparaît plus petite que sa largeur. Cette application est rendue plus robuste en testant plusieurs configurations et en comparant l'axe vertical obtenu depuis l'homographie avec l'axe vertical donné par la détection de points de fuite ou le capteur inertiel.

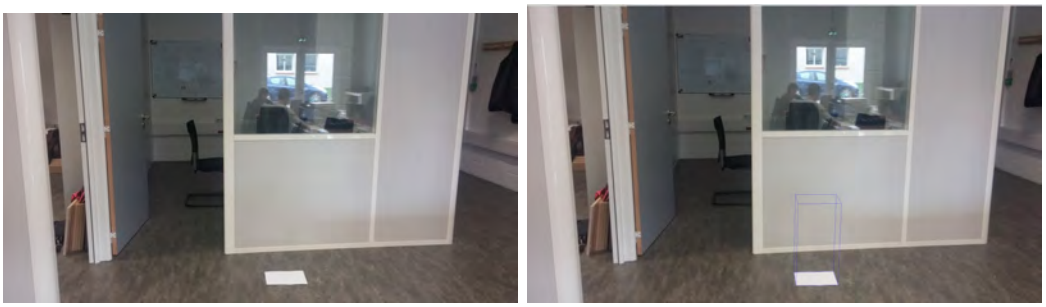


FIGURE 6.1 – Illustration de la méthode utilisant une feuille A4. La feuille détectée dans l'image d'origine (gauche) permet d'estimer la position relative de la caméra à l'échelle absolue et d'ainsi faire apparaître un objet 3D de manière réaliste dans l'image (droite), ici le modèle fil de fer d'une boîte de $50cm$ de hauteur.

6.1.2 Scénario image et centrale inertielle

Le second scénario suppose de ne pas utiliser un marqueur pour la localisation relative de la caméra par rapport au sol. Dans l'absolu, il est impossible de localiser la caméra à l'échelle à partir d'une seule image. En revanche, notre contexte permet certaines hypothèses a priori pour estimer grossièrement cette échelle. Le fait que la photo soit prise par un utilisateur contraint directement la hauteur de la caméra par rapport au sol. Dans notre cas, nous supposons la hauteur de la caméra lors de la prise de photos à $1m50$.

Cette notion seule ne suffit pas à obtenir les paramètres extrinsèques de la caméra. En revanche, en utilisant les capteurs inertiels, et plus particulièrement le vecteur gravité, il est possible d'obtenir une orientation relative du plan du sol et de la caméra. En effet, le vecteur gravité normalisé correspond à la normale au plan du sol. La normale et la distance au sol suffisent à définir le plan du sol dans le repère caméra.

Suffisamment d'informations sont alors disponibles pour pouvoir chercher l'intersection entre un rayon optique et le plan du sol. Cette intersection spécifie alors le centre du repère monde. La normale au plan du sol définit l'axe **z** du repère monde. L'axe **x** de ce repère monde s'obtient en projetant l'axe **x** de la caméra sur le plan du sol, et l'axe **y** en projetant l'axe **z** de la caméra sur ce plan. Ces axes du repère monde exprimés dans le repère caméra peuvent être mis en matrice afin d'obtenir la transformation monde vers caméra $\mathbf{T}_{c,w}$.

Le rayon optique considéré est généralement créé à partir du pixel au centre de l'image. Un élément virtuel peut alors être projeté dans l'image de manière réaliste (voir fig 6.2 et 6.3) : la transformation $\mathbf{T}_{c,w}$ sert à projeter les éléments de l'objet virtuel dans l'image, donc en positionnant le repère de cet objet sur le repère monde.

Nous avons utilisé les données inertielles afin d'obtenir le vecteur gravité et donc l'orientation relative de la caméra au sol. Il est tout à fait envisageable



FIGURE 6.2 – Exemple de résultats obtenus par notre méthode inertielle pour la projection d'une feuille A4 virtuelle. Ici, la feuille A4 blanche au centre ne sert que de comparaison.



FIGURE 6.3 – Projection d'un parallélépipède de $50 \times 30 \times 50$ dans l'image. Les dimensions correspondent à la desserte à droite de l'image.

de se passer de l'utilisation de la centrale inertielle et d'utiliser les points de fuite extraits de l'image afin d'obtenir cette orientation relative, d'une manière similaire à celle décrite dans [Lee 2009]. Des tests ont été menés dans ce sens et apportent des résultats convenables, mais la méthode reste très sensible à l'observation de suffisamment de lignes de fuite dans l'image.

Ces deux applications, très simples à mettre en place, reposent sur des principes connus dans la communauté vision, et apportent des résultats intéressants pour des applications de réalité augmentée. De nos jours, des procédés utilisant des approches par apprentissage automatique permettent l'estimation précise de cartes de profondeur à partir d'une seule image RGB [Eigen 2014, Liu 2018]. Cette information de profondeur pourrait permettre alors très simplement l'incrustation d'objets virtuels dans l'image de manière réaliste.

6.2 Réalité augmentée sur plateforme Android

La technologie de réalité augmentée d'Innersense reposait sur une méthode basée marqueur. Un premier besoin exprimé consistait à se passer du marqueur

en utilisant un SLAM monoculaire. Dans un premier temps, l'outil logiciel open source ORBSLAM a été choisi pour les raisons évoquées précédemment. Cette solution SLAM, développée en $C++$ et adaptée à un fonctionnement sur ordinateur sous Linux Ubuntu, doit être portée afin de fonctionner sur Android. Nous avons donc intégré nativement ORBSLAM en utilisant JNI (Java Native Interface) qui permet l'intégration de code natif dans une application Android. Le front-end de l'application développé en Java réalise alors un minimum d'opérations, à savoir la capture d'images et les transmissions de données avec le code natif.

6.2.1 Performances

L'adaptation brute du code d'ORBSLAM sous Android se heurte à des performances trop faibles en l'état pour une utilisation convenable. Les architectures des téléphones et des tablettes sont encore inférieures en capacité de calcul face aux ordinateurs sur lesquels ORBSLAM fonctionne en temps réel. En effet, même pour des appareils mobiles haut-de-gamme (tablette Samsung pro S2), la vitesse de traitement de l'algorithme parvient difficilement à se hisser au delà de $7fps$. Cette faible fréquence de traitement des images perturbe la robustesse du suivi de points et rend l'expérience incertaine.

En vue d'augmenter la fréquence, il est essentiel de se concentrer sur l'amélioration des parties de l'algorithme les plus coûteuses en temps de calcul. En étudiant les temps d'exécution des différentes étapes de l'algorithme à l'aide d'un outil de profilage, il est possible de dégager les opérations nécessitant une optimisation de celles dont le temps de calcul est négligeable.

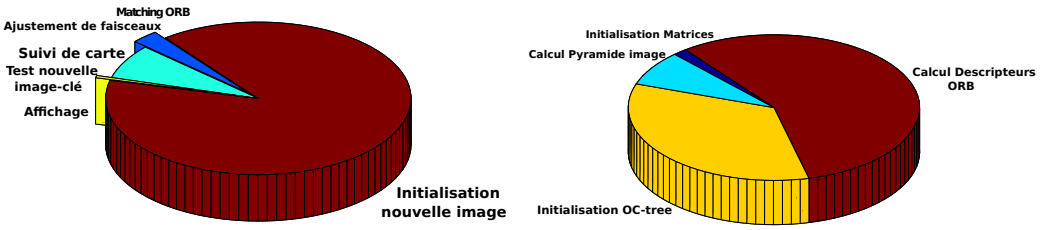


FIGURE 6.4 – Illustration du temps d'exécution de diverses étapes d'ORBSLAM sur tablette Android. A gauche, l'ensemble des étapes réalisées pour chaque image, et à droite, les détails de l'initialisation d'une nouvelle image (détails dans le texte).

La figure 6.4 illustre les temps requis pour différentes étapes réalisées par le SLAM. Bien qu'ORBSLAM utilise 4 processus en parallèle, nous nous concentrons sur celui dédié au suivi entre les images. Dans un premier temps, 6 étapes du suivi entre images sont étudiées (image de gauche) :

— La création d'une "frame", qui correspond notamment à l'initialisation de la matrice image en niveau de gris, la détection de points d'intérêts et l'extraction de descripteurs ORB.
— La mise en correspondance des points d'intérêts (Matching ORB).
— Une étape d'un ajustement de faisceaux $motion - only$ qui consiste à optimiser la position relative des 2 dernières caméras et d'un nuage de points local à partir des appariements.
— Le suivi de la carte locale qui comprend la reprojection de la carte locale dans l'image, un appariement et une nouvelle étape d'optimisation.
— Le test pour savoir si une nouvelle image-clé (KF pour KeyFrame) doit être insérée.
— L'affichage des résultats du SLAM sur l'image.

La première étape prend le plus de temps dans le suivi image($\approx 130ms$). La figure de droite détaille le temps d'exécution de 4 différentes étapes de cette initialisation des images :

— La mise en place des matrices images (redimensionnement, conversion niveau de gris, clonage matrice).
— Le calcul de la pyramide d'images nécessaire pour la robustesse à l'échelle de l'appariement et du suivi.
— La mise en place de l'octree (pour l'accélération de la recherche d'appariements entre les 8 niveaux de la pyramide d'images).
— Le calcul des descripteurs ORB.

Les trois dernières étapes sont les plus coûteuses, pour lesquels de nombreuses optimisations peuvent être réalisées afin de réduire le temps de calcul.

Dans un premier temps, OpenMP (Open Multi-Processing), interface de programmation pour le calcul parallèle, est utilisé afin de paralléliser les tâches qui peuvent l'être. C'est le cas pour le calcul de la pyramide d'images et de certaines étapes de la création de l'octree. Pour le calcul des descripteurs, les instructions ARM-NEON sont utilisées. Elles fournissent une accélération des calculs vectoriels très présents dans le calcul de l'orientation du descripteur.

Les optimisations proposées permettent de réduire le temps de calcul d'un facteur oscillant entre $\times 2$ et $\times 3$. Ces gains en performances permettent le fonctionnement du SLAM en temps réel ($20fps$)sur les tablettes haut-de-gamme (Samsung S2 pro), et un fonctionnement à temps fonctionnel ($\approx 12 - 15fps$) sur des tablettes bas de gamme (fréquence suffisante pour que l'algorithme fonctionne

6.2.2 Réalité augmentée

La solution libre ORBSLAM fournit une reconstruction partielle de l'environnement, sous la forme d'un ensemble épars d'amers ponctuels, ainsi qu'une localisation de la caméra dans l'environnement. Ajouter un élément virtuel devient alors trivial.

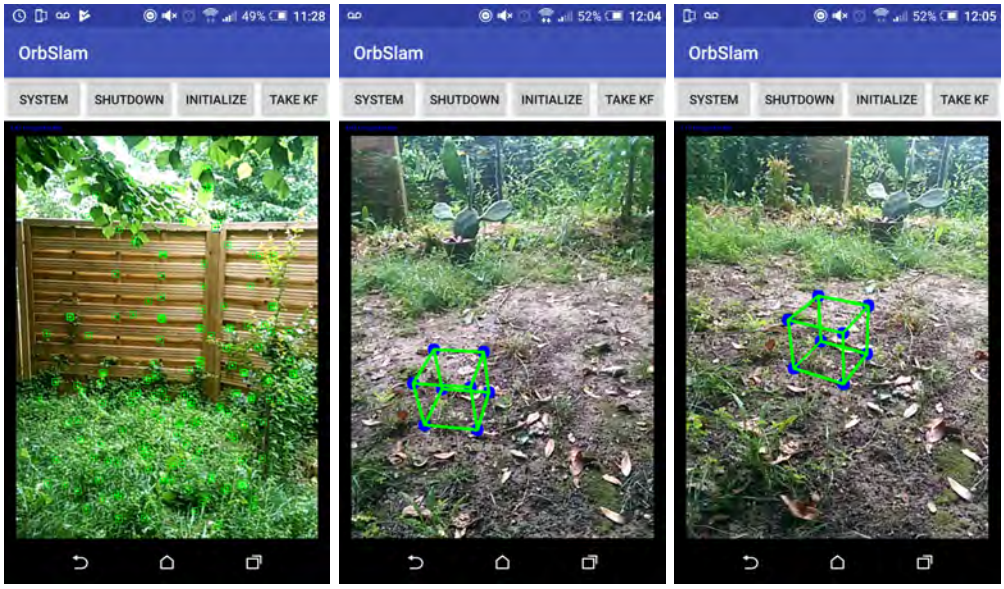


FIGURE 6.5 – Illustration de l'application de réalité augmentée intégrant ORBSLAM. Les résultats affichés sont réalisés sur un téléphone portable datant de 2014.

Nous choisissons de faire intervenir l'utilisateur afin d'initialiser l'incrustation de l'objet 3D dans l'image. L'utilisateur est ainsi invité à cliquer sur l'image à l'endroit où il souhaite voir apparaître l'élément virtuel. Un point d'intérêt est alors recherché dans la zone, et un cube virtuel est affiché au niveau du point 3D associé à ce point d'intérêt. Le plan du sol peut éventuellement être recherché par RANSAC [Fischler 1987] en utilisant le voisinage de ce point 3D, afin d'apposer correctement le cube 3D à la surface du sol. Le fonctionnement et l'interface de l'application sont illustrés figure 6.5.

Bien qu'ORBSLAM ne fournisse pas une échelle absolue de la scène en l'état, il est possible d'utiliser les méthodes décrites dans la section précédente afin d'obtenir une estimée initiale de l'échelle.

6.3 Réalité diminuée et réalité altérée

Rappelons que la réalité diminuée consiste à effacer un élément réel d'une image, et que la réalité altérée est la combinaison des réalités diminuée et augmentée, où des éléments réels sont effacés, et des objets virtuels rajoutés à la scène. Ce scénario est particulièrement intéressant dans le cas de l'ameublement virtuel, et constitue l'une des applications recherchées par les méthodes développées dans nos travaux. Nous avons intégré une première solution de réalité altérée, dans le cadre d'un projet collaboratif avec Innersense et l'équipe REVA (Réel Expression Vie Artificielle) de l'IRIT.

La réalité diminuée s'appuie sur des méthodes de complétion d'image, ou d'effacement ($in-painting$), afin de supprimer un élément réel d'une image.



FIGURE 6.6 – Exemple d'effacement sur une image initiale (droite) à partir des méthodes d'*in painting* basées diffusion (milieu) et basées patch (droite).

Cette méthode nécessite la connaissance d'une région d'image délimitant la zone de l'image qui doit être effacée. Les algorithmes d'effacement utilisent les informations image avoisinant (ou non) la région effacée afin de compléter le vide artificiel. Deux différentes méthodes sont généralement utilisées pour la complétion ; les approches par diffusion, et les approches basées patch d'image (voir 6.8).

L'un des problèmes principaux de ces méthodes d'effacement, est qu'elles ne tiennent pas compte de la géométrie 3D de l'environnement pour compléter une image. Ainsi, dans le cas des méthodes basées patch, des patchs d'image seront cherchés dans l'ensemble de l'image afin de compléter un vide. Pour un environnement d'intérieur, si la zone à compléter est située sur un seul plan de la pièce, par exemple le sol, on ne souhaite pas obtenir des patchs d'image issus des murs afin de compléter le vide. Ces méthodes ont donc intérêt à prendre en compte la géométrie de la pièce dans la recherche des patchs d'image. On parle alors d'effacement 3D afin de différencier de l'effacement 2D.

Nos méthodes de reconstruction de la structure de la pièce permettent d'obtenir les informations nécessaires pour que les méthodes de complétion utilisent des patchs d'image dans les zones planes supposées (voir fig. 6.6). L'apport de l'information 3D permet également d'obtenir un critère de confiance associé aux pixels, ce qui permet d'influencer le choix des patchs d'image sélectionnés. Ce critère suppose que les pixels correspondant à la projection de points 3D situés loin de la caméra sont trop bruités et entraîneraient une propagation de ce bruit dans la zone à compléter [Fayer 2017]. La figure 6.8 illustre l'effacement sans critère de confiance (milieu), qui offre des résultats bruités laissant deviner la zone complétée, au contraire de la méthode utilisant les zones de confiance pour la sélection de patch (droite).

L'ensemble des technologies sont donc disponibles pour la réalisation d'une application de réalité diminuée ou altérée. Nous proposons une architecture simple pour parvenir à nos besoins, qui se divise en 4 étapes principales :

— Prise de points de vue par l'utilisateur
— Ajout de données par l'utilisateur
— Reconstruction 3D de la structure de la pièce



FIGURE 6.7 – Illustration des méthodes d'effacement 2D (milieu) et celles prenant en compte l'information 3D (droite). L'image d'origine est affichée à gauche.



