UNIVERSITE D'EVRY VAL D'ESSONNE
Laboratoire d'Informatique, Biologie Intégrative et
Systèmes Complexes

Quelques contributions en localisation et cartographie simultanées multi-capteurs : application à la réalité augmentée

Thèse présentée pour obtenir le titre de
Docteur de l'Université d'Evry
Spécialité : Vision par Ordinateur

présentée par
Maxime Boucher

Table des matières

Introduction

Introduction générale

L'être humain perçoit le monde dans lequel il évolue par le biais de ses sens, le plus important étant sans doute celui de la vue. Grâce aux stimuli reçus, l'homme arrive à créer mentalement une représentation de son environnement. Lorsqu'une information légèrement différente est perçue, le cerveau est naturellement capable de l'associer à une information précédemment obtenue.

Récemment dans l'histoire de l'humanité, sont apparus les ordinateurs, machines dotées de formidables capacités de calcul. Tôt dans le développement de l'informatique, en intelligence artificielle, les scientifiques ont tenté d'apporter aux ordinateurs le sens de la vue. Parmi ces tentatives on peut citer les travaux de Roberts, [Roberts, 1963]. Les ordinateurs, tout en gagnant en puissance ont également gagné en miniaturisation. Aujourd'hui, on trouve dans nos poches des ordinateurs rivalisant en puissance avec les supercaculateurs d'il y a vingt ans. Ces ordinateurs miniatures, que l'on utilise généralement pour téléphoner ou naviguer sur internet, comme les plus gros, que l'on utilise pour travailler, se sont récemment vus dotés de capteurs fournissant des informations très analogues à celles des sens humains. Ainsi en est il des caméras qui apportent le sens de la vue aux machines. Cette information est cependant moins riche que celle de l'être humain doté de deux yeux. La vision stéréoscopique apporte en effet une information cruciale, celle de la profondeur de la scène. Les ordinateurs peuvent cependant être dotés de sens que ne possèdent pas les humains. Les *Systèmes de Guidages par Satellite*, ou GPS, en font partie, apportant une information de localisation absolue. Ces systèmes sont aujourd'hui utilisés dans un grand nombre d'applications, allant des véhicules aux téléphones portables. Ils apportent une information de localisation fiable et immédiate, mais ne répondent qu'à une partie de la

problématique. En effet, sur les six ou sept degrés de liberté qui caractérisent la machine par rapport à son environnement, les GPS n'en renseignent que deux. Si l'on souhaite apporter à une machine la capacité d'accéder à ces informations il faut aller au delà de ce type de mesures. Par exemple en utilisant des capteurs visuels. Cependant, à la différence de l'être humain, il n'est pas naturel pour une machine dotée d'une caméra de mettre en relation la connaissance à l'instant présent avec celle issue du passé. Il faut lui donner la possiblité de mettre en relation les informations acquises par le passé avec celles perçues à l'instant présent.

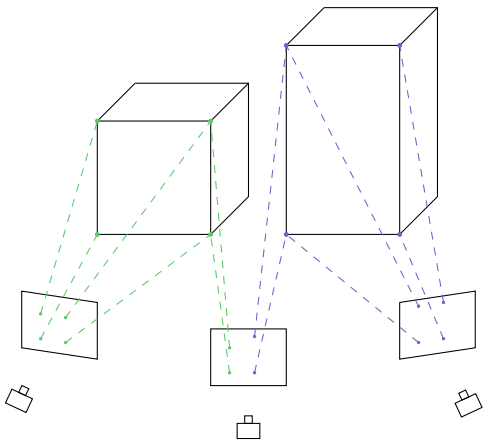


FIGURE 1 – Schématisation de l'observation d'une scène par une caméra selon plusieurs points de vues. Des points caractéristiques se projettent dans les images. L'observation d'un même point dans deux images permet d'en estimer la position.

Ces tâches difficiles qui consistent à tirer de l'information des images de la caméra au cours du temps pour cartographier l'environnement et se localiser à l'intérieur de celui-ci, si elle sont réalisées simultanément sont appelées *Localisation et Cartographie Simultanée* ou *SLAM*. La figure 1 illustre ce problème.

Développées à la fois par les communautés de robotique et vision par ordinateur les applications sont multiples. Des robots bénéficient de cette capacité en gagnant en autonomie. Par exemple les robots ménagers, *Roomba*, utilisent des techniques *SLAM* pour cartographier des intérieurs et s'y localiser. Des robots d'exploration spatiale en font également usage dans des environnements incomparablement plus vastes. Récemment des résultats impression-

nants ont été obtenus pour des applications à des véhicules autonomes. Ces systèmes mobiles complexes présentent toutefois l'avantage d'évoluer à hauteur constante ou presque. Les robots aériens, les quadricoptères par exemple, ne possèdent pas cette caractéristique simplificatrice. Généralement les applications robotiques ne sont pas réduites à l'usage d'une caméra, longtemps la télédétection par laser, *LIDAR* a été un moyen privilégié de percevoir l'environnement.

D'autres applications sont plus éloignées de la robotique. Pouvoir cartographier un environnement seulement en s'y déplaçant a des applications naturelles en architecture, en urbanisme et peut servir à construire des systèmes d'information géographique, *SIG*. Une autre famille d'applications particulièrement enthousiasmante tient de la réalité augmenté, où l'on sur-imprime des éléments virtuels au réel. L'intérêt de faire de la réalité augmentée dans le cadre du *SLAM* est que le système est capable de se localiser dans la carte de l'environnement. Cela offre la possibilité d'obtenir un rendu des éléments virtuels en accord avec les mouvements de l'utilisateur. Ainsi le cinéma peut bénéficier de techniques *SLAM* pour ajouter des éléments de décors ou des effets spéciaux. Des jeux vidéos en environnements réels deviennent réalisables. Des visites intéractives de musées, monuments historiques, villes bénéficieraient également grandement de l'utilisation de techniques *SLAM*. Enfin, l'assistance aux travailleurs effectuant des tâches de précision ou répétitives, induisant une baisse de vigilance, ou bien l'assistance à l'apprentissage comptent encore parmi les champs d'application du *SLAM*. Ces applications reposent beaucoup sur l'utilisation de caméras, dont l'exploitation a principalement été dévelopée par la communauté de vision par ordinateur et avant elle la communauté de photogrammétrie.

Dans le cadre de cette thèse nous nous sommes intéressés au *SLAM* dans une optique d'applications réalistes de réalité augmentée. Bien que le sujet ait été beaucoup exploré et que d'intéressants résultats aient été obtenus, la tâche n'est toujours pas parfaitement résolue. Le problème du SLAM est un sujet de recherche ouvert, aussi bien sur des aspects spatiaux (dérive, fermeture de boucle) que temporels (temps de traitement). De manière générale, nous nous sommes intéressés à la question de la précision des estimations.

Dans le cadre du *SLAM* monoculaire nous avons principalement adressé le problème de la dérive du système. Dans le cadre du *SLAM* multi-capteurs nous avons adressé le problème des mouvements de rotation difficilement gérables dans le cas monoculaire, et celui de la complexité calculatoire.

Résumé des chapitres

Ce manuscrit est organisé en 5 chapitres et termine sur quelques perspectives d'approfondissement des travaux présentés.

Chapitre 1. Le premier chapitre présente les trois formalismes les plus couramment employés pour résoudre le problème du *SLAM*. Nous y dressons également un état de l'art des travaux portant sur le *SLAM* monoculaire, le *SLAM* multi-capteurs et le *SLAM* bénéficiant d'informations *a priori* sur l'environnement.

Chapitre 2. Le deuxième chapitre présente les outils fondamentaux à l'élaboration d'un *SLAM* visuel. Nous y présentons notamment des méthodes de détection de points d'intérêt, d'estimation de pose, de triangulation de points, de robustification et de minimisation d'erreur.

Chapitre 3. Le troisième chapitre décrit les différents capteurs, caméras couleur ou profondeur et centrale inertielle, que nous avons utilisés. Les procédures de calibrations que nous avons employées sont y aussi présentées.

Chapitre 4. Le quatrième chapitre présente nos travaux sur le *SLAM* monoculaire. Nous présentons la méthode dont nous nous sommes inspirés, puis notre adaptation. Nous montrons ensuite deux applications de notre programme et une évaluation de son comportement sur une séries de séquences correspondant à des mouvements *canoniques*. Nous présentons finalement une étude d'une utilisation de l'ajustement de faisceaux pour réduire la dérive de poses de caméra de l'historique après une relocalisation du système.

Chapitre 5. Le cinquième chapitre présente nos travaux sur le *SLAM* multi-capteurs. Nous présentons en premier lieu différentes modalités de prise en compte d'un capteur de profondeur, et illustrons son intérêt au travers d'une application de réalité augmentée. Nous procédons ensuite à l'évaluation des différentes modalités sur les séquences de mouvements *canoniques*. Nous devisons suite aux résultats obtenus un schéma de *SLAM* multi-capteur allégé. Nous devisons également un schéma de *SLAM* hybride tirant optionnellement partie des données de profondeur. Enfin, nous présentons une utilisation de la centrale inertielle accélérant le processus et améliorant la précision des estimations.

Concepts

SLAM :
Le *SLAM* est le nom donné à la capacité d'un système mobile de se localiser dans un environnement inconnu et à en effectuer une reconstruction. Cette localisation et cartographie est effectuée au fur et à mesure de l'exploration.

Structure From Motion :
Le *Structure From Motion* est une méthode de localisation de plusieurs caméras et cartographie de la scène qu'elles observent. Historiquement développée dans un cadre non temps réel ces méthodes traitent les caméras par paquets et nécessitent un temps de calcul relativement important. Récemment des applications au cadre temps réel du *SLAM* les ont adaptées pour les appliquer de manière incrémentale.

Ajustement de faisceaux :
Composante essentielle des méthodes *Structure From Motion* et *SLAM-SFM*, l'ajustement de faisceaux est le nom donné à l'application de méthodes de minimisation d'erreur par moindres carrés non linéaires au cadre de la vision par ordinateur.

Filtrage de Kalman :
Le filtrage de Kalman est une méthode d'estimation de paramètres cachés indirectement inférables par le biais d'observations. Dans le cadre du *SLAM* elle est utilisée pour déterminer les paramètres de pose de la caméra et les positions de points de la carte de l'environnement. Cette méthode procède à chaque instant en deux étapes. Dans un premier temps la pose est prédite à partir de la pose à l'instant précédent. Dans un second temps elle est mise à jour en fonction des observations. La méthode fait notamment l'hypothèse que la loi gouvernant le mouvement du système est linéaire. Ce qui n'est pas notre cas. On appelle filtre de Kalman *étendu*, abrégé *EKF*, l'extension du filtre aux cas où les équations régissant les étapes d'observation et prédiction ne sont pas linéaires. Elles sont alors linéarisées par un développement en série de *Taylor*.

Filtrage particulaire :
Le filtrage particulaire est une solution inspirée de l'*EKF*. C'est une méthode de Monte-Carlo qui simule plusieurs estimations par *EKF* au travers de par-

ticules. Cette méthode a pour avantage d'offrir une plus grande robustesse aux erreurs d'estimation d'état.

Pose d'une caméra :
La pose d'une caméra représente à la fois sa position dans l'espace et son orientation.

Caméra :
Au delà du capteur, on utilisera généralement le terme de caméra pour désigner à la fois la pose d'une caméra et l'image enregistrée associée.

Caméra clé :
La notion de caméra clé a été introduite dans [E. Royer, 2005]. Une caméra devient une caméra clé lorsque le nombre de points appariés entre celle qui la suit et la dernière caméra clé est inférieure à un seuil, M, prédéterminé. Il n'existe malheureusement à notre connaissance pas de moyen rigoureux de définir ce seuil. L'ensemble des caméras clés définit un sous-ensemble de toutes les caméras représentant un squelette de la trajectoire de la caméra. L'idée est que ces caméra soient suffisament densément liées entre elles dans le but d'alléger la quantité de calculs tout en maintenant assez d'information pour que l'estimation de la trajectoire soit précise.

Triangulation :
Le processus de triangulation d'un point est un moyen par lequel est calculé la position dans $\mathbb{R}^3$ d'un point observé dans deux caméras dont les poses sont connues.

Points d'intérêts :
Les points d'intérets correspondent à des pixels ou zones d'images considérés comme caractéristiques. C'est à partir de ces points et de leur mise en correspondence que l'on cherche à résoudre le problème du *SLAM*.

Inliers/outliers :
Généralement employés au sujet d'appariements de points d'intérêt, les *inliers* sont les points correctement mis en correspondance. Les *outliers* réfèrent à des mises en correspondance erronées.

Dérive :
La précision des machines de calcul, des capteurs ainsi que celle des modélisations mises en oeuvre étant finies, chaque estimation d'état est toujours au mieux légèrement erronée. Le *SLAM* étant un processus incrémental, les erreurs de chaque étapes s'accumulent et conduisent le système à dériver.

Fermeture de boucle :
On appelle boucle une nouvelle visite d'un lieu déjà cartographié. Une conséquence de la dérive est que, pour des déplacements suffisamment larges, une boucle du système réel peut ne pas être modélisée comme telle. La tâche de fermeture boucle consiste alors à détecter ces cas et de corriger les estimations d'états lorsque nécessaire.

Notations

Scalaires et vecteurs $\in \mathbb{R}^2$: en minuscules
Matrices et vecteurs $\in \mathbb{R}^3$: en majuscules

X	:	point de la carte
j	:	indice de point, noté en lettre souscrite X_j
C	:	matrice de pose de la caméra
P	:	matrice de pose de la caméra, lorsque C peut prêter à confusion
i, k	:	indices de caméra, notés en exposant C^i
O^i	:	centre optique de la caméra C^i
C_k^i	:	matrice de pose relative entre les caméras C^k et C^i
X_j^i	:	point j dans le repère caméra i
m	:	mesure image d'un point
m_j^i	:	mesure de X_j dans C^i
z	:	mesure de profondeur
E	:	matrice essentielle
$T_\times$	:	matrice de préproduit vectoriel associée au vecteur T

Chapitre 1

Formalismes majeurs et état de l'art

La tâche du *SLAM* monoculaire est ardue. Historiquement, les premières recherches en robotique adressaient individuellement les tâches de localisation et cartographie. Et pour cause, une carte ne peut être reconstruite que si la pose du robot est connue et la pose du robot ne peut être obtenue que si l'on dispose d'une carte précise. Pourtant, le *SLAM* s'attelle à effectuer ces processus simultanément. Le problème est difficile car une mauvaise estimation de la pose résulte en la création d'une mauvaise carte qui a son tour corrompt l'estimation de la pose.

Dans ce chapitre nous présentons trois formalismes majeurs pour la résolution du *SLAM* visuel. Nous commençons par présenter les principes de l'approche par filtre de *Kalman*, puis de l'approche par filtrage particulaire, et enfin de l'approche *Structure From Motion*. Nous dressons un état de l'art des travaux faisant usage de ces méthodes, principalement dans le cadre du *SLAM* monoculaire. Nous passons ensuite en revue divers travaux de *SLAM* visuel multi-capteurs et de *SLAM* visuels faisant usage d'informations sur l'environnement connues *a priori*.

1.1 Slam basé sur filtre de Kalman étendu

Les premiers travaux portant sur le *SLAM* ne prenaient pas en compte la corrélation entre les tâches d'estimation de la pose de la caméra et de la construction de la carte. Les premiers résultats de bonne qualité furent ob-

tenus lorsqu'elle le fut. La première façon de le faire fut basée sur le filtrage de Kalman étendu. Une introduction à l'utilisation du filtre de Kalman pour résoudre le problème du *SLAM* est donnée dans [Durrant-Whyte and Bailey, 2006].

1.1.1 Théorie

Le filtre de Kalman est une méthode de filtrage statistique dont l'objectif est d'estimer itérativement l'état d'un système selon un flux de mesures. Dans son formalisme le filtre de Kalman fait l'hypothèse de systèmes linéaires et d'erreurs gaussiennes. On appelle filtre de Kalman étendu, *EKF*, l'application d'un filtre de Kalman à un système linéarisé par développement limité.

Un filtre de Kalman s'appuie sur des informations *a priori*, l'état estimé précédent, et une phase de prédiction. L'information *a priori* simplifie grandement le problème à la fois en permettant de restreindre l'espace de recherche de solutions et en diminuant d'autant la quantité d'observations nécessaires à l'obtention d'une solution. La phase de prédiction peut améliorer l'information *a priori* s'il est possible d'obtenir des informations sur les changements d'état du système, comme des commandes de contrôle, des données inertielles ou un modèle de mouvement.

On note :

- C^k le vecteur d'état décrivant la k-ième pose de la caméra
- u^k le vecteur de contrôle appliqué à l'instant $k-1$
- $u = u^{0:k} = \{u^0, u^1, ..., u^k\}$ l'historique de toutes les commandes de contrôle
- X_j le vecteur de position réelle du j-ème point de la carte
- $X = X_{0:m} = \{X_0, X_1, ..., X_m\}$ l'ensemble de tous les points de la carte
- On note m_j^k l'observation du point j dans la caméra k
- $m^{k:l}$ les observations des caméras $k+1$ à l
- $m = \{m_j^i\}_{j=0...}^{i=0...}$ l'ensemble des observations
- $A^k = \left(C^k, X\right)$ l'état du système à la k-ème caméra

On considère état et mesures comme des variables aléatoires. L'estimation de l'état du système A^k peut être réalisée par estimation bayésienne récursive grâce à la formule :

$$p(A^k|m^k) = \frac{p(m^{k-1:k}|A^k) \cdot p(A^k|m^{k-1})}{p(m^{k-1:k}|m^{k-1})} \tag{1.1}$$

Le filtre de Kalman procède en deux étapes. Dans un premier temps, la pose est prédite à l'aide des commandes de contrôle et/ou d'un modèle

de mouvement. Dans un second temps elle est mise à jour grâce aux observations. Dans les deux cas des estimations de l'état A^k sont calculées ainsi qu'une matrice K de covariance entre les points et la caméra. L'état du système est ainsi représenté par une variable aléatoire suivant une loi normale $\mathcal{N}(A, K)$. La matrice K est de la forme :

$$K = \begin{pmatrix} K_{cc} & K_{cx} \\ K_{cx}^\top & K_{xx} \end{pmatrix} \tag{1.2}$$

La prédiction de pose se fait à l'aide d'une fonction $f(.)$ modélisant la cinématique et en prenant en compte un bruit gaussien w^k de loi $\mathcal{N}(0, Q^k)$:

$$C^k = f(C^{k-1|k-1}, u^k) + w^k$$

Le modèle d'observation prend également en compte un bruit gaussien v^k de loi $N(0, R^k)$:

$$m^k = h(A^k) + v^k$$

Pour des raisons de performance la formulation standard du *SLAM* basé sur filtrage de Kalman ne suit que la dernière pose de la caméra et non la trajectoire. Le système est estimé comme :

$$\begin{pmatrix} \hat{C}^{k|k} \\ \hat{X}^k \end{pmatrix} = E\left[\begin{pmatrix} C^k \\ X \end{pmatrix} | m^{0:k} \right] \tag{1.3}$$

Avec la matrice de covariance :

$$K^{k|k} = E\left[\begin{pmatrix} C^k - \hat{C}^k \\ X - \hat{X}^k \end{pmatrix} \cdot \begin{pmatrix} C^k - \hat{C}^k \\ X - \hat{X}^k \end{pmatrix}^\top | m^{0:k} \right] \tag{1.4}$$

On présente maintenant les différentes étapes du filtre.

Initialisation
La pose est d'abord initialisée à une valeur $C^{0|0}$ et la matrice de covariance à $K^{0|0}$.