FIGURE 6.8 – Exemple d'effacement sans (gauche) et avec (droite) utilisation d'un critère de confiance. L'image de départ est affichée à gauche.

— Effacement image

6.3.1 Acquisition des images depuis différents points de vue

L'utilisateur est invité à initialiser un SLAM puis à prendre une photo de chaque coin de la pièce, et éventuellement des photos de zones de la pièce qu'il souhaiterait voir effacées. Une application interne a été développée pour faciliter la prise de données ; elle permet la sauvegarde des images, des poses caméras et de la distance focale. L'application utilise la méthode d'odométrie visuelle ARKit, une technologie récente qui fournit une estimée précise de la position de la caméra montée sur le dispositif mobile, à une échelle absolue. De plus, cet outil intègre une détection du plan du sol pour l'initialisation, ce qui permet d'obtenir un repère monde correctement orienté. La figure 6.9 illustre différentes prises de vue effectuées par l'utilisateur.



FIGURE 6.9 – Exemple de prises de vue acquises par l'utilisateur.

6.3.2 Acquisition de données haut niveau fournies par l'utilisateur

De nombreuses données en entrée doivent être fournies aux algorithmes de reconstruction et d'effacement afin de pouvoir fonctionner. Une partie de ces informations peut être obtenue automatiquement, notamment les observations de la structure du modèle comme détaillé dans le chapitre précédent. Cependant, afin d'assurer la robustesse de la suite du processus, c'est l'utilisateur qui fournit en entrée ces données pour cette application. L'utilisateur est également invité à fournir les masques d'effacement indiquant les zones à effacer, et éventuellement les lumières afin de pouvoir gérer fidèlement la luminosité de l'environnement. De plus, il est intéressant de demander à l'utilisateur de pointer une zone de texture des différents plans ; c'est utile dans le cas où ces plans ne sont pas unis. Cette dernière indication permet de segmenter la texture afin que l'effacement utilise uniquement des patchs issus de la texture du mur et non d'éléments perturbateurs présents sur le mur (poster, horloge, etc).

La figure 6.10 illustre les données fournies par l'utilisateur sur les prises de vues montrées en figure 6.10. Un masque d'effacement et l'image d'origine associée sont montrés dans les images du milieu et de gauche. Dans l'image de droite, les informations de textures sont indiquées par un point de couleur rouge, vert ou bleu selon l'appartenance au sol, murs ou plafond. La lumière est précisée par une icône de soleil et les observations des arêtes du modèle sont indiquées par des segments rouge, vert et bleu.

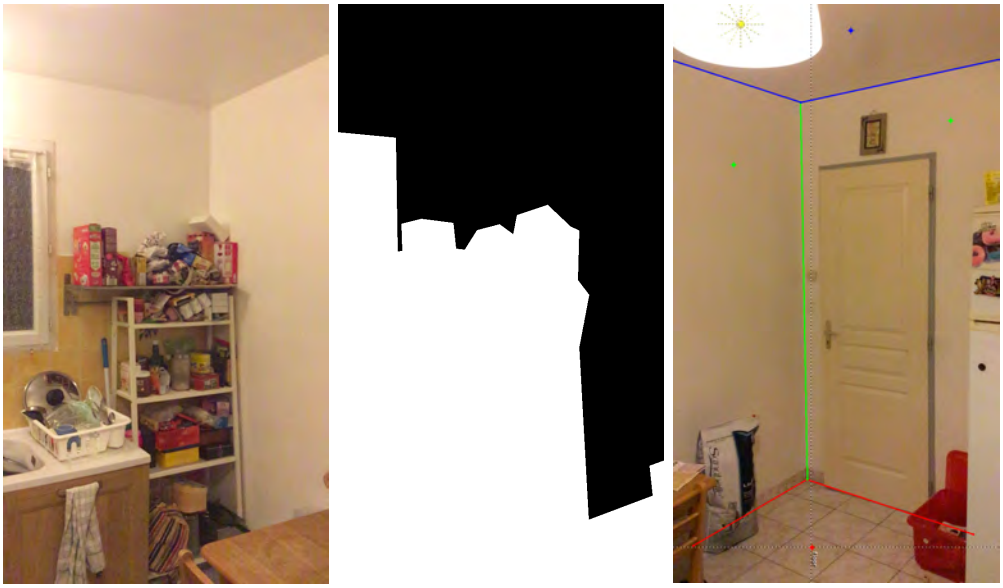


FIGURE 6.10 – Exemple des informations fournies par l'utilisateur, un masque d'effacement (milieu) et son image originale (gauche), ainsi que les informations de textures et d'intersections (droite).

6.3.3 Reconstruction du modèle de scène

A partir des informations renseignées par l'utilisateur, le modèle 3D de la pièce est reconstruit en utilisant les méthodes de reconstruction décrites dans le chapitre précédent. La figure 6.11 montre les résultats obtenus sur un jeu de données d'une cuisine.



FIGURE 6.11 – Reconstruction d'une cuisine à partir des informations fournies par l'utilisateur. En vert la reconstruction initiale, et en bleu la reconstruction après optimisation.

La reconstruction fournit un modèle de la structure de la pièce suffisamment fidèle pour que l'aspect qualitatif de la suite du processus ne soit pas dégradé. De plus, cette reconstruction est très rapide grâce au faible nombre de paramètres à optimiser. Cependant, les données en entrée sont limitées aux points de vue sur les coins de la pièce, l'ajout d'un nombre supplémentaire

d'images en entrée n'étant pas encore implémenté.

6.3.4 Effacement



FIGURE 6.12 – Illustration des images avant et après effacement en utilisant notre application.

En utilisant les informations fournies par l'utilisateur et la reconstruction 3D, il est désormais possible de réaliser un effacement des zones masquées sur les images, cela en tenant compte de la structure de la pièce pour utiliser une approche d'effacement 3D. Chaque prise de vue peut être segmentée selon les composants de la pièce (sol, murs et plafond).

Ainsi, dans chaque image, l'effacement est réalisé indépendamment pour chacun des plans de la pièce. L'image est alors segmentée, puis rectifiée (rectification perspective), et les algorithmes d'effacement sont finalement appliqués sur cette image rectifiée. Les résultats sont visibles dans la figure 6.12, où un effacement cohérent des images avec la structure de la pièce est observé.

6.3.5 Limitations

L'application exposée ici souffre de certaines limitations, la première étant principalement due à la-non optimisation des poses de la caméra dans la reconstruction de la structure de la pièce ; la méthode est donc très dépendante de la qualité de l'approche de localisation sous-jacente (ici ARKit). Ce problème est déjà évoqué au chapitre précédent, et un début de solution y est déjà apporté, notamment en augmentant le nombre d'images en entrée.

La figure 6.13 illustre un cas où une pose de la caméra est mal estimée. Ce cas est fréquent sur les dernières prises de vue car l'incertitude sur la position de la caméra augmente continuellement dans les méthodes d'odométrie visuelle mise en œuvre dans ARKit.



FIGURE 6.13 – Illustration d'un cas où l'erreur d'estimation de la pose de caméra entraîne une estimation de la structure de la pièce qui n'est pas en accord avec l'image d'origine, d'où une arête discontinu entre les deux murs.

6.4 Conclusion

Dans ce chapitre, diverses applications mises en œuvre pour l'ameublement d'intérieur ont été montrées, en utilisant les travaux des chapitres précédents. Ces applications répondent à différents besoins exprimés par Innersense selon les contraintes imposées.

Nous avons ainsi proposé deux solutions permettant l'ajout d'un élément virtuel dans une seule image à partir d'une feuille A4 rajoutée dans la scène, ou en utilisant soit des données inertielles soit la recherche des points de fuite dans l'image.

Une adaptation d'un algorithme de SLAM de l'état de l'art sur Android a également été implémentée, avec une optimisation des performances afin de pouvoir offrir une expérience confortable à l'utilisateur. Cependant, les solutions constructeur récemment arrivées sur le marché (ARKit, ARCore) offrent de meilleures performances, à moindre coût et avec estimation de l'échelle absolue. Ces performances ne peuvent être atteintes sans avoir accès aux données constructeur ; c'est particulièrement le cas pour les données inertielles qui sont horodatées par le système d'exploitation. Mais comme ce système n'est pas temps réel, cela empêche toute utilisation fine de ces données pour un développeur qui n'a pas accès aux données internes de Android ou Ios.

Grâce à la collaboration avec les collègues de l'IRIT, spécialistes dans les approches d'$in - painting$, nous proposons finalement une méthode de réalité diminuée, qui peut être exploitée ensuite pour la réalité altérée, donc pour rajouter un élément virtuel sur l'image diminuée ; cette partie est triviale connaissant la pose de la caméra ainsi qu'un modèle géométrique de l'environnement. Cette méthode offre des résultats qualitativement satisfaisants, bien

que les problèmes issus de la reconstruction évoqués précédemment soient encore présents. De plus, l'utilisateur est invité à renseigner un trop grand nombre de données, ce qui peut être fastidieux : l'intégration des méthodes de détection automatique évoquées au chapitre précédent s'avérerait bénéfique. L'application remplit tout de même l'objectif d'obtenir une visualisation d'une pièce vide de manière très rapide en limitant le coût calculatoire.

La plupart de ces applications sont encore au stade expérimental, et ne sont pas encore utilisées en production. De nombreuses améliorations et nouvelles fonctionnalités sont possibles et intégrables à court terme.

Conclusion

Ce manuscrit de thèse rapporte l'ensemble des travaux effectués afin de répondre aux problématiques de localisation et de reconstruction à partir d'images acquises par une tablette ou un téléphone portable déplacé par un usager dans un environnement intérieur ; ces dispositifs sont équipés avec des capteurs bas coût, en particulier des caméras à obturateur déroulant (OD).

Nous avons mené nos travaux entre le LAAS-CNRS et l'entreprise Innersense ; les besoins formulés par Innersense s'inscrivent dans un contexte d'essayage virtuel de meubles ou d'objets de décoration, pour des utilisateurs finaux. Initialement, Innersense utilisait à cet effet VUFORIA, une solution de réalité augmentée par suivi de marqueur disponible sur le marché. Cette technologie, malgré sa fiabilité et les avantages apportés par l'utilisation d'un marqueur, n'offrait pas de reconstruction de l'environnement et l'exploitation d'un marqueur est trop contraignante pour l'utilisateur.

Le principal objectif de cette thèse était d'améliorer ou de remplacer cette technologie, afin de pouvoir être capable de reconstruire la structure de la pièce en vue de vider virtuellement cette pièce. En effet, un des besoins principaux dans l'essayage de meubles par réalité augmentée est non seulement de pouvoir visualiser le meuble en situation et en temps réel, mais également de pouvoir placer ce meuble virtuel où l'utilisateur souhaiterait qu'il soit dans la pièce, et ce de manière physiquement cohérente. Pour ce faire, et en excluant la possibilité de demander à l'utilisateur de vider réellement sa pièce, il est nécessaire d'utiliser des méthodes de complétion d'image [Barnes 2009]. Cependant, ces méthodes offrent de mauvais résultats quand on les applique dans des environnements d'intérieur multi-planaires. La géométrie de l'environnement doit donc être prise en compte à la manière des travaux présentés dans [Kawai 2016], afin de pouvoir vider virtuellement une pièce (réalité diminuée) pour ensuite rajouter des éléments virtuels (réalité altérée).

Le but de nos travaux consistait donc à contribuer au passage de solutions de réalité augmentée ne tenant pas compte de l'environnement vers des solutions de réalité altérée en tenant compte du contexte.

Contributions

Afin de répondre aux différentes problématiques impliquées dans la réalité altérée en utilisant les composants équipant les téléphones et tablettes mobiles, nous avons choisi de traiter les problèmes successivement. Ainsi dans un premier temps, nous nous sommes concentrés sur l'influence des capteurs

disponibles sur les appareils mobiles, et en particulier sur l'impact qu'a l'exploitation d'une caméra OD sur les algorithmes de Vision, et ensuite sur le cas de la reconstruction dans un environnement d'intérieur.

La première partie des travaux de cette thèse est donc centrée sur l'étude et l'influence de la caméra OD présente sur les tablettes et téléphones mobiles du marché. En effet le fonctionnement de ces caméras étant différent de celui des caméras à obturateur global (OG), le modèle sténopé généralement utilisé dans les algorithmes de vision par ordinateur n'est plus adapté, notamment dans le cas de mouvements rapides de la caméra. Nous avons donc étudié l'impact de l'utilisation de caméra OD avec un modèle OG sur l'estimation de la localisation par SLAM visuel. Des jeux de données synthétiques ont été générés dans un premier temps afin de mettre en évidence le problème d'une modélisation non adaptée de la caméra sur le SLAM. Les expériences sur ces jeux de données ont montré une dérive de l'estimation de la trajectoire de la caméra lors de mouvements rapides en rotation.

Un jeu de données réelles a été acquis ensuite afin de confirmer ces résultats. Pour ce faire, nous avons mis en œuvre un dispositif équipé d'une caméra OD, d'une caméra OG et de balises pour acquérir la vérité terrain par un système de capture de mouvements. Diverses séquences d'images ont été acquises par les deux caméras montées sur ce dispositif déplacé à la main avec des mouvements plus ou moins rapides en translation et en rotation. Ces deux séquences d'images ont ensuite été traitées indépendamment par une solution Open source de SLAM visuel afin de pouvoir comparer les estimations des trajectoires suivies par les caméras, toutes les deux représentées par le modèle sténopé OG. Les trajectoires obtenues avec les différentes images et avec la capture de mouvement sont exprimées chacune dans des repères différents ; deux méthodes de recalage ont été développées, dont une nouvelle méthode basée sur l'interpolation qui permet d'obtenir des correspondances entre les poses des caméras indépendamment du fait que les données soit acquises à des instants différents.

Ces expériences ont permis de mettre en lumière les cas où une modélisation spécifique de la caméra OD est nécessaire, particulièrement pour des rotations rapides, car une dérive de l'estimation, voire une perte du suivi des amers dans les séquences acquises par la caméra OD sont généralement constatées dans ces cas. Nous avons également montré que la perte de suivi peut s'avérer utile car cela empêche la corruption de la carte lors des mouvements à forte dynamique. Cet aspect bénéfique est en revanche valable uniquement si la scène est ensuite ré-observée et la solution SLAM exploite des approches de détection de boucles et de relocalisation efficientes.

Ces expériences ont donc motivé nos travaux concernant la modélisation fine de la caméra OD pour les algorithmes de vision. En nous

inspirant des précédentes approches proposées pour la modélisation de caméra OD [Meingast 2005, Forssén 2010, Hedborg 2012, Oth 2013, Steven Lovegrove 2013], une modélisation de la trajectoire caméra en temps continu a été proposée, puis développée. Ce modèle utilise des poses de contrôle (PC) réparties uniformément dans le temps pour interpoler de manière continue la trajectoire par B-Splines cumulatives cubiques. Une amélioration de ce modèle a été proposée afin de relâcher la contrainte de répartition temporelle uniforme des PC, en utilisant les B-Spline non-uniformes. Deux méthodes de détermination de la distribution temporelle non uniforme des PC ont été développées, permettant l'ajout de PC où elles sont nécessaires le long de la trajectoire. Une première méthode se fonde sur des données inertielles acquises en même temps que les images sur tablettes ou téléphones portables ; elle permet d'ajouter une PC quand la dynamique de la trajectoire est trop élevée. La seconde méthode utilise une analyse temporelle des résidus, afin d'ajouter une PC entre le couple de PC pour lequel l'erreur est la plus élevée. L'efficacité de ces méthodes a été démontrée sur des données synthétiques. Le modèle d'interpolation par B-Splines non-uniformes a été intégré pour la résolution des problèmes du PnP et de l'ajustement de faisceaux ; ces algorithmes ont été testés sur des données simulées afin de démontrer leur bon comportement sur des données bruitées.

Un processus d'optimisation spatio-temporelle a également été présenté ; ce processus suppose l'intégration des horodatages des PC dans le processus d'optimisation. Diverses démarches sont possibles, soit en optimisant conjointement les PC et leurs horodatages, soit en optimisant les PC temporellement et spatialement de manière alternée. La démarche alternée offre de meilleurs résultats car elle permet d'éviter que l'optimisation converge dans des minima locaux de la fonction de coût, créés par l'optimisation conjointe des paramètres temporels et spatiaux.