Prédiction de l'état
A l'instant k on prédit la pose de la caméra :

$$\hat{C}^{k|k-1} = f(\hat{C}^{k-1|k-1}, u^k)$$

La matrice de covariance est mise à jour selon :

$$K_{cc}^{k|k-1} = \nabla f \cdot K_{cc}^{k-1|k-1} \cdot \nabla f^{\top} + Q^k$$

Où ∇f est la jacobienne de $f(.)$ évaluée au point $\hat{C}^{k-1|k-1}$. Si les points X^{k-1} sont fixes, il n'est pas nécessaire de prédire leur état. On a alors :

$$A^{k|k-1} = \begin{pmatrix} C^{k|k-1} \\ X^{k-1} \end{pmatrix}$$

Prédiction de la mesure
Les mesures sont prédites par :

$$m^{k|k-1} = h(A^{k|k-1})$$

Le vecteur

$$m^k - m^{k|k-1}$$

est appelé *l'innovation* et représente l'écart entre les prédictions et les observations. La covariance de l'innovation est représentée par S^k :

$$S^k = \nabla h \cdot K^{k|k-1} \cdot \nabla h^{\top} + R^k$$

Avec ∇h la jacobienne de $h()$ évaluée en $A^{k|k-1}$.

Mise à jour de l'état
L'estimation du système est finalement mise à jour selon :

$$\begin{pmatrix} \hat{C}^{k|k} \\ \hat{X}^k \end{pmatrix} = \begin{pmatrix} \hat{C}^{k|k-1} \\ \hat{X}^{k-1} \end{pmatrix} + W^k \cdot \left(m^k - m^{k|k-1}\right)$$

et

$$K^{k|k} = K^{k|k-1} - W^k \cdot S^k \cdot {W^k}^{\top}$$

avec

$$W^k = K^{k|k-1} \cdot \nabla h^{\top} \cdot {S^k}^{-1}$$

la *matrice de gain* qui sert à pondérer la différence entre les prédictions et les observations.

Conceptuellement, cette méthode pondère donc les observations par la dynamique et inversement. Elle permet également de représenter les incertitudes relatives. Il est fréquent de bien connaître les localisations relatives d'ensembles de points de la carte sans connaître avec précision leurs positions absolues.

1.1.2 Littérature

Le premier travail prenant en compte la corrélation entre la pose de la caméra et l'estimation de la carte fut probablement celui de Smith [Smith et al., 1988]. Dans cet article Smith *et al* introduisent la représentation probabiliste de la carte et la première formulation récursive du problème en modélisant le système, caméra et carte, selon une unique loi normale multidimensionnelle. Ils utilisent donc un filtre de Kalman et observent en particulier que si les mesures ne sont pas linéaires en pratique les résultats obtenus sont raisonnables que sous l'hypothèse d'un modèle linéaire, le *SLAM* basé sur filtre de Kalman atteint l'estimation optimale, observation confirmée dans la thèse de Newman [Newman, 1999].

Cette méthode a par la suite durablement influencé les recherches sur le *SLAM*. Dans les années 1990 le filtre de Kalman étendu est devenu l'approche standard pour la résolution du *SLAM*, voir Leonard *et al* [Leonard and Durrant-Whyte, 1991], Betgé-Brezetz *et al* [Betge-Brezetz et al., 1996], Newman [Newman, 1999].

Son intérêt a été démontré au cours de nombreuses expériences réelles et simulées, et ses faiblesses identifiées.

Premièrement, la complexité calculatoire de la méthode croît quadratiquement avec le nombre de points de la carte. Bien que plusieurs stratégies pour contourner ce problème aient été proposées, voir la section 4.6, cela réduit fortement la taille des environnements dans lesquels elle est applicable. Deuxièmenement, la linéarisation de fonction non linéaires (en particulier les mesures angulaires) peut conduire à des estimations erronnées. Dans [Julier and Uhlmann, 2001] Julier et Uhlmann démontrent que sous certaines conditions l'*EKF-SLAM* est condamné à diverger. Ils montrent en particulier que la stationarité du système conduit le filtre à avoir une confiance trop importante dans l'état estimé.

Dans [Castellanos et al., 2004] Castellanos *et al* montrent que les incohérences sont liées à l'incertitude du filtre. Ils illustrent le phénomème en montrant que des estimations incohérentes surviennent plus rapidement si l'état initial est incertain. Ils préconisent alors d'estimer le système en considérant l'état initial certain.

Dans MonoSLAM [Davison, 2003], étendu dans [A.J. Davison and Stasse, 2007], Davison utilise un *EKF* pour réaliser l'un des premiers systèmes *SLAM* monoculaire en temps-réel. L'étape de prédiction est réalisée en utilisant un modèle de mouvement à vitesse constante.

Dans [Thrun et al., 2004] Thrun *et al* adressent le problème du *SLAM*

en utilisant un filtre d'information, c'est à dire un filtre de Kalman basé la matrice d'information, l'inverse de la matrice de covariance. En normalisant les coefficients de la matrice ils observent que la majorité des coefficients sont proches de zéro. La complexité calculatoire est alors drastiquement réduite en considérant ces coefficients comme nuls. Leur méthode permet de résoudre le problème du *SLAM* en temps constant, mais avec des résultats moins précis que ceux d'un *EKF*.

Dans une adaptation de MonoSLAM, Holmes *et al* [Holmes et al., 2008] comparent filtre de Kalman étendu, filtre de Kalman *unscented* et leur version optimisée de ce dernier. Les filtres de Kalman *unscented* sont des variantes du filtre conçues afin de mieux prendre en compte des fonctions de prédiction et observation non linéaires. D'après leurs expériences les auteurs observent que les estimations obtenues par *UKF* sont plus justes mais que les coût calculatoire est plus de dix fois supérieur. Ils concluent que ce surcoût calculatoire ne permet pas de considérer l'approche *UKF* plus intéressante que l'*EKF*.

Dans [Servant, 2009], Servant adapte un filtre de *Kalman* au cas du *SLAM* monoculaire basé sur un suivi de structures planaires.

1.1.3 Qualités et défauts

La première qualité du filtrage de Kalman appliqué au *SLAM* est que le filtre se prête naturellement à la fusion de données.

Par contre, le coût calculatoire de chaque étape est important. L'estimation de la matrice de covariance le fait croître quadratiquement avec le nombre de points de la carte. En outre, la méthode est peu robuste, des associations incorrectes peuvent rapidement perdre le système. Enfin, elle est particulièrement sujette au phénomène de dérive : la linéarisation des équations régissant les étape de prédiction et d'observation introduit des erreurs s'accumulant au cours du temps.

Une solution pour résoudre le problème de la robustesse aux mauvais appariements et prendre en compte des incertitudes plus générales que simplement gaussiennes est le filtrage particulaire.

1.2 Slam basé sur filtrage particulaire

Le filtrage particulaire est une autre solution populaire. Inspirée du filtrage de Kalman, c'est une méthode de Monte-Carlo qui simule plusieurs estimations par *EKF* et offre ainsi une certaine robustesse aux erreurs d'estimation d'état. Cette solution a été introduite par l'article [Montemerlo et al., 2002] puis améliorée dans [Montemerlo et al., 2003] . Estimer la trajectoire du système et les points de la carte par filtrage particulaire serait extrêmement coûteux, alors l'approche *FastSLAM* adopte un filtrage particulaire *Rao-blackwellisé*. Le principe est que, si l'historique des poses de la caméra est parfaitement connu, les observations des points de la carte sont indépendantes. Ainsi leur estimation peut être réalisée en dehors du filtre particulaire. La pose du système est représentée par n_c particules et à chaque particule est associée une carte, soit n_m points. Chaque point de la carte est estimé par un filtre de Kalman très simple. Le filtre nécessite un modèle de mouvement pour faire évoluer les particules, on le nomme $\pi()$.

Le *SLAM* basé sur filtrage particulaire cherche à estimer :

$$\begin{aligned} p(C^{0:k}, X|m^{0:k}, u^{0:k}, c^0) &= p(C^{0:k}|m^{0:k}, u^{0:k}, c_0) \cdot p(X|C^{0:k}, m^{0:k}) \\ &= p(C^{0:k}|m^{0:k}, u^{0:k}, c^0) \cdot \prod_j^{n_m} p(X_j|C^{0:k}, m^{0:k}) \end{aligned} \tag{1.5}$$

Chaque particule est dotée d'un poids qui caractérise sa vraisemblance. On note $w^{k,i}$ le poids associé à la i-ème particule. A l'instant k l'état du système est estimé par l'ensemble des particules :

$$\{w^{k,(i)}, C^{0:k,(i)}, p(X|C^{0:k,(i)}, m^{0:k})\}_{i=1}^{n_c}$$

Initialisation

Comme pour l' *EKF-SLAM* la pose des particules est initialisée à la pose identité.

Mise à jour des particules

A chaque instant, la pose de la caméra de chaque particule mise à jour.

– Dans la première formulation du *FastSLAM* la pose est prédite en se reposant seulement sur le modèle de mouvement :

$$C^{k,(i)} \sim p(C^k|C^{k-1,(i)}, u^k) \sim \pi(C^k|C^{0:k-1,(i)}, u^k)$$

– Dans la seconde formulation les observations sont prises en compte :

$$C^{k,(i)} \sim \left(p(C^k | C^{0:k-1,(i)}, m^{0:k}, u^k) = \frac{1}{c} \cdot p(m^k | C^k, C^{0:k-1,(i)}, m^{0:k-1}) \cdot p(C^k | x^{k-1,(i)}, u^k) \right)$$

Où c est une constante de normalisation.

Mise à jour des poids
Le poids est re-calculé selon :

$$w^{k,(i)} = w^{k-1,(i)} \frac{P(m^k | C^{0:k,(i)}, m^{0:k-1}) P(C^{k,(i)} | C^{k-1,(i)}, u^k)}{\pi(C^{k,(i)} | C^{0:k,(i)}, m^{0:k}, u^k)}$$

L'ensemble des particules évolue avec le temps. Selon des critères empiriques l'ensemble est régulièrement ré-échantillonné. Les particules affectées sont choisies aléatoirement avec une probabilité fonction de leur poids. Après ré-échantillonnage les particules sont toutes de même poids $\frac{1}{n_c}$. Puis, pour chaque particule, les *EKF* des points observés sont mis à jour à partir de la position de la caméra.