Enfin, l'ajout d'une pose de contrôle par la méthode d'analyse des résidus a également été intégré dans la démarche d'optimisation alternée. Des tests sur des données réelles ont été conduits ; ils montrent l'apport de cette optimisation, qui permet de diminuer l'erreur après la convergence d'une optimisation spatiale seule. Le but de cette répartition non uniforme des PC est de permettre de diminuer le nombre de PC nécessaires pour l'interpolation d'une trajectoire, en adaptant leur distribution à la dynamique de la trajectoire. Une optimisation en utilisant cette distribution est ainsi moins coûteuse en temps de calcul et en mémoire ; ces gains dépendent de la dynamique de la trajectoire et de la précision souhaitée. Cet apport est important car le coût calculatoire d'un ajustement de faisceaux intégrant un modèle de caméra OD est très élevé, notamment à cause du calcul des jacobiennes des résidus en

fonction des paramètres des PC. Les jacobiennes analytiques associées sont également développées en annexe de ce document mais n'ont pas encore été intégrées dans un processus SLAM complet temps-réel. L'ensemble de ces travaux permet d'obtenir une modélisation plus fine d'une caméra OD pour l'estimation de trajectoire, avec une optimisation du coût calculatoire et de la répartition des PC a posteriori dans un traitement par lot hors ligne.

Indépendamment de nos travaux sur la modélisation fine d'une caméra OD, cette thèse inclue également des contributions sur la gestion de la contrainte d'environnement d'intérieur impliquée par notre contexte. L'échec des méthodes de reconstruction classiques sur ces milieux est démontré sur des données réelles.

Dans un premier temps, différents modèles géométriques d'environnement d'intérieur ont été proposés. Deux modèles principaux sont ainsi décrits, un modèle respectant l'hypothèse d'un monde Manhattan, et un modèle plus libre autorisant des murs non orthogonaux entre eux. La description de ces représentations inclue également différents modèles d'observation qui exploitent les coins de la pièce ou les lignes d'intersection entre les plans de la pièce. Des méthodes de détection automatique de ces observations ont été développées en se basant sur des travaux précédents [Lee 2009, Flint 2010a]. L'approche proposée se base sur notre implémentation de la détection des points de fuite à partir des lignes extraites dans une image et du calcul d'une carte d'orientation des pixels [Lee 2009]. Une méthode de balayage d'images selon les lignes de fuite a été exposée. Cette approche associe un score à chaque intersection créée par ce balayage, score calculé en fonction de la cohérence avec la carte d'orientation et avec les segments d'image détectés. Les résultats de cette méthode appliquée sur des images réelles sont illustrés et montrent le bon fonctionnement pour des cas simples. Pour améliorer la robustesse, l'utilisateur est invité à confirmer les intersections détectées et à les fournir le cas échéant. L'optimisation des modèles est ensuite présentée en utilisant ces lignes d'intersection comme observation du modèle.

Finalement, un processus d'acquisition complet est exposé, qui invite l'utilisateur à réaliser une photographie de chaque coin de la pièce, la caméra étant suivie par une solution SLAM : les observations sont ensuite extraites automatiquement ou fournies par l'utilisateur sur chacune de ces photos. Le modèle de structure de la pièce est ensuite initialisé puis optimisé à partir de ces observations. Les résultats sur des données réelles et en comparaison à des vérités terrain montrent des résultats quantitativement et qualitativement satisfaisants. Nous avons aussi obtenu des résultats préliminaires où nous cherchons à améliorer le processus en augmentant le nombre de données en entrées afin d'optimiser conjointement les poses de la caméra et le modèle.

Enfin nous présentons des applications pour l'aménagement d'intérieur dé-

veloppées au cours de cette thèse, intégrant en partie nos travaux. Ainsi, des méthodes de réalité augmentée sur une image seule à partir d'a priori ont été développées. Une solution Open source de SLAM visuel a été intégrée sous Android en optimisant les performances afin de gérer les contraintes liées aux faibles capacités de calcul disponibles sur les dispositifs mobiles. Dans le cadre d'un projet collaboratif avec l'IRIT, une application complète de réalité altérée a été développée, permettant de vider virtuellement une pièce réelle à l'aide de la connaissance de l'environnement ; cela permet d'incruster par la suite des éléments virtuels de manière cohérente, sans interférer avec les meubles existants.

Limitations

Des limitations sont encore présentes dans les différentes méthodes et applications développées.

Premièrement, le modèle d'une caméra OD par interpolation, uniforme ou non, est très coûteux ; cela empêche un traitement temps réel sur un ordinateur standard, et a fortiori sur un dispositif mobile. Les implémentations développées dans cette thèse se limitent à un traitement par lot sur des séquences enregistrées au préalable. Si les performances ne permettent pas un traitement temps réel, le coût calculatoire reste cependant abordable, de l'ordre d'une dizaine de secondes pour l'ajustement de faisceaux sur les exemples sur des données de synthèse (60 PC, 400 points) présentés en Chapitre 4.

Le procédé d'optimisation spatio-temporelle alternée, bien que permettant de diminuer l'erreur globale en comparaison des méthodes classiques d'optimisation spatiale seulement, est relativement lent car il suppose de réaliser des optimisations successives jusqu'à convergence.

Sur nos contributions sur la reconstruction d'environnement d'intérieur, la méthode de détection automatique des intersections dans l'image reste très sensible aux erreurs de l'estimation de la carte d'orientation et donc de la détection des points de fuite pour le calcul du score. Bien que de la même façon que[Flint 2011], notre méthode puisse utiliser d'autres cartes de segmentation des pixels pour le calcul du score [Hoiem 2005], la robustesse de la méthode repose toujours sur l'estimée initiale de l'orientation des pixels. Ces méthodes fonctionnent bien pour des cas relativement simples où les plans principaux des pièces sont bien observables et se projettent dans de larges zones de pixels dans l'image. Mais souvent dans notre contexte, et particulièrement dans le cas des cuisines, ces méthodes échouent à correctement estimer la disposition de la pièce dans l'image. C'est pourquoi l'utilisateur est encore invité à confirmer ou renseigner ces informations dans l'application proposée en réalité altérée ; cette interaction est fastidieuse pour l'utilisateur.

Notre méthode de reconstruction utilise uniquement une image par coin

de la pièce, avec la pose de la caméra associée. Les images ne disposant pas de recouvrement des observations entre elles, il est impossible d'optimiser conjointement ces poses caméra et la structure de la pièce. Ainsi, la méthode est entièrement dépendante de la précision des poses de la caméra fournies en entrée par une solution SLAM. Bien que nous ayons proposé une variante pour enrichir les données en entrées afin d'obtenir du recouvrement entre les images, et que nous ayons montré des résultats préliminaires sur cette variante, l'optimisation conjointe du modèle de la pièce et des poses de la caméra n'est pas encore fonctionnelle.

Perspectives

Les travaux développés dans cette thèse ouvrent la voix à la réalité altérée sur des appareils mobiles. Bien que les applications développées soient encore au stade expérimental, elles permettent déjà d'obtenir des résultats satisfaisants. Le comportement de ces méthodes a été exposé dans des environnements d'intérieur personnels, mais elles ont été et peuvent être appliquées à des environnements plus grands tels que des hangars. La taille de l'environnement n'influe pas sur les résultats de nos méthodes, à l'exception du cas où l'utilisateur doit éventuellement parcourir de longues distances pour l'acquisition des coins de la structure, augmentant le côté fastidieux de l'acquisition ainsi que la probabilité de dérive du SLAM.

En sortant du cadre initial de l'aménagement d'intérieur, les méthodes développées dans cette thèse peuvent être utilisées pour des cas plus larges, notamment pour l'architecture et la construction, afin d'obtenir un rendu rapide et de pouvoir visualiser à l'avance de possibles modifications (effacement de cloison, d'étage, de mezzanine, etc). La représentation par graphe de l'optimisation de la pièce permet de pouvoir changer simplement de modèle d'environnement, de capteur et d'observations. Des modèles plus simples ou plus complexes pourraient ainsi être intégrés sans perturber les processus d'effacement intervenant par la suite.

La méthode de recalage de trajectoire par interpolation développée dans cette thèse offre des résultats similaires aux méthodes classiques. Ceci est particulièrement dû au fait que les données que nous utilisons ont des fréquences d'acquisition relativement proches et que le gain apporté en précision est négligeable pour nos applications. Il semble évident que notre approche de recalage sera plus utile dans de nombreux cas d'applications où les fréquences des différents capteurs estimant des trajectoires sont très disparates.

En particulier nous n'avons pas eu connaissance de méthodes similaires

exploitées dans la communauté robotique sur des robots équipés en vision monoculaire.

Les travaux réalisés sur la modélisation de la caméra OD à partir de B-Splines non uniformes ne sont pas seulement adaptés pour les caméra OD. En effet, la modélisation en temps continu de la trajectoire d'un robot apporte certains avantages (prédiction vitesse et accélération) ainsi que certains inconvénients (coût calculatoire). La plupart des solutions de l'état de l'art utilise une distribution temporelle uniforme (DTU) des PC, entraînant des problèmes évoqués de sous-échantillonnage ou de sur-échantillonnage. Notre distribution temporelle non uniforme (DTNU) des PC permettrait de diminuer drastiquement le coût mémoire et calculatoire de ces méthodes, particulièrement pour les cas de robots effectuant des trajectoires ayant de longues parties pouvant être modélisées par peu de points de contrôle ; c'est particulièrement le cas des drones ou des automobiles.

Travaux futurs

Une des améliorations futures envisagées serait d'intégrer notre modèle de caméra OD dans un processus complet de SLAM. De plus, le modèle de B-Splines cubiques non uniformes conserve la continuité C^2 de la trajectoire interpolée, ce qui permet l'interpolation des vitesses et des accélérations. Les données de vitesse angulaire et d'accélération fournies par une centrale inertielle pourraient être intégrées tel que décrit dans [Steven Lovegrove 2013] ; leur intégration dans l'optimisation permettrait notamment l'estimation de l'échelle absolue de la scène. Notons que cette intégration suppose un horodatage très précis des données qui n'est pour le moment pas disponible sur les appareils mobiles.

Nous avons montré que le suivi sur les images acquises par des caméras OD ou OG ont des résultats comparables à l'exception des mouvements à forte dynamique en rotation, qui entraînent généralement une perte de suivi. Notre modèle permet d'éviter la corruption de la carte ; il est envisageable d'enregistrer des images durant la perte du suivi, et d'utiliser notre processus d'optimisation a posteriori afin de modéliser correctement la trajectoire durant cette perte de suivi.

Un des axes principaux d'amélioration concernant la reconstruction d'intérieur, concerne la détection automatique des observations du modèle dans les images. Les méthodes reposant sur l'apprentissage automatique offrent aujourd'hui des résultats inégalés pour l'estimation de la disposition de la pièce dans une image. L'intégration de ces méthodes pourraient être un axe d'amélioration important pour nos méthode de reconstruction, en particulier les

méthodes d'estimation des cartes de profondeur à partir d'images RGB.

Annexe A

Annexe

A.1 Dérivées analytiques

L'utilisation de dérivées calculées par différences finies peut poser problème pour deux principales raisons :

— Ce processus est coûteux, car il est nécessaire d'évaluer deux fois la fonction de coût (différence finie centrée) pour chacun des paramètres à optimiser.

— Le pas qui correspond à la valeur de la perturbation sur ces paramètres est généralement le même quelque soit le paramètre ($\varepsilon = 1e^{-9}$). Cette perturbation infinitésimale, si jamais élevée au carré, n'est plus pertinente et s'apparente à du bruit numérique pour un encodage de la valeur en flottant ou en simple précision.

Nous proposons dans cette section de donner les dérivées analytiques associées aux problèmes d'optimisation adaptés à des caméras OD.

A.1.0.1 Jacobienne de l'interpolation de pose

La jacobienne de la l'interpolation représente la dérivée de la pose $\mathbf{T}_w(t)$ en fonction de perturbations de dimension 6 paramétrées sur l'algèbre de Lie $\varepsilon_{i=0..3} \in \mathbb{R}^6$ (voir chapitre 2) sur chacune des PC $T_{w,i}$ considérée. La pose interpolée étant dépendante des composantes spatiales de 4 PC $\mathbf{T}_{w,i=0..3} \in \mathbb{R}^{3\times4}$, la jacobienne est donc une matrice $(3 \times 4) \times (6 \times 4) = 12 \times 24$. En adaptant la formulation donnée dans le contenu additionnel de [Kerl 2015] pour notre formulation non-uniforme, on obtient :

$$\mathbf{T}_w(t) = exp(\varepsilon_0)\mathbf{T}_{w,0}\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3 \tag{A.1}$$

avec

$$\mathbf{A}_i = exp(\tilde{B}_i(t)log((exp(\varepsilon_{i-1})\mathbf{T}_{w,i-1})^{-1}exp(\varepsilon_i)\mathbf{T}_{w,i}) \tag{A.2}$$

Chaque perturbation ε_i apparaît dans 2 facteurs $\mathbf{A}_i$ excepté la perturbation ε_3 sur $\mathbf{T}_{w,3}$. Les jacobiennes pour chaque perturbations s'écrivent alors :

$$\frac{\delta\mathbf{T}_w(t)}{\delta\varepsilon_0}\Big|_{\varepsilon_0=0} = \frac{\delta(exp(\varepsilon_0)\mathbf{T}_{w,0}\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3)}{\delta(\varepsilon_0)} + \frac{\delta(\mathbf{T}_{w,0}\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3)}{\delta(\mathbf{A}_1)}\frac{\delta\mathbf{A}_1}{\delta\varepsilon_0} \tag{A.3}$$

$$\frac{\delta \mathbf{T}_w(t)}{\delta \varepsilon_1}\Big|_{\varepsilon_1=0} = \frac{\delta(\mathbf{T}_{w,0}\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3)}{\delta(\mathbf{A}_1)}\frac{\delta \mathbf{A}_1}{\delta \varepsilon_1} + \frac{\delta(\mathbf{T}_{w,0}\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3)}{\delta(\mathbf{A}_2)}\frac{\delta \mathbf{A}_2}{\delta \varepsilon_1} \tag{A.4}$$

$$\frac{\delta \mathbf{T}_w(t)}{\delta \varepsilon_2}\Big|_{\varepsilon_2=0} = \frac{\delta(\mathbf{T}_{w,0}\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3)}{\delta(\mathbf{A}_2)}\frac{\delta \mathbf{A}_2}{\delta \varepsilon_2} + \frac{\delta(\mathbf{T}_{w,0}\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3)}{\delta(\mathbf{A}_3)}\frac{\delta \mathbf{A}_3}{\delta \varepsilon_2} \tag{A.5}$$

$$\frac{\delta \mathbf{T}_w(t)}{\delta \varepsilon_3}\Big|_{\varepsilon_3=0} = \frac{\delta(\mathbf{T}_{w,0}\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3)}{\delta(\mathbf{A}_3)}\frac{\delta \mathbf{A}_3}{\delta \varepsilon_3} \tag{A.6}$$

La difficulté principale réside dans l'obtention des dérivées $\frac{\delta \mathbf{A}_i}{\delta \varepsilon_i}$ et $\frac{\delta \mathbf{A}_{i+1}}{\delta \varepsilon_i}$. Les dérivées d'un produit de matrices par rapport à une matrice peuvent être obtenues à partir des formules données dans [Blanco 2014]. La difficulté consiste à évaluer les jacobiennes $\frac{\delta \mathbf{A}_i}{\delta \varepsilon_i}$ car elles comprennent les jacobiennes de l'application exponentielle et logarithmique évaluées pour un ε différent de l'identité. Ainsi, si on développe la jacobienne par la loi de composition des dérivées, on obtient :

$$\frac{\delta \mathbf{A}_i}{\delta \varepsilon_i}\Big|_{\varepsilon_i=0} = \frac{\delta exp(\mathbf{E})}{\delta \mathbf{E}}\Big|_{\mathbf{E}=\tilde{B}_i(u) log(\mathbf{T}_{w,i-1}\mathbf{T}_{w,i})} \tag{A.7}$$

$$\tilde{B}_i(u)\frac{\delta log(\mathbf{T}_{w,i-1} D \mathbf{T}_{w,i}}{\delta D}\Big|_{D=exp(\varepsilon_i=0)} \tag{A.8}$$

$$\frac{\delta exp(\varepsilon_i)}{\delta \varepsilon_i}\Big|_{\varepsilon_i=0} \tag{A.9}$$

La jacobienne de l'application exponentielle évaluée à l'identité est donnée dans le chapitre 2. En revanche, les deux autres jacobiennes de l'application logarithmique et exponentielle sont beaucoup plus complexes à obtenir.