1.2.1 Qualités et défauts

Le principal défaut de cette solution réside dans l'accentuation du phénomène des dérive, en effet en plus de la linéarisation des équations identique au cas de l'*EKF* la multiplicité des particules utilisées pour représenter la pose du système favorise le phénomène. Cette multiplicité de particules complique également la tâche de fermeture de boucle.

Selon Durrant-Whyte et Bailey dans [Durrant-Whyte and Bailey, 2006] cette méthode ne permet pas la fermeture de boucles.

La grande qualité du filtrage particulaire est de pouvoir représenter n'importe quelle distribution de probabilité en tant que somme de gaussiennes. Cela confère une intéressante robustesse aux estimations erronnées.

1.3 Slam basé ajustement de faisceaux

Une autre formalisation populaire du *SLAM* repose sur l'adaptation des méthodes *Structure From Motion* au temps réel. Nous cherchons toujours à estimer en continue la pose d'une caméra et la structure de la scène filmée.

Dans cette approche l'estimation de l'historique des poses, la trajectoire, est un élément crucial.

On note :
- $A = \{C^0, \ldots, C^n, P_0, \ldots, P_m\}$ l'état du système
- $a_j^i = (C^i, P_j)$ le couple formé de la i-ème caméra et du j-ième point
- $h(a_j^i)$ la fonction de projection du point P_j dans la caméra C^i
- $m = \{m_j^i\}_{j=0\ldots m}^{i=0\ldots n}$ l'ensemble des observations

On fait l'hypothèse que le bruit de mesure est gaussien, ce qui signifie :

$$p(m_j^i|a_j^i) = \frac{1}{\sigma_j^i\sqrt{2\pi}} e^{-\frac{1}{2}\cdot\left(\frac{m_j^i - h(a_j^i)}{\sigma_j^i}\right)^2} \tag{1.6}$$

On cherche à estimer la trajectoire la plus vraisemblable donc à maximiser la probabilité *a posteriori* de la trajectoire selon les observations :

$$p(A|m) \tag{1.7}$$

D'après le théorême de Bayes on sait que :

$$p(A|m) = \frac{p(m|A)\cdot p(A)}{p(m)} \tag{1.8}$$

Dans le cas général aucune connaissance a priori n'est disponible sur $p(A)$ et $p(m)$. Maximiser 1.7 revient alors à maximiser la vraisemblance des observations :

$$p(m|A) \tag{1.9}$$

Si une information a priori est disponible sur A. Maximiser 1.7 revient à maximiser :

$$p(m|A)\cdot p(A) \tag{1.10}$$

On fait l'hypothèse d'indépendance des erreurs de mesures, alors la vraisemblance des observations se factorise comme :

$$p(m|A) = \prod_{i=0,j=0}^{n,m} p(m_j^i|a_j^i) \tag{1.11}$$

Maximiser l'équation 1.9 revient à maximiser sa *log-vraisemblance* ou minimiser l'opposé de cette dernière. Sous l'hypothèse que l'erreur de reprojection des points suit une loi normale $\mathcal{N}(m_j^i, {\sigma_j^i}^2)$, c'est équivalent à minimiser

la fonction $F(.)$ suivante :

$$F(A) = \sum_{i=0,j=0}^{n,m} \left(\frac{m_j^i - h(a_j^i)}{\sigma_j^i} \right)^2 \tag{1.12}$$

On peut observer que $F(A)$ suit une loi du χ^2 à $\sim m+n$ degrés de libertés. Usuellement les erreurs sont représentées par un vecteur de résidus $\Delta(A)$:

$$\Delta(A) = \begin{pmatrix} \dots \\ \Delta_j^i(A) \\ \dots \end{pmatrix} \text{avec} \Delta_j^i(A) = m_j^i - h(a_j^i) \tag{1.13}$$

Minimiser $F(A)$ est équivalent à minimiser $\chi^2_\Sigma(A)$:

$$\chi^2_\Sigma(A) = \Delta(A)^\top \cdot \Sigma^{-1} \cdot \Delta(A) \tag{1.14}$$

Où $\Sigma = \text{diag}(\dots, {\sigma_j^i}^2, \dots)$ est la matrice diagonale composée des variances des mesures. Dans le cas général les variances des mesures ne sont pas accessibles. On considère alors qu'elles sont toutes égales, minimiser $\chi^2_\Sigma(A)$ est équivalent à minimiser $f(.)$:

$$f(A) = \Delta(A)^\top \cdot \Delta(A) \tag{1.15}$$

La résolution du *SLAM-SFM* se réalise en minimisant la fonction de coût $f(.)$ correspondant à la somme des carrés des erreurs de reprojection. Cette minimisation est réalisée par ajustement de faisceaux. Comme indiqué dans la partie A l'existence de minima locaux nécessite une initialisation des paramètres de bonne qualité.

1.3.1 Littérature

Les articles [Nistér et al., 2004], puis [Nistér et al., 2006], comptent parmi les travaux pionniers de l'application des méthodes *Structure From Motion* au cas temps réel et incrémental du *SLAM*. Dans ces articles les auteurs utilisent un ajustement de faisceaux local : une fenêtre glissante sur les trois dernières poses de caméras qui sont raffinées par ajustement de faisceaux. Cette approche est particulièrement sujette au phénomène de dérive.

Dans [Mouragnon et al., 2006], puis [Mouragnon et al., 2009], Mouragnon *et al* introduisent la notion de *keyframe*, un sous ensemble des poses et

images de la caméra. Ils généralisent le principe de la fenêtre glissante pour l'ajustement de faisceaux en l'appliquant uniquement sur un nombre fixe des dernières *keyframes*. Les plus anciennes étant supposées suffisamment optimisées elles ne sont pas ajustables dans le processus d'optimisation et servent ainsi à contrebalancer les erreurs éventuellement introduites dans le cas d'une mauvaise estimation de la pose des caméras les plus récentes. Cette approche est sensiblement moins sujette au phénomènene de dérive que la précédente mais nécessite une intialisation dont la qualité impacte grandement celle de la suite de la méthode. L'estimation de pose d'une nouvelle caméra est basée sur des correspondances *2D-3D* et ne tire pas partie de l'historique des poses ce qui en affaiblit la précision.

Dans [G. Klein, 2007], Klein *et al* reprennent la notion de *keyframes* dans une application destinée à la réalité augmentée. Une estimation précise de la pose est alors nécessaire pour chaque nouvelle caméra. Pour y parvenir les auteurs utilisent un modèle de mouvement pour faire évoluer la pose du système et cette prédiction est raffinée dans un processus d'optimisation non linéaire n'impliquant que les observations de la dernière caméra et les positions dans l'espace des points déjà triangulés. La notion d'ajustement de faisceaux local est reprise à la détection de chaque nouvelle *keyframe*. Les auteurs ajoutent également un ajustement de faisceaux global traité en parallèle. Cette approche est ainsi moins sujette au phénomène de dérive que les précédentes.

Dans [Hartley and Schaffalitzky, 2004] Hartley et Schaffalitzky proposent de résoudre le problème *Structure From Motion* en minimisant la norme L_∞ en lieu de la norme L_2 usuelle. Les auteurs montrent que l'usage de cette norme rend la fonction de coût optimisée convexe et qu'ainsi existe un seul minimum au lieu de plusieurs minima locaux. L'inconvénient est que cette méthode nécessite que les données ne soient contaminées par aucun *outlier*.

Dans [Strasdat et al., 2010], Strasdat *et al* effectuent une comparaison entre les approches filtrées et *SFM*. Ils prennent pour représentant des solutions filtrées celle de Eade introduite dans [E. Eade, 2007] et pour représentant des solutions *SFM* le *PTAM* de Klein. Selon les auteurs, ces solutions sont assez similaires car les deux utilisent des processus parallèles pour réaliser localisation et optimisation de la carte et réalisent les fermetures de boucles selon des approches visuelles. Les différences résident dans la façon dont la carte est construite. Le critère de comparaison employé porte sur sur la mesure du rapport entre la précision de la localisation et la vitesse d'exécution des méthodes. En partant d'une carte initialisée les auteurs étudient trois

mouvements types. Le premier en translation le long d'un plan, le second en rotation autour de l'axe de visée de la caméra face à un plan, le troisième en translation le long de l'axe de visée entre deux plans dans une configuration de type couloir. Pour chacun de ces trois mouvements les auteurs étudient la façon dont les erreurs de localisation évoluent selon l'augmentation du nombre de caméras clés et de points observés. Pour ces trois mouvements les résultats sont très similaires et montrent clairement qu'il est préférable d'augmenter le nombre de points observés plutôt que le nombre de points de vue pour améliorer la précision des estimations. En notant N le nombre de caméras clé et M le nombre de points de la carte, selon le coût d'une solution filtrée est au mieux de l'ordre de $O(M^2)$ et de l'ordre de $O(N^2 \cdot M)$ pour une approche ajustement de faisceaux. Il apparaît ainsi que l'augmentation du nombre de points de la carte permettant d'améliorer la qualité des estimations est beaucoup mieux amortie par les approches *SLAM-SFM* que filtrées. Toutefois, les auteurs remarquent que dans le cas de ressources calculatoires très limitées le filtrage peut s'avérer plus intéressant que l'ajustement de faisceaux.

Dans [Strasdat, 2012] Strasdat étend le formalisme de représentation de la pose de la caméra en y incorporant le facteur d'échelle. Cela lui permet de prendre en compte la dérive du facteur d'échelle lors de la correction de la trajectoire après une détection de fermeture de boucle. Les résultats obtenus sont considérablement améliorés.

1.4 Approches multi-capteurs

La tâche du *SLAM* visuel peut être simplifiée en tirant partie d'informations supplémentaires. Une manière de le faire consiste à employer des capteurs supplémentaires, souvent des centrales inertielles, ou *GPS* ou capteurs de profondeurs.