A.1.1 Approximation sur SE(3)

Dans un premier temps, nous avons cherché à obtenir les jacobiennes "générales" de l'application exponentielle à l'aide d'un outil de calcul symbolique à la manière de [Kerl 2015]. La formulation non compacte de l'application exponentielle comprend des formules trigonométriques coûteuses à évaluer, et l'outil de calcul fournit une formule très lourde et coûteuse à évaluer numériquement en l'état. Il est donc nécessaire de rechercher une formulation plus élégante de ces jacobiennes.

Des formulations pour l'approximation sur $\mathbb{SE}(3)$ et son algèbre de Lie associé $\mathfrak{se}(3)$ sont données dans [Chirikjian 2012], [Forster 2015]. Pour un mouvement infinitésimal ε_i et étant donné les générateurs de $\mathbb{SE}(3)$ $\mathbf{G}_i$, la linéari-

sation suivante est donnée :

$$exp(\varepsilon \mathbf{G}i) \approx \mathbf{I}_4 + \varepsilon \mathbf{G}i \tag{A.10}$$

Une formule d'approximation sur $\mathbb{SO}(3)$ est détaillée dans [Forster 2015], qui peut être étendue sur $\mathbb{SE}(3)$. Étant donné une variation de pose $[\mathbf{\Omega}]_\vee = [\mathbf{w}, \mathbf{a}]$ sur l'algèbre de Lie $\mathfrak{se}(3)$ et un mouvement infinitésimale ε on a :

$$exp(\mathbf{\Omega} + \varepsilon) \approx \exp(\mathbf{\Omega}) \exp(Jr(\mathbf{\Omega})\varepsilon) \tag{A.11}$$
$$exp(\mathbf{\Omega} + \varepsilon) \approx \exp(Jl(\mathbf{\Omega})\varepsilon) \exp(\mathbf{\Omega})) \tag{A.12}$$

$Jr(\Omega)$ et $Jl(\Omega)$ respectivement les Jacobiennes à droite et à gauche de $\mathbb{SE}(3)$. Le calcul de $Jr(\Omega)$ est abordé dans [Chirikjian 2012] en utilisant $\hat{Jr}(\omega)$ la jacobienne à droite de $\mathbb{SO}(3)$. Une formule finie est donnée pour la jacobienne à gauche dans [Eade 2017] :

$$\mathbf{Jl}(\mathbf{\Omega}) = \begin{pmatrix} \hat{\mathbf{Jl}}(\omega) & \mathbb{O}_3 \\ b_\theta [\mathbf{a}]_\times + c_\theta([\omega]_\times [\mathbf{a}]_\times + [\mathbf{a}]_\times [\omega]_\times) + (\omega^T \mathbf{a})\mathbf{Q}(\omega) & \hat{\mathbf{Jl}}(\omega) \end{pmatrix} \tag{A.13}$$

$$\theta = \|\omega\| \tag{A.14}$$

$$a_\theta = \frac{\sin\theta}{\theta} \tag{A.15}$$

$$b_\theta = \frac{1 - \cos\theta}{\theta^2} \tag{A.16}$$

$$c_\theta = \frac{1 - a_\theta}{\theta^2} \tag{A.17}$$

$$\mathbf{Q}(\omega) = \left(\frac{a_\theta - 2b_\theta}{\theta^2}\right)[\omega]_\times + \left(\frac{b_\theta - 3c_\theta}{\theta^2}\right)[\omega]_\times^2 \tag{A.18}$$

Nous avons testé cette formulation par rapport à la formule obtenue à partir d'un outil de calcul symbolique et obtenu des résultats similaires. La jacobienne de l'application logarithmique est également détaillée dans [Eade 2017]. Notons cependant que cette jacobienne n'a besoin d'être évaluée qu'une fois par intervalle de pose Tw_i, Tw_{i+1} au cours d'une itération de l'optimisation, et peut donc éventuellement être calculée par la méthode des dérivées numériques.

A.1.2 Composition des Jacobiennes

Les jacobiennes de l'interpolation par rapport aux perturbations de poses données dans la section précédente doivent être composées avec d'autres jacobiennes afin d'obtenir la jacobienne de la fonction de coût par rapport aux différents ε_i.

Soit $\mathbf{K}$ la matrice des paramètres intrinsèques et $\pi(.)$ la projection perspective de $\mathbb{P}2$ vers $\mathbb{R}2$. Étant donné les observations $\mathbf{p}_{t,k}$ associées à un point 3D $\mathbf{P}_k = [xyz]^T$ au temps t, la fonction de l'erreur de coût est définie par :

$$\mathbf{Err}(t,k) = \mathbf{p}_{t,k} - \pi([\mathbf{K}|0]\mathbf{T}_{w,c}(t)^{-1}\mathbf{P_k}) \tag{A.19}$$

La jacobienne de l'erreur de reprojection est donnée par :

$$\frac{\delta\mathbf{Err}(t,k)}{\delta\varepsilon_i} = -\frac{\delta\pi([\mathbf{K}|0]\mathbf{T}_w(t)^{-1}\mathbf{P_k})}{\delta\varepsilon_i} \tag{A.20}$$

Définissons un ensemble de variables utiles pour la calcul de la jacobienne :

$$\begin{aligned}
\mathbf{E} &= \mathbf{T}_w(t) \in \mathbb{R}^{3\times4} \\
\mathbf{D} &= (E)^{-1} \in \mathbb{R}^{3\times4} \\
\mathbf{C} &= D\mathbf{P_k} \in \mathbb{R}^{3\times1} \\
\mathbf{B} &= [\mathbf{K}|0]C \in \mathbb{R}^{3\times1} \\
\mathbf{A} &= \pi(B) \in \mathbb{R}^{2\times1}
\end{aligned} \tag{A.21}$$

La jacobienne est alors calculée par la loi de composition des dérivées :

$$\frac{\delta Err(t,k)}{\delta\varepsilon_i} = -\frac{\delta\mathbf{A}}{\delta\mathbf{B}}\frac{\delta\mathbf{B}}{\delta\mathbf{C}}\frac{\delta\mathbf{C}}{\delta\mathbf{D}}\frac{\delta\mathbf{D}}{\delta\mathbf{E}}\frac{\delta\mathbf{E}}{\delta\varepsilon_i}, \in \mathbb{R}^{2\times6} \tag{A.22}$$

Nous ne fournissons par les détails des autres jacobiennes étant donné qu'elles sont relativement simples et peuvent être aisément trouvées dans la littérature. Nous fournissons cependant les dimensions de chacune d'entre elles afin de montrer la cohérence de la loi de composition :

$$\begin{aligned}
\frac{\delta\mathbf{A}}{\delta\mathbf{B}} &\in \mathbb{R}^{2\times3}, \frac{\delta\mathbf{B}}{\delta\mathbf{C}} \in \mathbb{R}^{3\times3}, \frac{\delta\mathbf{C}}{\delta\mathbf{D}} \in \mathbb{R}^{3\times12} \\
\frac{\delta\mathbf{D}}{\delta\mathbf{E}} &\in \mathbb{R}^{12\times12}, \frac{\delta\mathbf{E}}{\delta\varepsilon_i} \in \mathbb{R}^{12\times6}
\end{aligned} \tag{A.23}$$

Le calcul analytique des jacobiennes détaillées ici est adapté à l'optimisation spatiale. Afin de pouvoir optimiser la position temporelle des PC, les dérivées de la fonction de coût doivent être calculées en fonction des horodatages des PC.

A.1.2.1 Jacobiennes par rapport aux horodatages des PC

Pour les B-Splines uniformes, la pose interpolée en t dépend uniquement de l'horodatage des deux points de contrôle temporellement voisin de t. Cependant, pour le cas non uniforme, 6 horodatages des PC environnantes $u_{1...6}$ interviennent dans le calcul de $\mathbf{T}_w(t)$ pour $t \in [t_{i+3}t_{i+4}]$. Sachant que la dérivée des sommes est la somme des dérivées, on peut calculer la dérivée de la matrice des coefficients $\mathbf{M}$ de l'équation 4.13 pour les $u_i, i \neq (3|4)$. Les jacobiennes seront organisées ici en matrice par soucis de cohérence avec la matrice M.

Dérivée par rapport à u_1 :

$$\frac{\delta \mathbf{M}}{\delta_{u_1}} = \begin{pmatrix} \frac{\delta M_{00}}{\delta_{u_1}} & \frac{\delta M_{01}}{\delta_{u_1}} & \frac{\delta M_{02}}{\delta_{u_1}} & \frac{\delta M_{03}}{\delta_{u_1}} \\ \frac{\delta M_{10}}{\delta_{u_1}} & \frac{\delta M_{11}}{\delta_{u_1}} & \frac{\delta M_{12}}{\delta_{u_1}} & \frac{\delta M_{13}}{\delta_{u_1}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{A.24}$$

Dérivée par rapport à u_2 :

$$\frac{\delta \mathbf{M}}{\delta_{u_2}} = \begin{pmatrix} \frac{\delta M_{00}}{\delta_{u_2}} & \frac{\delta M_{01}}{\delta_{u_2}} & \frac{\delta M_{02}}{\delta_{u_2}} & \frac{\delta M_{03}}{\delta_{u_2}} \\ \frac{\delta M_{10}}{\delta_{u_2}} & \frac{\delta M_{11}}{\delta_{u_2}} & \frac{\delta M_{12}}{\delta_{u_2}} & \frac{\delta M_{13}}{\delta_{u_2}} \\ \frac{\delta M_{20}}{\delta_{u_2}} & \frac{\delta M_{21}}{\delta_{u_2}} & \frac{\delta M_{22}}{\delta_{u_2}} & \frac{\delta M_{23}}{\delta_{u_2}} \\ \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{A.25}$$

Pour le cas de la dérivée par rapport aux horodatages u_3 et u_4, il est nécessaire de prendre en compte l'intervention de ces variables dans $u(t) = \frac{t-u_3}{u_4-u_3}$. Il est donc préférable de donner la dérivée du vecteur de base $\mathbf{B}(t) = M_i[1, u(t), u(t)^1, u(t)^3]^T$ par rapport à u_3 :

$$\frac{\delta \mathbf{B}(t)}{\delta_{u_3}} = \begin{pmatrix} \frac{\delta M_{00}}{\delta_{u_3}} + \frac{\delta (M_{01}u)}{\delta_{u_3}} + \frac{\delta (M_{02}u^2)}{\delta_{u_3}} + \frac{\delta (M_{03}u^3)}{\delta_{u_3}} \\ \frac{\delta M_{10}}{\delta_{u_3}} + \frac{\delta (M_{11}u)}{\delta_{u_3}} + \frac{\delta (M_{12}u^2)}{\delta_{u_3}} + \frac{\delta (M_{13}u^3)}{\delta_{u_3}} \\ \frac{\delta M_{20}}{\delta_{u_3}} + \frac{\delta (M_{21}u)}{\delta_{u_3}} + \frac{\delta (M_{22}u^2)}{\delta_{u_3}} + \frac{\delta (M_{23}u^3)}{\delta_{u_3}} \\ \\ \frac{\delta (M_{33}u^3)}{\delta_{u_3}} \end{pmatrix} \tag{A.26}$$

Et ce même vecteur dérivé par rapport à u_4 :

$$\frac{\delta \mathbf{B}(t)}{\delta_{u_4}} = \begin{pmatrix} \frac{\delta M_{00}}{\delta_{u_4}} + \frac{\delta M_{01}}{\delta_{u_4}} + \frac{\delta M_{02}}{\delta_{u_4}} + \frac{\delta M_{03}}{\delta_{u_4}} \\ \frac{\delta M_{10}}{\delta_{u_4}} + \frac{\delta M_{11}}{\delta_{u_4}} + \frac{\delta M_{12}}{\delta_{u_4}} + \frac{\delta M_{13}}{\delta_{u_4}} \\ \frac{\delta M_{20}}{\delta_{u_4}} + \frac{\delta M_{21}}{\delta_{u_4}} + \frac{\delta M_{22}}{\delta_{u_4}} + \frac{\delta M_{23}}{\delta_{u_4}} \\ \frac{\delta M_{33}}{\delta_{u_4}} \end{pmatrix} \tag{A.27}$$

Dérivée par rapport à u_5 :

$$\frac{\delta \mathbf{M}}{\delta_{u_5}} = \begin{pmatrix} \frac{\delta M_{00}}{\delta_{u_5}} & \frac{\delta M_{01}}{\delta_{u_5}} & \frac{\delta M_{02}}{\delta_{u_5}} & \frac{\delta M_{03}}{\delta_{u_5}} \\ \frac{\delta M_{10}}{\delta_{u_5}} & \frac{\delta M_{11}}{\delta_{u_5}} & \frac{\delta M_{12}}{\delta_{u_5}} & \frac{\delta M_{13}}{\delta_{u_5}} \\ \frac{\delta M_{20}}{\delta_{u_5}} & \frac{\delta M_{21}}{\delta_{u_5}} & \frac{\delta M_{22}}{\delta_{u_5}} & \frac{\delta M_{23}}{\delta_{u_5}} \\ 0 & 0 & 0 & \frac{\delta M_{33}}{\delta_{u_5}} \end{pmatrix}, \frac{\delta \mathbf{M}}{\delta_{u_5}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \frac{\delta M_{10}}{\delta_{u_5}} & \frac{\delta M_{11}}{\delta_{u_5}} & \frac{\delta M_{12}}{\delta_{u_5}} & \frac{\delta M_{13}}{\delta_{u_5}} \\ \frac{\delta M_{20}}{\delta_{u_5}} & \frac{\delta M_{21}}{\delta_{u_5}} & \frac{\delta M_{22}}{\delta_{u_5}} & \frac{\delta M_{23}}{\delta_{u_5}} \\ 0 & 0 & 0 & \frac{\delta M_{33}}{\delta_{u_5}} \end{pmatrix} \tag{A.28}$$

Dérivée par rapport à u_6 :

$$\frac{\delta \mathbf{M}}{\delta_{u_6}} = \begin{pmatrix} \frac{\delta M_{00}}{\delta_{u_6}} & \frac{\delta M_{01}}{\delta_{u_6}} & \frac{\delta M_{02}}{\delta_{u_6}} & \frac{\delta M_{03}}{\delta_{u_6}} \\ \frac{\delta M_{10}}{\delta_{u_6}} & \frac{\delta M_{11}}{\delta_{u_6}} & \frac{\delta M_{12}}{\delta_{u_6}} & \frac{\delta M_{13}}{\delta_{u_6}} \\ \frac{\delta M_{20}}{\delta_{u_6}} & \frac{\delta M_{21}}{\delta_{u_6}} & \frac{\delta M_{22}}{\delta_{u_6}} & \frac{\delta M_{23}}{\delta_{u_6}} \\ 0 & 0 & 0 & \frac{\delta M_{33}}{\delta_{u_6}} \end{pmatrix}, \frac{\delta \mathbf{M}}{\delta_{u_6}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\delta M_{23}}{\delta_{u_6}} \\ 0 & 0 & 0 & \frac{\delta M_{33}}{\delta_{u_6}} \end{pmatrix} \tag{A.29}$$

Les détails de ces dérivées sont données par la suite.

A.1.3 Dérivées pour spline sous formes cumulatives

Comme présenté précédemment, les B-Splines doivent être exprimées sous forme cumulative pour l'interpolation sur des espaces non euclidiens. La matrice des coefficients cumulative s'exprime en utilisant l'équation 4.16. La dérivée s'exprime donc par :

$$\frac{\delta \mathbf{M}}{\delta_{u_i}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{\delta M_{10}}{\delta_{u_i}} + \frac{\delta M_{20}}{\delta_{u_i}} & \frac{\delta M_{11}}{\delta_{u_i}} + \frac{\delta M_{21}}{\delta_{u_i}} & \frac{\delta M_{12}}{\delta_{u_i}} + \frac{\delta M_{22}}{\delta_{u_i}} & \frac{\delta M_{13}}{\delta_{u_i}} + \frac{\delta M_{23}}{\delta_{u_i}} + \frac{\delta M_{33}}{\delta_{u_i}} \\ \frac{\delta M_{20}}{\delta_{u_i}} & \frac{\delta M_{21}}{\delta_{u_i}} & \frac{\delta M_{22}}{\delta_{u_i}} & \frac{\delta M_{23}}{\delta_{u_i}} + \frac{\delta M_{33}}{\delta_{u_i}} \\ 0 & 0 & 0 & \frac{\delta M_{33}}{\delta_{u_i}} \end{pmatrix} \tag{A.30}$$

De même pour les cas de la dérivée par rapport aux horodatages u_3 et u_4, on exprime la dérivée du vecteur de bases cumulatives $\tilde{\mathbf{B}}(t) = \mathbf{C}[1, u(t), u(t)^1, u(t)^3]^T$. Soit B_k l'élément k du vecteur de base, on a alors :

$$\frac{\delta \tilde{\mathbf{B}}}{\delta_{u_i}} = \begin{pmatrix} \frac{\delta B_0}{\delta_{u_i}} + \frac{\delta B_1}{\delta_{u_i}} + \frac{\delta B_2}{\delta_{u_i}} + \frac{\delta B_3}{\delta_{u_i}} \\ \frac{\delta B_1}{\delta_{u_i}} + \frac{\delta B_2}{\delta_{u_i}} + \frac{\delta B_3}{\delta_{u_i}} \\ \frac{\delta B_2}{\delta_{u_i}} + \frac{\delta B_3}{\delta_{u_i}} \\ \frac{\delta B_3}{\delta_{u_i}} \end{pmatrix} \tag{A.31}$$

A.1.4 Détails des jacobiennes

Les détails du calcul de ces jacobiennes sont dérivés par la suite.

Dérivée par rapport à u_1 :

$$\frac{\delta M_i}{\delta_{u_1}} = \begin{pmatrix} \frac{\delta M_{00}}{\delta_{u_1}} & \frac{\delta M_{01}}{\delta_{u_1}} & \frac{\delta M_{02}}{\delta_{u_1}} & \frac{\delta M_{03}}{\delta_{u_1}} \\ \frac{\delta M_{10}}{\delta_{u_1}} & \frac{\delta M_{11}}{\delta_{u_1}} & \frac{\delta M_{12}}{\delta_{u_1}} & \frac{\delta M_{13}}{\delta_{u_1}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{A.32}$$

$$\begin{pmatrix} \frac{\mathrm{u43}^2}{\mathrm{u41}^2\,\mathrm{u42}} & -\frac{3\,\mathrm{u43}^2}{\mathrm{u41}^2\,\mathrm{u42}} & \frac{3\,\mathrm{u43}^2}{\mathrm{u41}^2\,\mathrm{u42}} & -\frac{\mathrm{u43}^2}{\mathrm{u41}^2\,\mathrm{u42}} \\ \frac{\mathrm{u31}\,\mathrm{u43}}{\mathrm{u41}^2\,\mathrm{u42}} - \frac{\mathrm{u43}}{\mathrm{u41}\,\mathrm{u42}} & \frac{3\,\mathrm{u43}^2}{\mathrm{u41}^2\,\mathrm{u42}} & \frac{3\,\mathrm{u43}^2}{\mathrm{u41}^2\,\mathrm{u42}} & \frac{\mathrm{u43}^2}{\mathrm{u41}^2\,\mathrm{u42}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{A.33}$$

Avec :

$$\frac{\delta M_{00}}{\delta_{u_1}} = \frac{u_{43}^2}{u_{42}u_{41}^2}, \frac{\delta M_{01}}{\delta_{u_1}} = -3M_{00}, \frac{\delta M_{02}}{\delta_{u_1}} = 3M_{00}, \frac{\delta M_{03}}{\delta_{u_1}} = -M_{00} \tag{A.34}$$

$$\frac{\delta M_{10}}{\delta_{u_1}} = \frac{u_{31}u_{43}}{u_{42}u_{41}^2} - \frac{u_{43}^2}{u_{42}u_{41}}, \frac{\delta M_{11}}{\delta_{u_1}} = 3M_{00}, \frac{\delta M_{12}}{\delta_{u_1}} = 3M_{00}, \frac{\delta M_{13}}{\delta_{u_1}} = M_{00} \tag{A.35}$$

Dérivée par rapport à u_2 :

$$\frac{\delta M_i}{\delta_{u_2}} = \begin{pmatrix} \frac{\delta M_{00}}{\delta_{u_2}} & \frac{\delta M_{01}}{\delta_{u_2}} & \frac{\delta M_{02}}{\delta_{u_2}} & \frac{\delta M_{03}}{\delta_{u_2}} \\ \frac{\delta M_{10}}{\delta_{u_2}} & \frac{\delta M_{11}}{\delta_{u_2}} & \frac{\delta M_{12}}{\delta_{u_2}} & \frac{\delta M_{13}}{\delta_{u_2}} \\ \frac{\delta M_{20}}{\delta_{u_2}} & \frac{\delta M_{21}}{\delta_{u_2}} & \frac{\delta M_{22}}{\delta_{u_2}} & \frac{\delta M_{23}}{\delta_{u_2}} \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{A.36}$$

Avec :

$$
\begin{aligned}
\frac{\delta M_{00}}{\delta_{u_2}} &= \frac{\mathrm{u43}^2}{\mathrm{u41}\,\mathrm{u42}^2}, \\
\frac{\delta M_{01}}{\delta_{u_2}} &= -\frac{3\,\mathrm{u43}^2}{\mathrm{u41}\,\mathrm{u42}^2}, \\
\frac{\delta M_{02}}{\delta_{u_2}} &= \frac{3\,\mathrm{u43}^2}{\mathrm{u41}\,\mathrm{u42}^2}, \\
\frac{\delta M_{03}}{\delta_{u_2}} &= -\frac{\mathrm{u43}^2}{\mathrm{u41}\,\mathrm{u42}^2} \\
\frac{\delta M_{10}}{\delta_{u_2}} &= \frac{\mathrm{u31}\,\mathrm{u43}}{\mathrm{u41}\,\mathrm{u42}^2} - \frac{\mathrm{u53}}{\mathrm{u42}\,\mathrm{u52}} + \frac{\mathrm{u32}\,\mathrm{u53}}{\mathrm{u42}\,\mathrm{u52}^2} + \frac{\mathrm{u32}\,\mathrm{u53}}{\mathrm{u42}^2\,\mathrm{u52}}, \\
\frac{\delta M_{11}}{\delta_{u_2}} &= \frac{3\,\mathrm{u43}^2}{\mathrm{u41}\,\mathrm{u42}^2} - \frac{3\,\mathrm{u43}}{\mathrm{u42}\,\mathrm{u52}} + \frac{3\,\mathrm{u32}\,\mathrm{u43}}{\mathrm{u42}\,\mathrm{u52}^2} + \frac{3\,\mathrm{u32}\,\mathrm{u43}}{\mathrm{u42}^2\,\mathrm{u52}}, \\
\frac{\delta M_{12}}{\delta_{u_2}} &= \frac{3\,\mathrm{u43}^2}{\mathrm{u41}\,\mathrm{u42}^2} - \frac{3\,\mathrm{u43}^2}{\mathrm{u42}\,\mathrm{u52}^2} - \frac{3\,\mathrm{u43}^2}{\mathrm{u42}^2\,\mathrm{u52}}, \\
\frac{\delta M_{13}}{\delta_{u_2}} &= \mathrm{u43}^2\left(\frac{1}{\mathrm{u41}\,\mathrm{u42}^2} + \frac{1}{\mathrm{u42}\,\mathrm{u52}^2} + \frac{1}{\mathrm{u42}^2\,\mathrm{u52}} + \frac{1}{\mathrm{u52}^2\,\mathrm{u53}}\right) \\
\frac{\delta M_{20}}{\delta_{u_2}} &= \frac{\mathrm{u32}^2}{\mathrm{u42}\,\mathrm{u52}^2} - \frac{2\,\mathrm{u32}}{\mathrm{u42}\,\mathrm{u52}} + \frac{\mathrm{u32}^2}{\mathrm{u42}^2\,\mathrm{u52}}, \\
\frac{\delta M_{21}}{\delta_{u_2}} &= \frac{3\,\mathrm{u32}\,\mathrm{u43}}{\mathrm{u42}\,\mathrm{u52}^2} - \frac{3\,\mathrm{u43}}{\mathrm{u42}\,\mathrm{u52}} + \frac{3\,\mathrm{u32}\,\mathrm{u43}}{\mathrm{u42}^2\,\mathrm{u52}}, \qquad \text{(A.37)} \\
\frac{\delta M_{22}}{\delta_{u_2}} &= \frac{3\,\mathrm{u43}^2}{\mathrm{u42}\,\mathrm{u52}^2} + \frac{3\,\mathrm{u43}^2}{\mathrm{u42}^2\,\mathrm{u52}}, \\
\frac{\delta M_{23}}{\delta_{u_2}} &= -\mathrm{u43}^2\left(\frac{1}{\mathrm{u42}\,\mathrm{u52}^2} + \frac{1}{\mathrm{u42}^2\,\mathrm{u52}} + \frac{1}{\mathrm{u52}^2\,\mathrm{u53}}\right)
\end{aligned}
\qquad \text{(A.38)}
$$

Pour le cas de la dérivée par rapport aux horodatages u_3 et u_4, il est nécessaire de prendre en compte $u^{=}\frac{t-u_3}{u_4-u_3}$ dans le calcul, ainsi ce sont les dérivées du vecteur de base $B(t) = M_i * [1, u, u^1, u^3]^T$ qui sont données. Dérivé par rapport à u_3 :

$$
\frac{\delta B}{\delta_{u_3}} = \begin{pmatrix}
\frac{\delta M_{00}}{\delta_{u_3}} + \frac{\delta (M_{01} u)}{\delta_{u_3}} + \frac{\delta (M_{02} u^2)}{\delta_{u_3}} + \frac{\delta (M_{03} u^3)}{\delta_{u_3}} \\
\frac{\delta M_{10}}{\delta_{u_3}} + \frac{\delta (M_{11} u)}{\delta_{u_3}} + \frac{\delta (M_{12} u^2)}{\delta_{u_3}} + \frac{\delta (M_{13} u^3)}{\delta_{u_3}} \\
\frac{\delta M_{20}}{\delta_{u_3}} + \frac{\delta (M_{21} u)}{\delta_{u_3}} + \frac{\delta (M_{22} u^2)}{\delta_{u_3}} + \frac{\delta (M_{23} u^3)}{\delta_{u_3}} \\
\\
\frac{\delta (M_{33} u^3)}{\delta_{u_3}}
\end{pmatrix} \qquad \text{(A.39)}
$$

Avec :

$$\begin{aligned}
\frac{\delta M_{00}}{\delta_{u_3}} &= -\frac{2\,\mathrm{u43}}{\mathrm{u41}\,\mathrm{u42}},\\
\frac{\delta(M_{01}u)}{\delta_{u_3}} &= \frac{3\,\mathrm{u43}}{\mathrm{u41}\,\mathrm{u42}} + \frac{3\,(t-\mathrm{u3})}{\mathrm{u41}\,\mathrm{u42}},\\
\frac{\delta(M_{02}u^2)}{\delta_{u_3}} &= -\frac{3\,(2\,t-2\,\mathrm{u3})}{\mathrm{u41}\,\mathrm{u42}},\\
\frac{\delta(M_{03}u^3)}{\delta_{u_3}} &= \frac{3\,(t-\mathrm{u3})^2}{\mathrm{u41}\,\mathrm{u42}\,\mathrm{u43}} - \frac{(t-\mathrm{u3})^3}{\mathrm{u41}\,\mathrm{u42}\,\mathrm{u43}^2}\\
\frac{\delta M_{10}}{\delta_{u_3}} &= \frac{\mathrm{u43}}{\mathrm{u41}\,\mathrm{u42}} - \frac{\mathrm{u32}}{\mathrm{u42}\,\mathrm{u52}} - \frac{\mathrm{u31}}{\mathrm{u41}\,\mathrm{u42}} + \frac{\mathrm{u53}}{\mathrm{u42}\,\mathrm{u52}},\\
\frac{\delta(M_{11}u)}{\delta_{u_3}} &= \frac{\left(\frac{3\,\mathrm{u43}^2}{\mathrm{u41}\,\mathrm{u42}} + \frac{3\,\mathrm{u32}\,\mathrm{u43}}{\mathrm{u42}\,\mathrm{u52}}\right)(t-\mathrm{u3})}{\mathrm{u43}^2} - \frac{\frac{3\,\mathrm{u43}^2}{\mathrm{u41}\,\mathrm{u42}} + \frac{3\,\mathrm{u32}\,\mathrm{u43}}{\mathrm{u42}\,\mathrm{u52}}}{\mathrm{u43}} \qquad (A.40)\\
&- \frac{(t-\mathrm{u3})\left(\frac{3\,\mathrm{u32}}{\mathrm{u42}\,\mathrm{u52}} + \frac{6\,\mathrm{u43}}{\mathrm{u41}\,\mathrm{u42}} - \frac{3\,\mathrm{u43}}{\mathrm{u42}\,\mathrm{u52}}\right)}{\mathrm{u43}},\\
\frac{\delta(M_{12}u^2)}{\delta_{u_3}} &= \frac{2\left(\frac{3\,\mathrm{u43}^2}{\mathrm{u41}\,\mathrm{u42}} - \frac{3\,\mathrm{u43}^2}{\mathrm{u42}\,\mathrm{u52}}\right)(t-\mathrm{u3})^2}{\mathrm{u43}^3} - \frac{(2\,t-2\,\mathrm{u3})\left(\frac{3\,\mathrm{u43}^2}{\mathrm{u41}\,\mathrm{u42}} - \frac{3\,\mathrm{u43}^2}{\mathrm{u42}\,\mathrm{u52}}\right)}{\mathrm{u43}^2} \qquad (A.41)\\
&- \frac{\left(\frac{6\,\mathrm{u43}}{\mathrm{u41}\,\mathrm{u42}} - \frac{6\,\mathrm{u43}}{\mathrm{u42}\,\mathrm{u52}}\right)(t-\mathrm{u3})^2}{\mathrm{u43}^2},\\
\frac{\delta(M_{13}u^3)}{\delta_{u_3}} &= \frac{(t-\mathrm{u3})^3\left(\frac{1}{\mathrm{u41}\,\mathrm{u42}} + \frac{1}{\mathrm{u42}\,\mathrm{u52}} + \frac{1}{\mathrm{u52}\,\mathrm{u53}}\right)}{\mathrm{u43}^2} - \frac{3\,(t-\mathrm{u3})^2\left(\frac{1}{\mathrm{u41}\,\mathrm{u42}} + \frac{1}{\mathrm{u42}\,\mathrm{u52}} + \frac{1}{\mathrm{u52}\,\mathrm{u53}}\right)}{\mathrm{u43}} \qquad (A.42)\\
&+ \frac{(t-\mathrm{u3})^3}{\mathrm{u43}\,\mathrm{u52}\,\mathrm{u53}^2}\\
\frac{\delta M_{20}}{\delta_{u_3}} &= \frac{2\,\mathrm{u32}}{\mathrm{u42}\,\mathrm{u52}},\\
\frac{\delta(M_{21}u)}{\delta_{u_3}} &= \frac{3\,(t-\mathrm{u3})}{\mathrm{u42}\,\mathrm{u52}} - \frac{3\,\mathrm{u32}}{\mathrm{u42}\,\mathrm{u52}}, \qquad (A.43)\\
\frac{\delta(M_{22}u^2)}{\delta_{u_3}} &= -\frac{3\,(2\,t-2\,\mathrm{u3})}{\mathrm{u42}\,\mathrm{u52}},\\
\frac{\delta(M_{23}u^3)}{\delta_{u_3}} &= \frac{3\,(t-\mathrm{u3})^2\left(\frac{1}{\mathrm{u42}\,\mathrm{u52}} + \frac{1}{\mathrm{u52}\,\mathrm{u53}} + \frac{1}{\mathrm{u53}\,\mathrm{u63}}\right)}{\mathrm{u43}} - \frac{(t-\mathrm{u3})^3\left(\frac{1}{\mathrm{u42}\,\mathrm{u52}} + \frac{1}{\mathrm{u52}\,\mathrm{u53}} + \frac{1}{\mathrm{u53}\,\mathrm{u63}}\right)}{\mathrm{u43}^2} \qquad (A.44)\\
&- \frac{(t-\mathrm{u3})^3\left(\frac{1}{\mathrm{u52}\,\mathrm{u53}^2} + \frac{1}{\mathrm{u53}\,\mathrm{u63}^2} + \frac{1}{\mathrm{u53}^2\,\mathrm{u63}}\right)}{\mathrm{u43}}\\
\frac{\delta(M_{33}u^3)}{\delta_{u_3}} &= \frac{3\,(t-\mathrm{u3})^2}{\mathrm{u53}\,\mathrm{u63}\,(\mathrm{u3}-\mathrm{u4})} + \frac{(t-\mathrm{u3})^3}{\mathrm{u53}\,\mathrm{u63}\,(\mathrm{u3}-\mathrm{u4})^2} - \frac{(t-\mathrm{u3})^3}{\mathrm{u53}\,\mathrm{u63}^2\,(\mathrm{u3}-\mathrm{u4})} - \frac{(t-\mathrm{u3})^3}{\mathrm{u53}^2\,\mathrm{u63}\,(\mathrm{u3}-\mathrm{u4})}\\
& \qquad (A.45)
\end{aligned}$$