1.4.1 Capteurs stéréo, inertiels et positionnels

Dans [Jung and Lacroix, 2003] Jung et Lacroix utilisent un *EKF-SLAM* pour un système de stéréo-vision. Le capteur stéréo est utilisé pour obtenir une carte dense de profondeur. Alors l'étape de prédiction est réalisée par odométrie visuelle.

Dans [M. Aron, 2004] Aron *et al* supposent la scène planaire par morceaux. La localisation est effectuée grâce à suivi hybride par une caméra et une centrale inertielle. La contribution plus notable réside de la prédiction de position des points d'intérêts d'une caméra à l'autre à partir des données de la centrale inertielle : en utilisant la matrice de covariance de la centrale, ils peuvent prédire l'ellipse d'incertitude autour des positions prédites et ainsi de limiter la recherche de points à apparier.

Dans l'article [Pollefeys et al., 2008] Akbarzadeh *et al* proposent une méthode de *SLAM* multi-capteurs multi-paradigmes. Cette approche se base principalement sur un filtrage de *Kalman* nourrit des informations visuelles, inertielles et géo-référentielles. Lorsque les informations inertielles et référentielles ne sont pas accessibles, l'algorithme opte pour une approche *SLAM-SFM* [Nistér et al., 2006]. Les expérimentations conduites sur un véhicule muni de quatres caméras sur chaque flanc, une centrale inertielle et un GPS montrent que l'algorithme est temps-réel et fournit des estimations d'excellente qualité sur de longs trajets. Les cartes des points obtenues comportant beaucoup de points (1000) et la méthode étant conçue pour tirer parti de la planarité des scènes urbaines conjuguée à la réduction des degrés de liberté induites par le véhicule, les plans au sol et de façade peuvent être efficacement estimés. La multiplicité des caméra permettant d'obtenir plusieurs cartes de points à chaque étape, un fusion de carte permet de rejeter des estimations erronées. Finalement, une carte dense est obtenue comme un maillage texturé issu d'une triangulation de Delaunay. Plusieurs parties de l'algorithme sont implémentées sur GPU pour assurer des performances temps réel. La fermeture de boucle est assurée par les contraintes issues des données inertielles et référentielles.

Dans [Servant, 2009], Servant montre que l'utilisation d'une centrale inertielle dans un *SLAM* filtré basé sur des structure planes apporte de la robustesse au flou de bougé et aux occlusions d'objets suivis, tout en amélioration la précision de la localisation de la caméra.

Dans [Michot, 2010], Michot montre la possibilité d'optimiser simultannément des erreurs de reprojections à d'autres types d'erreurs, notamment en prenant en compte des données inertielles, en adaptant dynamiquement les coefficients de pondération des différentes sources d'erreurs. Ces travaux sont particulièrement intéressants dans le cas où les variances des différentes sources ne pas connues *a priori* où si elles évoluent au cours du temps.

Dans [Scaramuzza et al., 2009], Scaramuzza *et al* développent un nouveau formalisme de *SLAM* pour une caméra montée sur un véhicule. Ils

nomment ce modèle "mouvement planaire circulaire". Il utilisés dans son cas une caméra omnidirectionnelle et l'information de vitesse du véhicule. Dans ce cas une pose relative est fonction de trois inconnues : deux pour la translation, exprimées en coordonnées polaires, et une pour l'angle paramétrisant la rotation. Les auteurs dérivent une expression simplifiée de la contrainte épipolaire et constatent que la direction de la translation peut être obtenue en résolvant une seule équation. Ainsi une seule paire de points est suffisante. C'est particulièrement intéressant lorsqu'appliqué dans un processus *RANSAC*, puisque la complexité du processus en est drastiquement réduite. Le formalisme est testé dans un environnment urbain très dynamique. La précision est moin bonne que pour des modèles de mouvement plus généraux. Cependant, il supporte mieux les situations critiques que ses concurrents.

Dans [Leutenegger et al., 2013] Leutenegger *et al* utilisent une centrale inertielle et un système stéréo-vision dans une approche *SLAM-SFM*. Ils adoptent une formulation relative du problème, et s'appuient sur une calibration très fine des capteurs visuels et inertiel. L'usage des données inertielles leur permet d'optimiser des caméras clés arbitrairement espacées en les contraignant selon ces mesures. Le système stéréo apporte la profondeur aux points d'intérêts. L'intégration des données inertielles permet alors de rejeter les appariements *outliers* simplement en les reprojetant. Les mesures inertielles et visuelles sont optimisées simultanément.

1.4.2 Capteur de profondeur

Dans [Henry et al., 2010] Henry *et al* présentent une méthode de *SLAM* utilisant un capteur de la société *PrimeSense*. L'estimation de chaque nouvelle pose de caméra est réalisée par appariement de points d'intérêts avec la dernière caméra clé. Les points d'intérêts sont ensuite augmentés de la profondeur mesurée correspondante et les appariements sont filtrés par une procédure *RANSAC* appliquée à une méthode d'estimation de pose *P3P*. Puis la pose est ajustée selon une méthode hybride entre une *ICP*, minimisant une distance point-plan, et une méthode de minisation de moindres carrés non linéaire éparse des distances des points d'intérêts appariés dans $\mathbb{R}^3$. La pondération des deux méthodes n'est pas formellement spécifiée. A chaque ajout de caméra clé, lorsqu'il y a trop peu d'appariements, une détection de boucle est effectuée en tentant d'apparier la caméra avec l'ensemble des caméras clés précédentes. Puis l'ensemble des caméras clés est optimisé avec la bibliothèque *TORO*, [Grisetti et al., 2009]. Les expériences montrent que

quelque soit la pondération adoptée lors de la minimisation d'erreur hybride, elle est toujours plus efficace que l'une ou l'autre des méthodes hybridées prise isolément. Cependant, le coût calculatoire est très important, de l'ordre de $500ms$ d'après les auteurs. Les expériences montrent une cartographie de bonne qualité sur plus d'une centaine de mètres. La détection de boucle à chaque ajout de caméra clé joue certainement un rôle important dans la qualité de ces résultats.

Dans [Fioraio and Konolige, 2011] Fioraio et Konolige présentent une méthode de *SLAM* avec *Kinect* basée sur une *ICP* selon une approche basée sur des caméras clés. Chaque nouvelle caméra est alignée avec la dernière enregistrée puis avec la dernière caméra clé. Des points d'intérêts peuvent être utilisés de manière optionnelle. S'ils sont utilisés lors de l'étape d'alignement ils servent à estimer une pose initiale, sinon celle-ci est considérée comme étant l'identité. Ensuite, pour chaque image environ 1000 points sont sélectionnés dans une grille régulière dans l'image de profondeur de chaque caméra. La pose initiale permet de créer des correspondances entre ces points d'une image à l'autre. Les correspondances sont sauvegardées si les différences de normales et distances ne sont pas trop importantes et qu'aucun des deux membres n'est associé à trop d'autres candidats. La pose est optimisée dans un graphe de contraintes, représentant les poses, en minimisant l'erreur euclidienne quadratique des correspondances. Trois stratégies sont possibles : soit point à point, soit point à plan, soit plan à plan. Si des points d'intérêts sont utilisés leurs erreurs de reprojections sont ajoutées. Les optimisations sont réalisées avec la bibliothèque logicielle g^2o, [Kuemmerle et al., 2011]. Les expérimentations montrent que les meilleurs résultats sont obtenus en minimisant les erreurs point-plan et en faisant usage de points d'intérêt. Les environnements testés sont de petites tailles, de l'ordre de quelques mètres.

Dans [Newcombe et al., 2011] Newcombe *et al* utilisent uniquement les données de profondeur d'une caméra *Kinect* se basant sur un usage intensif d'une carte graphique haut de gamme *GeforceGTX580* de *Nvidia* néanmoins raisonnablement chère. La carte de l'environnement est modélisée par une structure voxellique. Chaque voxel contient un poids et une *signed distance function*, *sdf*, tronquée, *tsdf* issue des travaux de la communauté de réalité virtuelle. Le poids correspond à une sorte d'indice de confiance en la mesure, il est défini par le cosinus de l'angle en la normale à la surface la plus proche et l'axe de visée de la caméra. Une *sdf* représente la distance signée d'un point de l'espace à la surface la plus proche. Une valeur de zéro signifie qu'un point est à la surface, positive que le point est au dessus de la surface, négative

que le point est en dessus. Dans le cas du *SLAM* il n'est pas possible de déterminer avec certitude qu'un point est sous la surface, la *tsdf* permet de représenter ces incertitudes. Dans le cas des voxels, les valeurs de *tdsf* sont définies par la moyenne pondérée des valeurs observées pour chaque caméra et les poids comme leur somme. L'algorithme est composé de quatre étapes. La première : pour chaque nouvelle caméra des sommets et normales sont extraits des mesures de profondeur, les trous sont comblés. La seconde : la carte est prédite selon une pose initiale, la précédente ou celle en cours de raffinement. Elle est effectuée par ray-casting dans la structure voxellique pour trouver les voxels proches de zéro. La troisième : la pose est estimée par une procédure d'*ICP* multi-échelle, trois, entre la caméra courante et la carte prédite. Cette étape est adaptée à l'hypothèse spécifique que les changements d'orientation sont minimes. Les paramètres optimaux sont déterminés par minimisation de moindres carrés non linéaires, les matrices sont formées par GPGPU. Les *outliers*, en profondeur, évidents ne sont pas pris en compte tant qu'ils ne constituent qu'une faible partie de l'observation de la caméra. La quatrième : la carte est fusionnée avec les mesures de la caméra lorsque sa pose est estimée, cette étape est réalisée par GPGPU. Un avantage très intéressant est la fusion naturelle des mesures et de la carte lors des fermetures de boucle tant que la pose estimée n'a pas trop dérivé. Selon les auteurs la méthode est efficace dans de petits environnements, de quelques mètres. Elle nécessite toutefois d'observer des variations de profondeur, une image constituée d'un large plan pouvant être un cas d'échec.

Dans [Endres et al., 2012] Endres *et al* utilisent une caméra *Kinect* et adoptent une approche inspirée de [G. Klein, 2007] : deux *threads* distincts sont chargés de l'estimation de la pose de la caméra et du raffinement en continu de la carte, basé sur l'utilisation de la bibliothèque g^2o. Les auteurs remarquent que les données de profondeur peuvent être tronquées le long d'arêtes d'objets. Alors ils ne les prennent en compte que couplées à des points d'intérêt (SIFT, SURF, ORB) et au travers d'un processus *RANSAC* d'estimation de pose *P3P*, [Haralick et al., 1994], dans le premier *thread*. Notons que l'image courante n'est pas appariée qu'avec la seule image clé précédente, mais avec les 3 images clés précédentes et 17 images clés sélectionnées uniformément. Les auteurs ne précisent pas de quelle forme est l'erreur minimisée dans le *thread* d'optimisation. La méthode est efficace sur plusieurs dizaines de mètres.

1.5 Approches utilisant une connaissance de l'environnement

Une autre manière de simplifier la tâche du *SLAM* visuel est de supposer une partie de l'environnement ou sa totalité déjà connue.

Dans [I. Gordon, 2006] Gordon et Lowe utilisent un modèle, construit hors-ligne, pour des applications de réalité augmentée. Leur localisation est presque temps réel. Le principe employé est celui du *SLAM-SFM* avec un ajustement de faisceaux bi-objectif afin de réduire l'effet de gigue sur la pose.

Dans [Sourimant et al., 2007], Sourimant *et al* utilisent un GPS, un système d'information géographique (GIS) et une caméra. Le GPS sert à initialiser la position dans le modèle. Celle-ci est corrigée manuellement par l'utilisateur, le modèle étant superposé à l'image. Elle est finalement raffinée en utilisant des correspondances $2D$-$3D$. En fin d'initialisation les modèles filaires sont texturés grâce aux points d'intérêt *KLT* extraits de la caméra. En conséquence les positions des points sont interpolées à partir des sommets des faces sur lesquelles ils reposent. La première image devient l'image de référence. Dans les images de caméra suivantes ces points sont suivis, mais pas extraits, et les poses sont estimées grâce aux appariements $3D$-$2D$ obtenus. Lorsque le nombre de points suivis devient trop faible, la dernière image devient l'image de référence. Des points en sont extraits et utilisés pour texturer le modèle.

Dans [Lothe et al., 2009] Lothe *et al* utilisent un modèle planaire grossier mais couvrant entièrement les scènes parcourues. La méthode agit en post-process. Elle se décompose en trois étapes. La première consiste à appliquer le *SLAM* de Mouragnon *et al* [Mouragnon et al., 2006]. Dans la seconde, des segments de trajectoire, comprenant des points et les caméras observatrices, sont identifiés. Ces segments sont utilisés dans une procédure d'ICP adaptée pour recaler la reconstruction sur le modèle. La dernière étape consiste en un ajustement de faisceaux. Constatant qu'une méthode classique peut perdre la correction apportée par l'ICP les auteurs modifient l'ajustement en reprojetant perspectivement les points 3D sur le modèle et en utilisant l'erreur de reprojection de ces nouveaux points dans la fonction de coût. Finalement, les points 3D sont corrigés par triangulation à partir des nouveaux paramètres de pose, ils ne reposent donc pas nécessairement sur le modèle. Les *outliers* 3D sont gérés par l'usage d'un M-estimateur. La méthode améliore grandement la fidélité au terrain par rapport à celle de [Mouragnon et al., 2006]. Elle n'est toutefois pas applicable en temps réel et se limite au post-processing.

Le modèle très précis obtenu peut être ultérieurement utilisé dans des tâches de navigation seule.

Dans [Lothe et al., 2010] Lothe *et al* reprennent l'idée précédente appliquée au temps réel. La localisation est effectuée à partir du modèle grossier non raffiné. L'application est réalisée sur un véhicule mobile en environnement urbain : la pose n'a alors que trois degrés de libert, deux en translation et un en orientation. Deux points sont détaillés dans l'article. L'un concerne l'ajustement de faisceaux, l'autre concerne une estimation en ligne et fréquente du facteur d'échelle local. L'ajustement de faisceaux reste très semblable à celui de [Lothe et al., 2009], il est toujours question de minimiser la distance des points 3D reconstruits au plan le plus proche. Dans le framework utilisé, [Mouragnon et al., 2006], les poses sont estimées à un facteur d'échelle près. La dérive du facteur d'échelle peut être très dérangeante car peut conduire à une mauvaise association points 3D / plans du modèle. Pour cela, les auteurs introduisent une estimation du facteur d'échelle. Pour ce faire, ils utilisent le plan au sol. Afin de l'identifier deux hypothèses sont posées : la normale au plan est fixe et la distance de la caméra au sol subit de faibles amplitudes, 15cm au maximum. Entre deux images, l'homographie peut être décrite par la rotation, la translation, la normale au plan et la distance de la première caméra au plan. Ainsi la seule inconnue est la norme de la translation. Elle peut alors être rapidement évaluée selon un processus de minimisation de moindres carrés, puis raffinée par moindres carrés non linéaires. Si le facteur d'échelle est manquant entre deux caméras, il est approximé pour interpolation linéaire.

Dans [Irschara et al., 2009], Irschara *et al* utilisent pour modèle des reconstructions obtenues par *Structure From Motion*, voir [Agarwal et al., 2009]. De tels modèles sont constitués des images utilisées pour la reconstruction, des poses de caméras correspondantes, d'un nuage de points et pour chaque point une liste de descripteurs *SIFT* ainsi qu'une pseudo-normale. Les descripteurs sont compressés pour réduire les tailles des modèles. Des images synthétiques sont créées lorsque nécessaire pour couvrir les zones du modèles. Seul un sous ensemble de vues assurant une bonne couverture de l'ensemble est sauvegardé. Les auteurs font l'hypothèse que les mouvements du système ne sortent pas du modèle, seule la tâche de navigation est réalisée. En cours d'utilisation, les descripteurs de l'image de la caméra sont comparés à ceux du modèle, par le biais d'un arbre de vocabulaire, [Nister and Stewenius, 2006]. La vue du modèle obtenue est appariées avec l'image courante. La pose peut alors être estimée grâce aux correspondances $3D$-$2D$.

1.6 Synthèse

Depuis l'identification de la problématique du *SLAM* de nombreuses approches pour y répondre ont été devisées. Historiquement, les approches filtrées, de *Kalman* puis particulaire, sont apparues les premières dans un contexte de *SLAM* général puis ont été adaptées au cas du *SLAM* visuel et monoculaire. Dans ce contexte sont ensuite apparues les approches *Structure From Motion*, issues de la communauté de vision. Dans le cadre du *SLAM* monoculaire, ces deux familles ont permis d'obtenir des résultats intéressants, mais se heurtent à la double difficulté de la nature incrémentale des estimations et de la nature parcellaire des informations à disposition.

Pour répondre à ces difficultés, divers travaux ont été menés tirant parti d'informations issues ou bien de capteurs additionnels ou bien d'une connaissance de l'environnement disponible *a priori*. Naturellement, les résultats obtenus sont de meilleure qualité, mais ces informations peuvent ne pas être toujours disponibles. Malgré la multiplicité des travaux et la qualité de certains résultats, le problème du *SLAM* n'est toujours pas parfaitement résolu.

Suivant les observations de Strasdat *et al*, [Strasdat et al., 2010], dans nos travaux nous avons adopté une approche de type *Structure From Motion*. A présent que nous avons défini le formalisme selon lequel nous adressons la tâche du *SLAM*, nous allons passer en revue certaines des notions et *briques* nécessaires à sa mise en oeuvre.

Chapitre 2

Fondamentaux

Dans ce chapitre nous introduisons quelques outils et notions utiles à la compréhension de ce manuscrit ou bien au lecteur souhaitant s'essayer à l'implémentation d'un système *SLAM*. Pour certaines parties, plus de détails sont disponibles dans les annexes.

Nous commençons par présenter la géométrie projective et la notation en coordonnées homogènes. Nous résumons quelques outils d'algèbre linéaire et introduisons la notion de pose de caméra. Puis nous présentons la géométrie épipolaire, une méthode de triangulation de points et le principe de l'ajustement de faisceaux. Suivant chacun de ces trois éléments nous introduisons trois manières d'estimer la pose de la caméra. Nous voyons ensuite quelques détecteurs et descripteurs de points d'intérêt, et finalement la méthode *RANSAC* servant à filtrer les appariements de ces derniers.

2.1 Géométrie projective et coordonnées homogènes

Le premier capteur employé dans le *SLAM* visuel est une caméra. Une caméra observe le monde par projection dans le plan image, voir 3.1. La géométrie projective, dont l'objet est l'étude des propriétés inchangées de figures par projection, est donc le cadre formel qui convient à la modélisation du système. Nous en donnons à présent une rapide introduction.

Soit un espace vectoriel E de dimension $n+1$, l'espace projectif associé,$P_n(E)$, est l'espace topologique quotient de E pour la relation d'équivalence

$\mathcal{R}$:

$$(\lambda \cdot X)\ \mathcal{R}\ X \tag{2.1}$$

En ce qui nous concerne E est $\mathbb{R}$ et souvent $n = 2$, parfois $n = 3$. A la géométrie projective est souvent associée la notion de coordonnées homogènes qui sont un moyen pratique de représenter les transformations dans un espace projectif.

Un vecteur v de $\mathbb{R}^n$ est représenté en coordonnées homogènes par $n + 1$ paramètres. Par convention et par commodité lorsque des transformations sont appliquées au vecteur la $n + 1$-ème coordonnée est fixée 1. Ce vecteur augmenté correspond au représentant dans $P_n(\mathbb{R})$ de l'ensemble des vecteurs $(\lambda \cdot v; \lambda)$ appartenant à $\mathbb{R}^{n+1}$. Il n'est pas toujours possible de ramener la $n+1$-ème à 1, dans ce cas elle a pour valeur 0 et signifie que le point représenté se situe à l'infini. Pouvoir ainsi définir de manière finie des points situés à l'infini est un autre atout des coordonnées homogènes. Aux transformations usuellement représentables sous forme matricielle en géométrie euclidienne, en géométrie projective par l'usage des coordonnées homogènes il est possible de représenter les translations sous forme matricielle :

$$\begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + a \\ y + b \\ 1 \end{pmatrix} \tag{2.2}$$

Le test d'appartenance d'un vecteur à un hyper-plan est également aisément représentable par produit scalaire. Par exemple, la droite d'équation $y = x+1$ est représentée sous forme homogène par le vecteur $d = \begin{pmatrix} 1 & -1 & 1 \end{pmatrix}^\top$ et la distance d'un point x de $P_2(\mathbb{R})$ à d se calcule simplement par $d \cdot x$.