Dérivée par rapport à u_4 :

$$\frac{\delta B}{\delta_{u_4}} = \begin{pmatrix} \frac{\delta M_{00}}{\delta_{u_4}} + \frac{\delta M_{01}}{\delta_{u_4}} + \frac{\delta M_{02}}{\delta_{u_4}} + \frac{\delta M_{03}}{\delta_{u_4}} \\ \frac{\delta M_{10}}{\delta_{u_4}} + \frac{\delta M_{11}}{\delta_{u_4}} + \frac{\delta M_{12}}{\delta_{u_4}} + \frac{\delta M_{13}}{\delta_{u_4}} \\ \frac{\delta M_{20}}{\delta_{u_4}} + \frac{\delta M_{21}}{\delta_{u_4}} + \frac{\delta M_{22}}{\delta_{u_4}} + \frac{\delta M_{23}}{\delta_{u_4}} \\ \frac{\delta M_{33}}{\delta_{u_4}} \end{pmatrix} \qquad (A.46)$$

Avec :

$$
\begin{aligned}
\frac{\delta M_{00}}{\delta_{u_4}} &= \frac{2\,\mathrm{u43}}{\mathrm{u41}\,\mathrm{u42}} - \frac{\mathrm{u43}^2}{\mathrm{u41}\,\mathrm{u42}^2} - \frac{\mathrm{u43}^2}{\mathrm{u41}^2\,\mathrm{u42}},\\
\frac{\delta(M_{01}u)}{\delta_{u_4}} &= \frac{3\,\mathrm{u43}\;(t-\mathrm{u3})}{\mathrm{u41}\,\mathrm{u42}^2} - \frac{3\;(t-\mathrm{u3})}{\mathrm{u41}\,\mathrm{u42}} + \frac{3\,\mathrm{u43}\;(t-\mathrm{u3})}{\mathrm{u41}^2\,\mathrm{u42}},\\
\frac{\delta(M_{02}u^2)}{\delta_{u_4}} &= -\frac{3\,(t-\mathrm{u3})^2}{\mathrm{u41}\,\mathrm{u42}^2} - \frac{3\,(t-\mathrm{u3})^2}{\mathrm{u41}^2\,\mathrm{u42}},\\
\frac{\delta(M_{03}u^3)}{\delta_{u_4}} &= \frac{(t-\mathrm{u3})^3}{\mathrm{u41}\,\mathrm{u42}\,\mathrm{u43}^2} + \frac{(t-\mathrm{u3})^3}{\mathrm{u41}\,\mathrm{u42}^2\,\mathrm{u43}} + \frac{(t-\mathrm{u3})^3}{\mathrm{u41}^2\,\mathrm{u42}\,\mathrm{u43}}\\
\frac{\delta M_{10}}{\delta_{u_4}} &= \frac{\mathrm{u31}}{\mathrm{u41}\,\mathrm{u42}} - \frac{\mathrm{u31}\,\mathrm{u43}}{\mathrm{u41}\,\mathrm{u42}^2} - \frac{\mathrm{u31}\,\mathrm{u43}}{\mathrm{u41}^2\,\mathrm{u42}} - \frac{\mathrm{u32}\,\mathrm{u53}}{\mathrm{u42}^2\,\mathrm{u52}},\\
\frac{\delta(M_{11}u)}{\delta_{u_4}} &= -\frac{\left(\frac{3\,\mathrm{u43}^2}{\mathrm{u41}\,\mathrm{u42}} + \frac{3\,\mathrm{u32}\,\mathrm{u43}}{\mathrm{u42}\,\mathrm{u52}}\right)(t-\mathrm{u3})}{\mathrm{u43}^2} - \frac{(t-\mathrm{u3})\left(\frac{3\,\mathrm{u43}^2}{\mathrm{u41}\,\mathrm{u42}^2} - \frac{6\,\mathrm{u43}}{\mathrm{u41}\,\mathrm{u42}} - \frac{3\,\mathrm{u32}}{\mathrm{u42}\,\mathrm{u52}} + \frac{3\,\mathrm{u43}^2}{\mathrm{u41}^2\,\mathrm{u42}} + \frac{3\,\mathrm{u32}\,\mathrm{u43}}{\mathrm{u42}^2\,\mathrm{u52}}\right)}{\mathrm{u43}},\\
\frac{\delta(M_{12}u^2)}{\delta_{u_4}} &= -\frac{(t-\mathrm{u3})^2\left(\frac{6\,\mathrm{u43}}{\mathrm{u42}\,\mathrm{u52}} - \frac{6\,\mathrm{u43}}{\mathrm{u41}\,\mathrm{u42}} + \frac{3\,\mathrm{u43}^2}{\mathrm{u41}\,\mathrm{u42}^2} + \frac{3\,\mathrm{u43}^2}{\mathrm{u41}^2\,\mathrm{u42}} - \frac{3\,\mathrm{u43}^2}{\mathrm{u42}^2\,\mathrm{u52}}\right)}{\mathrm{u43}^2}\\
&\quad - \frac{2\left(\frac{3\,\mathrm{u43}^2}{\mathrm{u41}\,\mathrm{u42}} - \frac{3\,\mathrm{u43}^2}{\mathrm{u42}\,\mathrm{u52}}\right)(t-\mathrm{u3})^2}{\mathrm{u43}^3},\\
\frac{\delta(M_{13}u^3)}{\delta_{u_4}} &= -\frac{(t-\mathrm{u3})^3\left(\frac{1}{\mathrm{u41}\,\mathrm{u42}^2} + \frac{1}{\mathrm{u41}^2\,\mathrm{u42}} + \frac{1}{\mathrm{u42}^2\,\mathrm{u52}}\right)}{\mathrm{u43}} - \frac{(t-\mathrm{u3})^3\left(\frac{1}{\mathrm{u41}\,\mathrm{u42}} + \frac{1}{\mathrm{u42}\,\mathrm{u52}} + \frac{1}{\mathrm{u52}\,\mathrm{u53}}\right)}{\mathrm{u43}^2}\\
\frac{\delta M_{20}}{\delta_{u_4}} &= -\frac{\mathrm{u32}^2}{\mathrm{u42}^2\,\mathrm{u52}},\\
\frac{\delta(M_{21}u)}{\delta_{u_4}} &= -\frac{3\,\mathrm{u32}\;(t-\mathrm{u3})}{\mathrm{u42}^2\,\mathrm{u52}},\\
\frac{\delta(M_{22}u^2)}{\delta_{u_4}} &= -\frac{3\,(t-\mathrm{u3})^2}{\mathrm{u42}^2\,\mathrm{u52}},\\
\frac{\delta(M_{23}u^3)}{\delta_{u_4}} &= \frac{(t-\mathrm{u3})^3\left(\frac{1}{\mathrm{u42}\,\mathrm{u52}} + \frac{1}{\mathrm{u52}\,\mathrm{u53}} + \frac{1}{\mathrm{u53}\,\mathrm{u63}}\right)}{\mathrm{u43}^2} + \frac{(t-\mathrm{u3})^3}{\mathrm{u42}^2\,\mathrm{u43}\,\mathrm{u52}}\\
\frac{\delta(M_{33}u^3)}{\delta_{u_4}} &= -\frac{(t-\mathrm{u3})^3}{\mathrm{u43}^2\,\mathrm{u53}\,\mathrm{u63}}
\end{aligned}
\tag{A.47}
\tag{A.48}
\tag{A.49}
$$

Dérivée par rapport à u_5 :

$$
\frac{\delta M_i}{\delta_{u_5}} = \begin{pmatrix}
\frac{\delta M_{00}}{\delta_{u_5}} & \frac{\delta M_{01}}{\delta_{u_5}} & \frac{\delta M_{02}}{\delta_{u_5}} & \frac{\delta M_{03}}{\delta_{u_5}}\\
\frac{\delta M_{10}}{\delta_{u_5}} & \frac{\delta M_{11}}{\delta_{u_5}} & \frac{\delta M_{12}}{\delta_{u_5}} & \frac{\delta M_{13}}{\delta_{u_5}}\\
\frac{\delta M_{20}}{\delta_{u_5}} & \frac{\delta M_{21}}{\delta_{u_5}} & \frac{\delta M_{22}}{\delta_{u_5}} & \frac{\delta M_{23}}{\delta_{u_5}}\\
0 & 0 & 0 & \frac{\delta M_{33}}{\delta_{u_5}}
\end{pmatrix}
\tag{A.50}
$$

$$\frac{\delta M_i}{\delta_{u_5}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \frac{\delta M_{10}}{\delta_{u_5}} & \frac{\delta M_{11}}{\delta_{u_5}} & \frac{\delta M_{12}}{\delta_{u_5}} & \frac{\delta M_{13}}{\delta_{u_5}} \\ \frac{\delta M_{20}}{\delta_{u_5}} & \frac{\delta M_{21}}{\delta_{u_5}} & \frac{\delta M_{22}}{\delta_{u_5}} & \frac{\delta M_{23}}{\delta_{u_5}} \\ 0 & 0 & 0 & \frac{\delta M_{33}}{\delta_{u_5}} \end{pmatrix} \tag{A.51}$$

Avec :

$$\begin{aligned}
\frac{\delta M_{00}}{\delta_{u_5}} &= \frac{\delta M_{01}}{\delta_{u_5}} = \frac{\delta M_{02}}{\delta_{u_5}} = \frac{\delta M_{03}}{\delta_{u_5}} = 0 \\
\frac{\delta M_{10}}{\delta_{u_5}} &= \frac{\mathrm{u32}}{\mathrm{u42\,u52}} - \frac{\mathrm{u32\,u53}}{\mathrm{u42\,u52^2}}, \\
\frac{\delta M_{11}}{\delta_{u_5}} &= -\frac{3\,\mathrm{u32\,u43}}{\mathrm{u42\,u52^2}}, \\
\frac{\delta M_{12}}{\delta_{u_5}} &= \frac{3\,\mathrm{u43^2}}{\mathrm{u42\,u52^2}}, \\
\frac{\delta M_{13}}{\delta_{u_5}} &= -\mathrm{u43^2}\left(\frac{1}{\mathrm{u42\,u52^2}} + \frac{1}{\mathrm{u52\,u53^2}} + \frac{1}{\mathrm{u52^2\,u53}}\right) \\
\frac{\delta M_{20}}{\delta_{u_5}} &= -\frac{\mathrm{u32^2}}{\mathrm{u42\,u52^2}}, \\
\frac{\delta M_{21}}{\delta_{u_5}} &= -\frac{3\,\mathrm{u32\,u43}}{\mathrm{u42\,u52^2}}, \\
\frac{\delta M_{22}}{\delta_{u_5}} &= -\frac{3\,\mathrm{u43^2}}{\mathrm{u42\,u52^2}}, \\
\frac{\delta M_{23}}{\delta_{u_5}} &= \mathrm{u43^2}\left(\frac{1}{\mathrm{u42\,u52^2}} + \frac{1}{\mathrm{u52\,u53^2}} + \frac{1}{\mathrm{u52^2\,u53}} + \frac{1}{\mathrm{u53^2\,u63}}\right) \\
\frac{\delta M_{33}}{\delta_{u_5}} &= -\frac{\mathrm{u43^2}}{\mathrm{u53^2\,u63}}
\end{aligned} \tag{A.52}$$

(A.53)

Dérivée par rapport à u_6 :

$$\frac{\delta M_i}{\delta_{u_6}} = \begin{pmatrix} \frac{\delta M_{00}}{\delta_{u_6}} & \frac{\delta M_{01}}{\delta_{u_6}} & \frac{\delta M_{02}}{\delta_{u_6}} & \frac{\delta M_{03}}{\delta_{u_6}} \\ \frac{\delta M_{10}}{\delta_{u_6}} & \frac{\delta M_{11}}{\delta_{u_6}} & \frac{\delta M_{12}}{\delta_{u_6}} & \frac{\delta M_{13}}{\delta_{u_6}} \\ \frac{\delta M_{20}}{\delta_{u_6}} & \frac{\delta M_{21}}{\delta_{u_6}} & \frac{\delta M_{22}}{\delta_{u_6}} & \frac{\delta M_{23}}{\delta_{u_6}} \\ 0 & 0 & 0 & \frac{\delta M_{33}}{\delta_{u_6}} \end{pmatrix} \tag{A.54}$$

$$\frac{\delta M_i}{\delta_{u_6}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\delta M_{23}}{\delta_{u_6}} \\ 0 & 0 & 0 & \frac{\delta M_{33}}{\delta_{u_6}} \end{pmatrix} \tag{A.55}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\mathrm{u43}^2}{\mathrm{u53}\,\mathrm{u63}^2} \\ 0 & 0 & 0 & -\frac{\mathrm{u43}^2}{\mathrm{u53}\,\mathrm{u63}^2} \end{pmatrix} \tag{A.56}$$

avec :

$$\frac{\delta M_{23}}{\delta_{u_6}} = \frac{\mathrm{u43}^2}{\mathrm{u53}\,\mathrm{u63}^2}, \frac{\delta M_{33}}{\delta_{u_6}} = -\frac{\mathrm{u43}^2}{\mathrm{u53}\,\mathrm{u63}^2} \tag{A.57}$$

A.2 Modèle pièce points multi-hauteurs

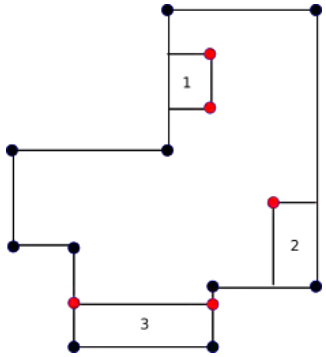


FIGURE A.1 – Illustration du modèle multi hauteur, qui admet des points à différentes hauteurs (rouge) permettant de modéliser certaines particularité des environnements d'intérieur.

Une grande partie des milieux d'intérieur peuvent être représentés par les modèles précédemment décrit. Cependant, une grande partie des pièces possèdent des saillies ou des renfoncements. Ces saillies ne peuvent être simulées à l'aide des modèles précédents étant donné qu'elles possèdent des hauteurs variables comprises entre le sol et le mur. Nous considérons ici uniquement des saillies qui sont rattachées aux murs de la pièce. Afin de pouvoir les modéliser, nous proposons d'ajouter une classe de point possédant des degrés de liberté supplémentaires sur la hauteur, avec deux scalaires h_0 et h_1 définissant respectivement le bas et le haut de la saillie. Un tel point est défini de la manière suivante :

$$p_h = (x, y, h_0, h_1) \quad 0 \leq h_0 < h_1 \leq h \tag{A.58}$$

On distingue alors 4 classes de points suivant les hauteurs associées au point :
- $h_0 = 0$ et $h_1 = h$, c'est un point associé à un coin de la pièce, soutenant la structure principale de la pièce.
- $h_0 = \alpha$ et $h_1 = h$ C'est un point associé à une saillie relié au plafond
- $h_0 = 0$ et $h_1 = \beta$ C'est un point associé à une saillie relié au sol
- $h_0 = \alpha$ et $h_1 = \beta$ C'est un point associé à une saillie n'étant ni reliée au sol, ni au plafond(placard, autre)

On distingue prioritairement la structure "nue" de la pièce définie par l'ensemble des points $p_s = (x, y, 0, h)$. Il est possible d'associer un booléen à chaque autre point définissant

si ce point est à l'intérieur ou à l'extérieur du polygone principal, permettant de modéliser les saillies si $b_{in} = 1$ ou les renfoncements si $b_{in} = 0$.

Il existe plusieurs configuration pour ces points (voir fig) :

— cas 1 : Le point est associé à autre point et à un mur
— cas 2 : Le point est seul et associé à deux murs
— cas 3 : Le point est associé à un autre point et sont associés à 3 murs

Les cas 1 et 3 nécessitent ici deux points pour être représentés, mais ces deux points sont contraints entre eux, partageant les mêmes informations de hauteur, et sont également alignés le long de la direction du mur auquel ils sont associés.

Le cas 1 est caractérisé par deux points, équivalent à 8 degrés de liberté (DL), $p_0 = (x_0, y_0, \alpha_0, \beta_0)$ et $p_1 = (x_1, y_1, \alpha_1, \beta_1)$. On peut aisément réduire(contraindre) la paramétrisation à 6 DL sachant que $\alpha_0 = alpha_1$ et $\beta_0 = \beta_1$. De plus, ces points étant associés à un plan défini par des points de la structure p_s, il est possible d'obtenir une direction $\vec{v}$, qui permet à partir d'un point $p_0 = (x_0, y_0, \alpha_0, \beta_0)$ et d'une distance d le long de cette direction de retrouver le second point. On a donc non plus 6DL, mais uniquement 5DL.

La paramétrisation du cas 3 peut quant à elle être simplifiée de 8DL à 3DL. En effet, étant associée à 3 murs, on considérera toujours que la direction de cette saillie est égale à la direction du mur du milieu auquel elle est associée. Ce cas peut donc être paramétré par 3 composantes, une distance orthogonale au mur du milieu d, ainsi que les paramètres de hauteur α_0 et β_0.

Bibliographie

[Agarwal 2010] Sameer Agarwal, Keir Mierle et Others. *Ceres Solver*. `http://ceres-solver.org`, 2010. (Cité en page 47.)

[Albl 2015] Cenek Albl, Zuzana Kukelova et Tomas Pajdla. *R6P - Rolling Shutter Absolute Camera Pose*. June 2015. (Cité en page 69.)

[Albl 2016] Cenek Albl, Zuzana Kukelova et Tomas Pajdla. *Rolling Shutter Absolute Pose Problem With Known Vertical Direction*. Dans The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016. (Cité en page 69.)

[Barnes 2009] Connelly Barnes, Eli Shechtman, Adam Finkelstein et Dan B Goldman. *PatchMatch : A randomized correspondence algorithm for structural image editing*. ACM Transactions on Graphics-TOG, vol. 28, no. 3, page 24, 2009. (Cité en page 147.)

[Bartoli 2003] Adrien Bartoli. *Reconstruction et alignement en vision 3D : points, droites, plans et caméras*. Theses, Institut National Polytechnique de Grenoble - INPG, septembre 2003. (Cité en page 10.)

[Bartoli 2005] Adrien Bartoli et Peter Sturm. *Structure-from-motion using lines : Representation, triangulation and bundle adjustment*. Computer Vision and Image Understanding, vol. 100, page 2005, 2005. (Cité en page 10.)

[Bay 2008] Herbert Bay, Andreas Ess, Tinne Tuytelaars et Luc Van Gool. *Speeded-Up Robust Features (SURF)*. Comput. Vis. Image Underst., vol. 110, no. 3, pages 346–359, juin 2008. (Cité en page 12.)

[Blanco 2014] Jose-luis Blanco. *A tutorial on SE(3) transformation parameterizations and on-manifold optimization*, 2014. (Cité en pages 22, 46 et 156.)

[Bouguet 2000] Jean-Yves Bouguet. *Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm*, 2000. (Cité en page 11.)

[Brown 1966] D. C. Brown. *Decentering Distortion of Lenses*. vol. 32, no. 3, pages 444–462, 1966. (Cité en page 33.)

[Cadena 2016] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid et John J Leonard. *Past, present, and future of simultaneous localization and mapping : Toward the robust-perception age*. IEEE Transactions on Robotics, vol. 32, no. 6, pages 1309–1332, 2016. (Cité en pages 7 et 8.)

[Calonder 2010] Michael Calonder, Vincent Lepetit, Christoph Strecha et Pascal Fua. *BRIEF : Binary Robust Independent Elementary Features*. Dans Proceedings of the 11th European Conference on Computer Vision : Part IV, ECCV'10, pages 778–792, Berlin, Heidelberg, 2010. Springer-Verlag. (Cité en page 12.)

[Canny 1987] John Canny. *A computational approach to edge detection*. Dans Readings in Computer Vision, pages 184–203. Elsevier, 1987. (Cité en page 113.)

[Chirikjian 2012] Gregory Chirikjian. Stochastic Models, Information Theory, and Lie Groups Volume 2. Applied and Numerical Harmonic Analysis. Birkhäuser/Springer, New York, 2012. (Cité en pages 22, 24, 156 et 157.)

[Civera 2006] Javier Civera, Andrew J. Davison et J. M. Martínez Montiel. *Unified inverse depth parametrization for monocular slam*. Dans In Proceedings of Robotics : Science and Systems, 2006. (Cité en page 14.)

[Comport 2006] Andrew I Comport, Eric Marchand, Muriel Pressigout et Francois Chaumette. *Real-time markerless tracking for augmented reality : the virtual visual servoing framework*. IEEE Transactions on visualization and computer graphics, vol. 12, no. 4, pages 615–628, 2006. (Cité en pages 11 et 128.)

[Concha Belenguer 2015] Alejo Concha Belenguer et Javier Civera Sancho. *DPPTAM : Dense Piecewise Planar Tracking and Mapping from a Monocular Sequence*. 2015. (Cité en page 105.)

[Dai 2016] Yuchao Dai, Hongdong Li et Laurent Kneip. *Rolling Shutter Camera Relative Pose : Generalized Epipolar Geometry*. CoRR, vol. abs/1605.00475, 2016. (Cité en page 69.)

[Davison 1998] Andrew J Davison et David W Murray. *Mobile robot localisation using active vision*. Dans European Conference on Computer Vision, pages 809–825. Springer, 1998. (Cité en page 13.)

[Davison 2003] Andrew J. Davison. *Real-Time Simultaneous Localisation and Mapping with a Single Camera*. Dans 9th IEEE International Conference on Computer Vision (ICCV 2003), 14-17 October 2003, Nice, France, pages 1403–1410, 2003. (Cité en page 14.)

[de Boor 1972] Carl de Boor. *On calculating with B-splines*. Journal of Approximation Theory, vol. 6, no. 1, pages 50 – 62, 1972. (Cité en pages 37 et 77.)

[Dellaert 2006] Frank Dellaert et Michael Kaess. *Square Root SAM : Simultaneous localization and mapping via square root information smoothing*. The International Journal of Robotics Research, vol. 25, no. 12, pages 1181–1203, 2006. (Cité en page 49.)

[Dubrofsky 2009] Elan Dubrofsky. *Homography Estimation*, 2009. (Cité en page 34.)

[Duda 1972] Richard O Duda et Peter E Hart. *Use of the Hough transformation to detect lines and curves in pictures*. Communications of the ACM, vol. 15, no. 1, pages 11–15, 1972. (Cité en pages 10 et 113.)

[Eade 2013] Ethan Eade. *Lie groups for 2d and 3d transformations*. 2013. (Cité en pages 22 et 28.)

[Eade 2017] Ethan Eade. *Differential of the Exponential Map*, 2017. (Cité en page 157.)

[Eigen 2014] David Eigen, Christian Puhrsch et Rob Fergus. *Depth map prediction from a single image using a multi-scale deep network*. Dans Advances in neural information processing systems, pages 2366–2374, 2014. (Cité en pages 130 et 136.)

[Engel 2014] J. Engel, T. Schöps et D. Cremers. *LSD-SLAM : Large-Scale Direct Monocular SLAM*. Dans ECCV, September 2014. (Cité en page 18.)

[Esparza-Jiménez 2014] J. O. Esparza-Jiménez, M. Devy et J. L. Gordillo. *EKF-based SLAM fusing heterogeneous landmarks*. Dans 17th International Conference on Information Fusion (FUSION), pages 1–8, July 2014. (Cité en page 10.)

[Esparza-Jiménez 2016] Jorge Othón Esparza-Jiménez, Michel Devy et José L. Gordillo. *Visual EKF-SLAM from Heterogeneous Landmarks*. Sensors, vol. 16, no. 4, 2016. (Cité en page 10.)

[Faugeras 1988] Olivier Faugeras et F. Lustman. *Motion and structure from motion in a piecewise planar environment*. Rapport technique RR-0856, INRIA, juin 1988. (Cité en page 33.)

[Fayer 2017] Julien Fayer, Simone Gasparini, Géraldine Morin et Maxime Daisy. *Critère de confiance géométrique pour l ?Inpainting basée patch*. 2017. (Cité en page 140.)

[Fischler 1987] Martin A Fischler et Robert C Bolles. *Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography.* Dans Readings in computer vision, pages 726–740. Elsevier, 1987. (Cité en page 139.)

[Flint 2010a] A. Flint, C. Mei, I. Reid et D. Murray. *Growing semantically meaningful models for visual SLAM.* Dans 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 467–474, June 2010. (Cité en pages 105, 114 et 150.)

[Flint 2010b] Alex Flint, Christopher Mei, David Murray et Ian Reid. *A Dynamic Programming Approach to Reconstructing Building Interiors.* Dans Kostas Daniilidis, Petros Maragos et Nikos Paragios, editeurs, Computer Vision – ECCV 2010, pages 394–407, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. (Cité en pages 105, 115, 116 et 119.)

[Flint 2011] A. Flint, D. Murray et I. Reid. *Manhattan scene understanding using monocular, stereo, and 3D features.* Dans 2011 International Conference on Computer Vision, pages 2228–2235, Nov 2011. (Cité en pages 105 et 151.)

[Forssén 2010] P. E. Forssén et E. Ringaby. *Rectifying rolling shutter video from handheld devices.* Dans Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 507–514, June 2010. (Cité en pages 39, 69 et 149.)

[Forster 2015] Christian Forster, Luca Carlone, Frank Dellaert et Davide Scaramuzza. *IMU Preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation.* Dans Proceedings of Robotics : Science and Systems, Rome, Italy, July 2015. (Cité en pages 18, 19, 156 et 157.)

[Furgale 2012] P. Furgale, T. D. Barfoot et G. Sibley. *Continuous-time batch estimation using temporal basis functions.* Dans Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 2088–2095, May 2012. (Cité en page 40.)

[Furgale 2015] Paul Furgale, Chi Hay Tong, Timothy D. Barfoot et Gabe Sibley. *Continuous-time Batch Trajectory Estimation Using Temporal Basis Functions.* Int. J. Rob. Res., vol. 34, no. 14, pages 1688–1710, décembre 2015. (Cité en page 40.)

[Furukawa 2010a] Yasutaka Furukawa, Brian Curless, Steven M. Seitz et Richard Szeliski. *Towards Internet-scale Multi-view Stereo.* Dans CVPR, 2010. (Cité en page 103.)

[Furukawa 2010b] Yasutaka Furukawa et Jean Ponce. *Accurate, dense, and robust multiview stereopsis.* IEEE transactions on pattern analysis and machine intelligence, vol. 32, no. 8, pages 1362–1376, 2010. (Cité en pages 102 et 103.)

[Gallup 2007] David Gallup, Jan-Michael Frahm, Philippos Mordohai, Qingxiong Yang et Marc Pollefeys. *Real-time plane-sweeping stereo with multiple sweeping directions.* Dans Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, pages 1–8. IEEE, 2007. (Cité en pages 102 et 105.)

[Gálvez-López 2012] Dorian Gálvez-López et J. D. Tardós. *Bags of Binary Words for Fast Place Recognition in Image Sequences.* IEEE Transactions on Robotics, vol. 28, no. 5, pages 1188–1197, October 2012. (Cité en page 17.)

[Gohard 2018a] Philippe-Antoine Gohard, Bertrand Vandeportaele et Michel Devy. *Interpolation et optimisation spatio-temporelle pour l'estimation de poses de caméra à obturateur déroulant.* Dans RFIAP. INSTICC, SciTePress, 2018. (Cité en page 19.)

[Gohard 2018b] Philippe-Antoine Gohard, Bertrand Vandeportaele et Michel Devy. *Spatio-temporal Optimization for Rolling Shutter Camera Pose Interpolation.* Dans CCIS. Springer, 2018. (Cité en page 19.)

[Gonzalez 2013] Aurélien Gonzalez. *Localisation par vision multi-spectrale. Application aux systèmes embarqués.* Theses, INSA de Toulouse, juillet 2013. (Cité en pages 15 et 19.)

[Grisetti 2010] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss et Wolfram Burgard. *A tutorial on graph-based SLAM.* IEEE Intelligent Transportation Systems Magazine, vol. 2, no. 4, pages 31–43, 2010. (Cité en pages 43 et 45.)

[Guo 2014] Chao Guo, Dimitrios Kottas, Ryan DuToit, Ahmed Ahmed, Ruipeng Li et Stergios Roumeliotis. *Efficient Visual-Inertial Navigation using a Rolling-Shutter Camera with Inaccurate Timestamps.* Dans Proceedings of Robotics : Science and Systems, Berkeley, USA, July 2014. (Cité en page 55.)

[Harris 1987] C G Harris et J M Pike. *3D Positional Integration from Image Sequences.* Dans In Proc. Alvey Vision Conference, Cambridge.England, 1987. (Cité en pages 13 et 14.)

[Harris 1988] Chris Harris et Mike Stephens. *A combined corner and edge detector.* Dans In Proc. of Fourth Alvey Vision Conference, pages 147–151, 1988. (Cité en page 9.)

[Hartley 1997] Richard I Hartley et Peter Sturm. *Triangulation.* Computer vision and image understanding, vol. 68, no. 2, pages 146–157, 1997. (Cité en page 35.)

[Hartley 2004] R. I. Hartley et A. Zisserman. Multiple view geometry in computer vision. Cambridge University Press, ISBN : 0521540518, second édition, 2004. (Cité en pages 34, 43 et 134.)

[Hedborg 2011] J. Hedborg, E. Ringaby, P. E. Forssén et M. Felsberg. *Structure and motion estimation from rolling shutter video.* Dans Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, pages 17–23, Nov 2011. (Cité en page 69.)

[Hedborg 2012] J. Hedborg, P. E. Forssén, M. Felsberg et E. Ringaby. *Rolling shutter bundle adjustment.* Dans Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 1434–1441, June 2012. (Cité en pages 39, 59, 60, 69, 75 et 149.)

[Hertzberg 2008] Christoph Hertzberg. *A framework for sparse, non-linear least squares problems on manifolds.* Dans UNIVERSITÄT BREMEN. Citeseer, 2008. (Cité en page 45.)

[Hoiem 2005] Derek Hoiem, Alexei A Efros et Martial Hebert. *Geometric context from a single image.* Dans Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on, volume 1, pages 654–661. IEEE, 2005. (Cité en page 151.)

[Kawai 2016] Norihiko Kawai, Tomokazu Sato et Naokazu Yokoya. *Diminished reality based on image inpainting considering background geometry.* IEEE transactions on visualization and computer graphics, vol. 22, no. 3, pages 1236–1247, 2016. (Cité en page 147.)

[Kerl 2015] C. Kerl, J. Stückler et D. Cremers. *Dense Continuous-Time Tracking and Mapping with Rolling Shutter RGB-D Cameras.* Dans 2015 IEEE International Conference on Computer Vision (ICCV), pages 2264–2272, Dec 2015. (Cité en pages 155 et 156.)

[Kim 1995] Myoung-Jun Kim, Myung-Soo Kim et Sung Yong Shin. *A general construction scheme for unit quaternion curves with simple high order derivatives.* Proceedings of the 22nd annual conference on Computer graphics and interactive techniques - SIGGRAPH '95, pages 369–376, 1995. (Cité en pages 38, 40 et 77.)

[Kim 2016] J. H. Kim, C. Cadena et I. Reid. *Direct semi-dense SLAM for rolling shutter cameras*. Dans 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 1308–1315, May 2016. (Cité en page 102.)

[Klein 2007] Georg Klein et David Murray. *Parallel Tracking and Mapping for Small AR Workspaces*. Dans Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07), Nara, Japan, November 2007. (Cité en page 16.)

[Klein 2008] Georg Klein et David Murray. *Improving the Agility of Keyframe-Based SLAM*. Dans Proc. 10th European Conference on Computer Vision (ECCV'08), pages 802–815, Marseille, October 2008. (Cité en page 10.)