2.2 Outils d'algèbre linéaire et un peu plus

Les phénomènes que nous étudions sont souvent représentables, sinon approximables, dans le cadre de l'algèbre linéaire. Nous introduisons à présent brièvement quelques outils standards qui en sont issus ainsi que les quaternions, utiles pour représenter des rotations.

Les outils d'algèbre linéaire particulièrement utiles comprennent la décomposition en valeurs singulières, *SVD*, voir B.2. C'est une méthode de factorisation de matrices, fréquement employée pour déterminer le noyau d'une application linéaire, ou encore produire la pseudo-inverse d'une matrice non inversible.

Une décomposition en valeurs singulière est cependant relativement coûteuse, voir B.2 pour plus d'informations sur les coût calculatoires.

Les matrices pseudo-inverses sont une extension de la notion d'inverse aux matrices non inversibles, voir B.3. De manière générale elles peuvent être formées à partir de la décomposition en valeurs singulières de la matrice de départ. Sous certaines conditions, voir B.3 il est possible d'obtenir par d'autres moyens des matrices pseudo-inverses pour des coûts calculatoires inférieurs à celui d'une *SVD*.

Les *matrices compagnons* constituent un autre outil utile servant à déterminer les racines d'un polynôme. Nous les avons notamment utilisées dans la partie 2.3.2 pour le calcul de matrices essentielles. Plus d'informations sont disponibles dans la partie B.4.

Les matrices de rotations, matrices orthogonales de déterminant égal à 1, sont intensivement utilisées pour représenter les orientations de la caméra. Elles présentent toutefois le défaut de représenter 3 degrés de libertés au travers de 9 coefficients.

Une représentation alternative des rotations dans l'espaces est possible en utilisant des quaternions unitaires, voir B.1. Mieux paramétrés, les quaternions unitaires ne contiennent que 4 coefficients. Cette meilleure paramétrisation des quaternions rend parfois leur utilisation préférable. C'est par exemple les cas lorsqu'il s'agit de déterminer une rotation à partir d'un système linéaire, voir 3.3.2.

La pose d'une caméra est définie par une orientation et une translation représentant la transformation d'un repère de référence vers le repère local de la caméra. Ainsi un point de l'espace repéré dans le repère de référence peut être transformé vers le repère d'une caméra en utilisant la pose de la caméra. Nous y reviendrons plus en détails dans la partie 3.1.

En géométrie projective, une application projective est une transformation qui préserve le bi-rapport de quatre points alignés. Des transformations rigides d'une caméra dans l'espace, produisent dans le plan image des observations liées par des applications projectives bijectives, encore appelées homographies. La partie C.1 donne plus de détails à ce sujet.

On appelle *direct linear transformation*, abrégé *DLT*, un algorithme permettant de déterminer une transformation S définie au facteur d'échelle près selon les observations, m^1 et m^2, de deux images. Alors qu'un système linéaire ordinaire est de la forme $m^1 = S \cdot m^2$, dans ce cas on a : $m^1 \sim S \cdot m^2$ ou $\sim$ représente le facteur d'échelle inconnu. Une méthode *DLT* consiste à réécrire le système sous forme homogène et linéaire, puis à le résoudre de manière

standard.

2.3 Géométrie épipolaire et localisation

Nous introduisons à présent la géométrie épipolaire. La géométrie épipolaire, redécouverte et introduite par Longuet-Higgins dans [Longuet-Higgins, 1987], caractérise les relations géométriques entre des objets de l'espace et/ou leurs observations dans les images de deux caméras. Les objets auxquels nous intéressons plus particulièrement sont des points.

Soit un point X de $\mathbb{R}^3$ de coordonnées X^1 et X^2 dans les bases définies par deux poses de caméras C^1 et C^2 de centres optiques O^1 et O^2 et ses observations m^1 et m^2 en coordonnées normalisées, partie 3.1. On note I^1 et I^2 les plans images de C^1 et C^2. Pour des raisons de simplicité on considère dans la suite de C^1 est la pose identité et $C^2 = (R|T)$.

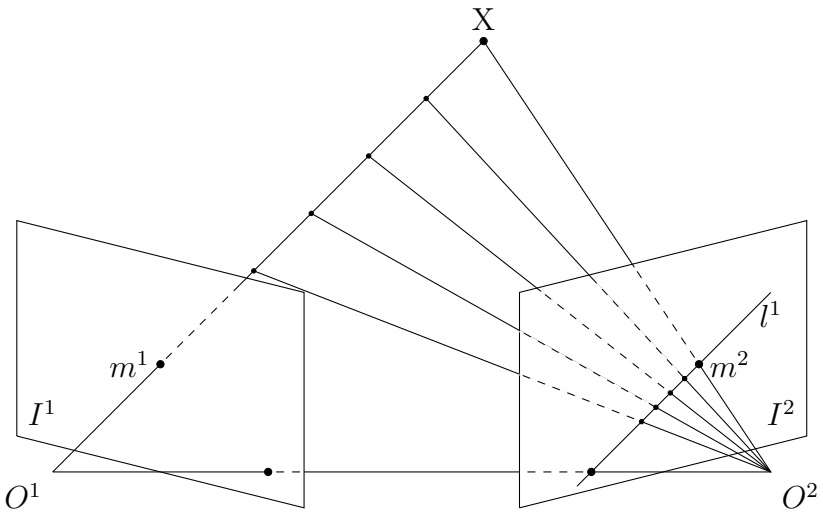


FIGURE 2.1 – Illustration de la géométrie épipolaire : le point X est observé en m^1 dans C^1. Cette mesure définit une demi-droite de $\mathbb{R}^3$, qui se projette en la droite épipolaire l^2 dans C^2.

Par définition $X_1 = \alpha \cdot m^1$ et en l'absence de connaissance sur la profondeur de la scène obervée α est inconnu. L'ensemble des points défini par $\alpha \cdot m^1$ pour $\alpha \in [0, \mathbb{R}^+]$ définit une demi-droite L dans $\mathbb{R}^3$. L'ensemble des projections dans I^2 des points $\alpha \cdot m^1$ correspond à la projection de L et définit

ainsi une droite, l^2 dans I^2. On appelle l^2 la droite épipolaire correspondant à m^1. L'inverse est vrai pour m^2 définissant une ligne épipolaire l^1 dans I^1. Cette propriété est notamment utile lorsqu'il s'agit de restreindre la recherche d'appariements $m^1 \leftrightarrow m^2$ ou encore d'évaluer la vraisemblance de ces appariements.

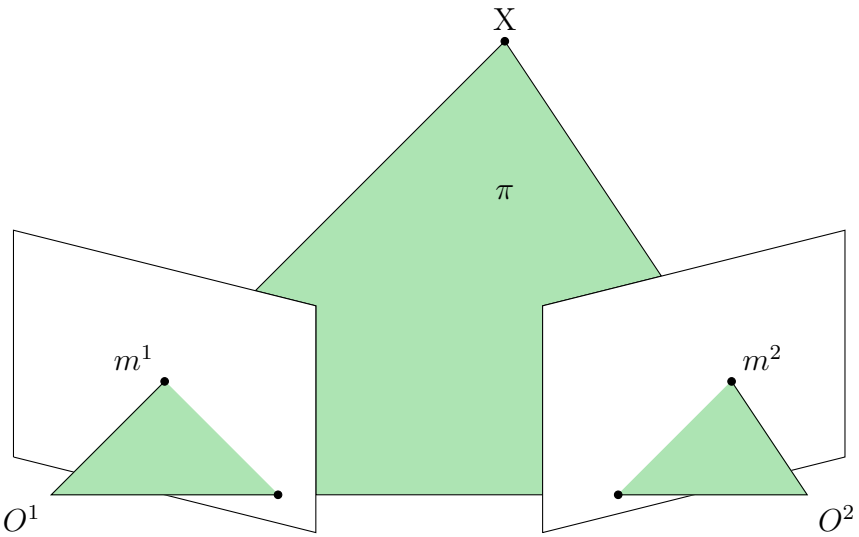


FIGURE 2.2 – Illustration de la géométrie épipolaire : les points X, O^1 et O^2 définissent un plan π intersectant les plans images de chaque caméra.

Une autre approche de la géométrie de deux vues est d'observer que les points X, O^1 et O^2 définissent un plan π dans $\mathbb{R}^3$. Par définition les points m^1 et m^2 appartiennent à π. Ainsi les vecteurs $\overrightarrow{O^1m^1}$, $\overrightarrow{O^2m^2}$ et $\overrightarrow{O^1O^2}$ sont coplanaires. Alors la relation suivante est vérifiée :

$$\begin{aligned} & \overrightarrow{O^1m^1} \cdot [\overrightarrow{O^2m^2} \times \overrightarrow{O^1O^2}] = 0 \\ \Leftrightarrow & \\ & m^1 \cdot [T \times (R \cdot m^2)] = 0 \\ \Leftrightarrow & \\ & {m^1}^\top \cdot (T_\times \cdot R) \cdot m^2 = 0 \\ \Leftrightarrow & \\ & {m^1}^\top \cdot E \cdot m^2 = 0 \end{aligned} \tag{2.3}$$

Avec la matrice $T_\times$ représentant le produit vectoriel $T \times x = T_\times \cdot x$:

$$T_\times = \begin{pmatrix} 0 & t_3 & -t_2 \\ -t_3 & 0 & t_1 \\ t_2 & -t_1 & 0 \end{pmatrix} \tag{2.4}$$

La matrice E ainsi mise en valeur est la matrice épipolaire. Le produit $E \cdot m^2$ définit la droite épipolaire l_2 et le produit ${E^\top} \cdot m^1$ la droite épipolaire l_1.