[Klein 2009] Georg Klein et David Murray. *Parallel Tracking and Mapping on a Camera Phone*. Dans Proc. Eigth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'09), Orlando, October 2009. (Cité en pages 65 et 68.)

[Konolige 2008] Kurt Konolige et Motilal Agrawal. *FrameSLAM : From bundle adjustment to real-time visual mapping*. IEEE Transactions on Robotics, vol. 24, no. 5, pages 1066–1077, 2008. (Cité en page 16.)

[Kuemmerle 2011] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige et W. Burgard. *g2o : A General Framework for Graph Optimization*. Dans Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pages 3607–3613, Shanghai, China, May 2011. (Cité en pages 47, 89 et 92.)

[Lee 2009] D. C. Lee, M. Hebert et T. Kanade. *Geometric reasoning for single image structure recovery*. Dans 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 2136–2143, June 2009. (Cité en pages 104, 113, 114, 118, 119, 136 et 150.)

[Lee 2017] Chen-Yu Lee, Vijay Badrinarayanan, Tomasz Malisiewicz et Andrew Rabinovich. *Roomnet : End-to-end room layout estimation*. arXiv preprint arXiv :1703.06241, 2017. (Cité en page 130.)

[Lefaudeux 2013] Benjamin Lefaudeux. *Détection, localisation et suivi des obstacles et objets mobiles à partir d'une plate forme de stéréo-vision*. PhD thesis, Paris, ENMP, 2013. (Cité en page 12.)

[Leutenegger 2011] Stefan Leutenegger, Margarita Chli et Roland Y Siegwart. *BRISK : Binary robust invariant scalable keypoints*. Dans Computer Vision (ICCV), 2011 IEEE International Conference on, pages 2548–2555. IEEE, 2011. (Cité en page 12.)

[Leutenegger 2015] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart et Paul Furgale. *Keyframe-based Visual-inertial Odometry Using Nonlinear Optimization*. Int. J. Rob. Res., vol. 34, no. 3, pages 314–334, mars 2015. (Cité en page 19.)

[Li 2013] M. Li, B.H. Kim et A. I. Mourikis. *Real-Time Motion Estimation on a Cellphone using Inertial Sensing and a Rolling-Shutter Camera*. Dans Proceedings of the IEEE International Conference on Robotics and Automation, pages 4697–4704, Karlsruhe, Germany, May 2013. (Cité en page 55.)

[Li 2014] M. Li, H. Yu, X. Zheng et A. I. Mourikis. *High-fidelity sensor modeling and calibration in vision-aided inertial navigation*. Dans Proceedings of the IEEE International Conference on Robotics and Automation, pages 409–416, Hong Kong, May 2014. (Cité en page 16.)

[Liu 2018] Chen Liu, Jimei Yang, Duygu Ceylan, Ersin Yumer et Yasutaka Furukawa. *PlaneNet : Piece-wise Planar Reconstruction from a Single RGB Image*, 2018. cite arxiv :1804.06278Comment : CVPR 2018. (Cité en pages 130 et 136.)

[LLC 2015] Vuforia LLC. *Vuforia website*, 2015. (Cité en pages 6 et 133.)

[LocherMichael 2016] A. LocherMichael, P. Riemenschneider, H. Riemenschneider et L. Van Gool. *Mobile phone and cloud — A dream team for 3D reconstruction.* Dans 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), March 2016. (Cité en page 102.)

[Lourakis 2009] Manolis IA Lourakis et Antonis A Argyros. *SBA : A software package for generic sparse bundle adjustment.* ACM Transactions on Mathematical Software (TOMS), vol. 36, no. 1, page 2, 2009. (Cité en pages 15 et 47.)

[Lowe 2004] David G. Lowe. *Distinctive Image Features from Scale-Invariant Keypoints.* Int. J. Comput. Vision, vol. 60, no. 2, pages 91–110, novembre 2004. (Cité en pages 9 et 12.)

[Lu 1997] Feng Lu et Evangelos Milios. *Globally consistent range scan alignment for environment mapping.* Autonomous robots, vol. 4, no. 4, pages 333–349, 1997. (Cité en page 47.)

[Lucas 1981] Bruce D. Lucas et Takeo Kanade. *An Iterative Image Registration Technique with an Application to Stereo Vision.* Dans Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc. (Cité en page 11.)

[Magerand 2010] Ludovic Magerand et Adrien Bartoli. *A generic rolling shutter camera model and its application to dynamic pose estimation.* 2010. (Cité en page 68.)

[Magerand 2012] Ludovic Magerand, Adrien Bartoli, Omar Ait-Aider et Daniel Pizarro. *Global optimization of object pose and motion from a single rolling shutter image with automatic 2d-3d matching.* Dans European Conference on Computer Vision, pages 456–469. Springer, 2012. (Cité en page 68.)

[Magerand 2014] Ludovic Magerand. *Calcul de pose dynamique avec les caméras CMOS utilisant une acquisition séquentielle.* PhD thesis, Université Blaise Pascal-Clermont-Ferrand II, 2014. (Cité en page 68.)

[Meingast 2005] Marci Meingast, Christopher Geyer et Shankar Sastry. *Geometric Models of Rolling-Shutter Cameras.* CoRR, vol. abs/cs/0503076, 2005. (Cité en pages 52, 54, 68 et 149.)

[Mičušík 2010] Branislav Mičušík et Jana Košecká. *Multi-view Superpixel Stereo in Urban Environments.* International Journal of Computer Vision, vol. 89, no. 1, pages 106–119, Aug 2010. (Cité en pages 102 et 105.)

[Mouragnon 2006] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser et P. Sayd. *Real Time Localization and 3D Reconstruction.* Dans 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), volume 1, pages 363–370, 2006. (Cité en page 16.)

[Mourikis 2007] A. I. Mourikis et S. I. Roumeliotis. *A multi-state constraint Kalman filter for vision-aided inertial navigation.* Dans Proceedings of the IEEE International Conference on Robotics and Automation, pages 3565–3572, Rome, Italy, April 10-14 2007. (Cité en pages 16 et 19.)

[Mur-Artal 2015a] R. Mur-Artal, J. M. M. Montiel et J. D. Tardós. *ORB-SLAM : A Versatile and Accurate Monocular SLAM System.* IEEE Transactions on Robotics, vol. 31, no. 5, pages 1147–1163, Oct 2015. (Cité en pages 17 et 57.)

[Mur-Artal 2015b] Raul Mur-Artal et Juan D. Tardós. *Probabilistic Semi-Dense Mapping from Highly Accurate Feature-Based Monocular SLAM.* Dans Robotics : Science

and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015, 2015. (Cité en page 17.)

[Mur-Artal 2016] Raul Mur-Artal et Juan D. Tardós. *ORB-SLAM2 : an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras.* CoRR, vol. abs/1610.06475, 2016. (Cité en page 17.)

[Mur-Artal 2017] Raul Mur-Artal et Juan D. Tardós. *Visual-Inertial Monocular SLAM With Map Reuse.* IEEE Robotics and Automation Letters, vol. 2, no. 2, pages 796–803, 2017. (Cité en pages 17 et 19.)

[Newcombe 2011] R. A. Newcombe, S. J. Lovegrove et A. J. Davison. *DTAM : Dense tracking and mapping in real-time.* Dans 2011 International Conference on Computer Vision, pages 2320–2327, Nov 2011. (Cité en page 18.)

[Nistér 2004] David Nistér. *An Efficient Solution to the Five-Point Relative Pose Problem.* IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, no. 6, pages 756–777, juin 2004. (Cité en page 34.)

[Ortiz 2012] Raphael Ortiz. *FREAK : Fast Retina Keypoint.* Dans Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), CVPR '12, pages 510–517, Washington, DC, USA, 2012. IEEE Computer Society. (Cité en page 12.)

[Oth 2013] L. Oth, P. Furgale, L. Kneip et R. Siegwart. *Rolling Shutter Camera Calibration.* Dans Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pages 1360–1367, June 2013. (Cité en page 149.)

[Pizzoli 2014] M. Pizzoli, C. Forster et D. Scaramuzza. *REMODE : Probabilistic, monocular dense reconstruction in real time.* Dans 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 2609–2616, May 2014. (Cité en page 18.)

[Ramadasan 2015] Datta Ramadasan. *SLAM temporel à contraintes multiples.* PhD thesis, 2015. Thèse de doctorat dirigée par Chateau, Thierry Vision pour la Robotique Clermont-Ferrand 2 2015. (Cité en page 39.)

[Renaud 2016] Marc Renaud. *Étude des groupes et algèbres de Lie. Théorème de réduction/reconstruction de Lie-Poisson. Application aux groupes spécial orthogonal SO(3) et spécial euclidien SE(3) et à leurs algèbres so(3) et se(3).* Research Report Rapport LAAS n° 16042, LAAS-CNRS ; INSA Toulouse, mars 2016. (Cité en pages 22 et 27.)

[Rodrigues 1840] Olinde Rodrigues. Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace : et de la variation des cordonnées provenant de ces déplacements considérés indépendamment des causes qui peuvent les produire. 1840. (Cité en page 27.)

[Rosten 2006] Edward Rosten et Tom Drummond. *Machine learning for high-speed corner detection.* Dans European conference on computer vision, pages 430–443. Springer, 2006. (Cité en page 9.)

[Rosten 2010] Edward Rosten, Reid Porter et Tom Drummond. *Faster and better : A machine learning approach to corner detection.* IEEE transactions on pattern analysis and machine intelligence, vol. 32, no. 1, pages 105–119, 2010. (Cité en page 9.)

[Rother 2002] Carsten Rother. *A new approach to vanishing point detection in architectural environments.* Image Vision Comput., vol. 20, no. 9-10, pages 647–655, 2002. (Cité en pages 105 et 113.)

[Roussillon 2011] Cyril Roussillon, Aurélien Gonzalez, Joan Solà, Jean-Marie Codol, Nicolas Mansard, Simon Lacroix et Michel Devy. *RT-SLAM : A Generic and Real-Time Visual SLAM Implementation*. Dans James L. Crowley, Bruce A. Draper et Monique Thonnat, editeurs, Computer Vision Systems, pages 31–40, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. (Cité en pages 15 et 19.)

[Roussillon 2012] Cyril Roussillon, Aurélien Gonzalez, Joan Solà, Jean-Marie Codol, Nicolas Mansard, Simon Lacroix et Michel Devy. *RT-SLAM : A Generic and Real-Time Visual SLAM Implementation*. CoRR, vol. abs/1201.5450, 2012. (Cité en page 15.)

[Rublee 2011] Ethan Rublee, Vincent Rabaud, Kurt Konolige et Gary Bradski. *ORB : An Efficient Alternative to SIFT or SURF*. Dans Proceedings of the 2011 International Conference on Computer Vision, ICCV '11, pages 2564–2571, Washington, DC, USA, 2011. IEEE Computer Society. (Cité en page 12.)

[Schwing 2013] Alexander G Schwing, Sanja Fidler, Marc Pollefeys et Raquel Urtasun. *Box in the box Joint 3d layout and object reasoning from single images*. Dans Computer Vision (ICCV), 2013 IEEE International Conference on, pages 353–360. IEEE, 2013. (Cité en pages 116 et 119.)

[Serafin 2017] Jacopo Serafin. *Using Extended Measurements and Geometric Features for Robust Long-Term Localization and Mapping*. PhD thesis, Department of Computer, Control, and Management Engineering "Antonio Ruberti" at Sapienza University of Rome, Rome, Italy, 2 2017. (Cité en page 10.)

[Shi 1994] Jianbo Shi et Carlo Tomasi. *Good Features to Track*. pages 593–600, 1994. (Cité en pages 9 et 11.)

[Sibley 2006] Gabe Sibley. *Sliding window filters for slam*. Rapport technique, 2006. (Cité en page 16.)

[Šmıd 2017] Matej Šmıd et Jirı Matas. *Rolling Shutter Camera Synchronization with Sub-millisecond Accuracy*. Dans Proc. 12th Int. Conf. Comput. Vis. Theory Appl, page 8, 2017. (Cité en page 54.)

[Snavely 2006] Noah Snavely, Steven M Seitz et Richard Szeliski. *Photo tourism : exploring photo collections in 3D*. Dans ACM transactions on graphics (TOG), volume 25, pages 835–846. ACM, 2006. (Cité en page 47.)

[Solà 2012] Joan Solà, Teresa Vidal-Calleja, Javier Civera et José María Martínez Montiel. *Impact of Landmark Parametrization on Monocular EKF-SLAM with Points and Lines*. International Journal of Computer Vision, vol. 97, no. 3, pages 339–368, May 2012. (Cité en page 10.)

[Solà 2017] Joan Solà. *Quaternion kinematics for the error-state Kalman filter*. CoRR, vol. abs/1711.02508, 2017. (Cité en pages 22, 24 et 26.)

[Solà 2005] Joan Solà, André Monin, Michel Devy et Thomas Lemaire. *Undelayed initialization in bearing only SLAM*. Dans IROS, pages 2499–2504. IEEE, 2005. (Cité en page 14.)

[Steven Lovegrove 2013] Gabe Sibley Steven Lovegrove Alonso Patron-Perez. *Spline Fusion : A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras*. Dans Proceedings of the British Machine Vision Conference. BMVA Press, 2013. (Cité en pages 40, 41, 42, 70, 71, 75, 77, 84, 92, 149 et 153.)

[Strasdat 2010] Hauke Strasdat, JMM Montiel et Andrew J Davison. *Real-time monocular SLAM : Why filter ?* Dans Robotics and Automation (ICRA), 2010 IEEE International Conference on, pages 2657–2664. IEEE, 2010. (Cité en page 16.)

[Strasdat 2012] Hauke Strasdat. *Local accuracy and global consistency for efficient visual slam*. PhD thesis, Citeseer, 2012. (Cité en pages 16, 17, 22 et 24.)

[Tamaazousti 2013] Mohamed Tamaazousti. *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, 2013. (Cité en page 2.)

[Tateno 2017] Keisuke Tateno, Federico Tombari, Iro Laina et Nassir Navab. *CNN-SLAM : Real-time dense monocular SLAM with learned depth prediction*. 2017. (Cité en page 130.)

[Tomasi 1991] Carlo Tomasi et Takeo Kanade. *Detection and Tracking of Point Features*. Rapport technique, International Journal of Computer Vision, 1991. (Cité en page 11.)

[Triggs 1999] Bill Triggs, Philip F McLauchlan, Richard I Hartley et Andrew W Fitzgibbon. *Bundle adjustment—a modern synthesis*. Dans International workshop on vision algorithms, pages 298–372. Springer, 1999. (Cité en pages 43 et 49.)

[Triggs 2000] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley et Andrew W. Fitzgibbon. *Bundle Adjustment — A Modern Synthesis*. Dans Bill Triggs, Andrew Zisserman et Richard Szeliski, editeurs, Vision Algorithms : Theory and Practice, pages 298–372, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. (Cité en pages 15, 43 et 49.)

[Vandeportaele 2017] Bertrand Vandeportaele, Philippe-Antoine Gohard, Michel Devy et Benjamin Coudrin. *Pose Interpolation for Rolling Shutter Cameras using Non Uniformly Time-Sampled B-splines*. Dans Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 6 : VISAPP, (VISIGRAPP 2017), pages 286–293. INSTICC, SciTePress, 2017. (Cité en pages 19, 92 et 94.)

[von Gioi 2010] Rafael Grompone von Gioi, Jérémie Jakubowicz, Jean-Michel Morel et Gregory Randall. *LSD : A Fast Line Segment Detector with a False Detection Control*. IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 4, pages 722–732, 2010. (Cité en pages 10, 113 et 127.)

[Yang 2009] Huixian Yang, WenLong Yue, Yabin He, Huixian Huang et Haixia Xia. *The Deduction of Coefficient Matrix for Cubic Non-Uniform B-Spline Curves*. 2009 First International Workshop on Education Technology and Computer Science, no. 3, pages 607–609, 2009. (Cité en pages 38, 77, 78, 79 et 80.)

[Yi 2016] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit et Pascal Fua. *LIFT : Learned Invariant Feature Transform*. CoRR, vol. abs/1603.09114, 2016. (Cité en page 12.)

[Zhang 2013] Lilian Zhang et Reinhard Koch. *An Efficient and Robust Line Segment Matching Approach Based on LBD Descriptor and Pairwise Geometric Consistency*. J. Vis. Comun. Image Represent., vol. 24, no. 7, pages 794–805, octobre 2013. (Cité en pages 13 et 128.)