Il est important de noter que la géométrie épipolaire est bien définie lorsqu la translation n'est pas nulle. Dans notre cas, le *SLAM*, il est possible de rencontrer des cas ou la translation est nulle ou presque.

2.3.1 Estimation d'une matrice essentielle avec 8 points

Dans [Longuet-Higgins, 1987] Longuet-Higgins présente une méthode d'estimation directe de matrice essentielle à partir de 8 appariements $\{m^1 \leftrightarrow m^2\}_{1...8}$ de points dans les images I^1 et I^2 de deux caméras. Une matrice essentielle étant définie au facteur d'échelle près, il suffit de fournir 8 contraintes pour estimer ses 9 coefficients. La relation épipolaire reliant m^1 et m^2 définie par ${m^2}^\top \cdot E \cdot m^1 = 0$ ne fournissant qu'une seule contrainte, 8 appariements sont nécessaires.

On peut reformuler la relation épipolaire ${m^2}^\top \cdot E \cdot m^1 = 0$ sous la forme $\bar{m} \cdot \bar{E} = 0$ avec

$$\begin{aligned} \bar{m} &= \begin{pmatrix} m^1_x \cdot m^2_x & m^1_y \cdot m^2_x & m^1_z \cdot m^2_x & m^1_x \cdot m^2_y & m^1_y \cdot m^2_y & m^1_z \cdot m^2_y & m^1_x \cdot m^2_z & m^1_y \cdot m^2_z & m^1_z \cdot m^2_z \end{pmatrix} \\ \bar{E} &= \begin{pmatrix} E_{11} & E_{12} & E_{13} & E_{21} & E_{22} & E_{23} & E_{31} & E_{32} & E_{33} \end{pmatrix}^T \end{aligned} \tag{2.5}$$

Les 8 appariements permettent de former 8 vecteurs $\bar{m}_{i=1...8}$ et de les assembler en une matrice $M_{8\times 9}$. Dans le cas général, *i.e.* la configuration n'est pas dégénérée, la matrice M est de rang 8. Une décomposition en valeurs singulières fournit le noyau de M, c'est à dire le vecteur $\bar{E}$.

Cette approche facile à mettre en oeuvre est toutefois dégénérée pour les configurations suivantes :

- 4 points sont alignés
- 7 points sont coplanaires
- 6 points sont les sommets d'un hexagone régulier
- les 8 points correspondent aux sommets d'un cube

2.3.2 Estimation d'une matrice essentielle avec 5 points

Dans [Nistér et al., 2004] Nister propose une méthode de référence d'estimation de matrice essentielle à partir de 5 appariements $\{m^1 \leftrightarrow m^2\}_{1...5}$ de points dans les images I^1 et I^2 de deux caméras. La matrice de rotation R composante de la matrice essentielle E possèdant 3 degrés de liberté et le vecteur de translation T en possèdant 2, la matrice E possède au total 5 degrés de liberté. La relation épipolaire reliant m^1 et m^2 ne fournissant toujours qu'une seule contrainte, au moins 5 appariements sont effectivement nécessaires à son estimation.

Les 5 appariements permettent de former 5 vecteurs $\bar{m}_{i=1...5}$ et de les assembler en une matrice $M_{5\times 9}$. Dans le cas général, *i.e.* la configuration n'est pas dégénérée, la matrice M est de rang 5. Une décomposition en valeurs singulières fournit le noyau de M, dans ce cas quatre vecteurs X, Y, Z, W, de $V^\top$ associés à quatres valeurs nulles. La matrice essentielle est alors de la forme :

$$\bar{E} = x \cdot X + y \cdot Y + z \cdot Z + w \cdot W \tag{2.6}$$

La matrice essentielle étant définie au facteur d'échelle, les coefficients sont considérés normalisés de sorte que $w = 1$.

Nister remarque que
- le déterminant de E est nul
- les valeurs propres de E sont de la forme $\begin{bmatrix}\lambda & \lambda & 0\end{bmatrix}$, alors

$$E \cdot E^\top \cdot E - \frac{1}{2} \cdot \mathrm{trace}(E \cdot E^\top) \cdot E = 0 \tag{2.7}$$

Appliqué à 2.6 cela fournit dix contraintes cubiques en x, y et z. Un système linéaire composé des coefficients des différents polynômes est formé puis diagonalisé. Cela permet d'exprimer un nouveau système linéaire dont les coefficients sont des polynômes en z. Le déterminant de ce système, un polynôme de degré 10, étant nul, son expansion permet de déterminer un ensemble d'hypothèses pour z. A l'aide ce ces hypothèses sont calculées les hypothèses pour x et y. Un maximum de dix hypothèses est possible pour la matrice essentielle.

Nous avons utilisé le code[1] fournit par Nister permettant d'évaluer la matrice des contraintes dans notre implémentation de la méthode. Un résumé

1. `http://vis.uky.edu/~stewe/FIVEPOINT/calibrated_fivepoint_helper.c`

en pseudo-code est donné à l'algorithme 1.

Algorithme 1: Calcul d'une matrice essentielle avec 5-points

. Soient 5 appariements $(m^1 \leftrightarrow m^2)_{i=1\ldots5}$
. Former 5 $\bar{m}_i$ selon 2.5
. Former $M_{5\times9}$
. Calculer la décomposition en valeurs singulière de M, extraire X, Y, Z, W
. Appliquer les 10 contraintes liées au déterminant, et aux valeurs propres de la matrice essentielle en fonction de x, y, z pour obtenir la matrice A
. Réordonner les dix contraintes A selon

$$\begin{pmatrix} x^3 & y^3 & x^2y & xy^2 & x^2z & x^2 & y^2z & y^2 & xyz & \ldots \\ xy & xz^2 & xz & x & yz^2 & yz & y & z^3 & z^2 & \ldots \\ z & 1 & & & & & & & & \end{pmatrix}$$

. Diagonaliser A par pivot de Gauss
. Factoriser A selon

$$\begin{pmatrix} x^3 & \ldots & xy & x\begin{pmatrix} z^2 & z & 1 \end{pmatrix} & y\begin{pmatrix} z^2 & z & 1 \end{pmatrix} & \begin{pmatrix} z^3 z^2 z 1 \end{pmatrix} \end{pmatrix}$$

. Ne garder que les trois dernières colonnes de polynômes
. Multiplier les lignes 10, 8 et 6 de A par z
. Former un système B à partir des lignes $10-9$, $8-7$ et $6-5$
. Développer le polynôme en z correspondant au déterminant de B
. Calculer les racines du polynôme, par exemple à l'aide d'une matrice compagnon
. Pour chaque racine réelle z_i calculer x_i et y_i
. Pour chaque quadruplet $\begin{Bmatrix} x_i & y_i & z_i & 1 \end{Bmatrix}$ former la matrice

$$E_i = x_i \cdot X + y_i \cdot Y + z_i \cdot Z + W$$

Finalement, notons que contrairement à la méthode précédente, celle-ci ne souffre pas de configurations dégénérées.

2.3.3 Discussion : méthode 8 points vs 5 points

La méthode des 8 points est plus simple et nécessite moins de calculs, mais présente des configurations dégénérées et nécessite plus d'observations pour être appliquée. Nous nous sommes alors demandé quelle méthode il est préférable d'utiliser.

En effet, le nombre d'opérations nécessaires à la décomposition en valeurs singulières optimale, où seules la matrice V et les valeurs singulières sont calculées, de la matrice $M_{8\times 9}$ est de :

$$4 \cdot 8^2 \cdot 9 + 8 * 9^3 = 8136 \tag{2.8}$$

Dans le cas non-optimal où la matrice U est également calculée :

$$4 \cdot 8^2 \cdot 9 + 8 \cdot 8 \cdot 9^2 + 9 \cdot 9^3 = 14049 \tag{2.9}$$

Dans le cas de la méthode des *5-points* selon la méthode optimale le nombre d'opérations nécessaires est :

$$4 \cdot 5^2 \cdot 9 + 8 * 9^3 = 6732 \tag{2.10}$$

Dans le cas non optimal :

$$4 \cdot 5^2 \cdot 9 + 8 \cdot 5 \cdot 9^2 + 9 \cdot 9^3 = 10701 \tag{2.11}$$

A quoi il faut ajouter les coûts des étapes suivantes de la méthode, la plus importante étant celle correspondant au développement de la matrice A (environ 5000 multiplications).

A des fins de confirmation nous avons généré aléatoirement 50000 échantillons constitués d'une matrice de pose M et de 8 appariements. Les orientations des matrices de poses sont générées selon une loi uniforme telles que les rotations autour de chaque axe sont de l'ordre de 15°, et les translations selon chaque axe d'après une loi uniforme centrée en 0. Les appariements sont calculés comme les projections de points générés aléatoirement mais situés en faces des caméras dont les poses sont l'identité et M. Pour chaque échantillon nous avons calculé la matrice essentielle de référence E_r à partir de M. Nous avons calculé plusieurs matrices essentielles E_{s5} à partir de 5 des 8 appariements et la matrice essentielle E_8 à partir des 8 appariements. Afin de comparer une matrice E_i avec la matrice de référence E_r, nous réordonnons les matrices en vecteurs. Les vecteurs sont ensuite normalisés puis multipliés par le signe

de leur premier coefficient. La différence entre les matrices/vecteurs E_i et E_r peut alors être calculée comme la norme de leur différence. Pour chaque échantillon nous n'avons conservé pour matrice/vecteur E_5 que celle parmi les solutions Es_5 dont la différence avec E_r est la plus faible. Le tableau 2.1 montre les différences moyennes et leurs écarts types ainsi que les temps d'exécution moyen et leurs écarts types pour les deux méthodes.

	différence		temps (en μs)	
	moyenne	écart type	moyen	écart type
méthode des 5 points	$9.79 \cdot 10^{-6}$	$1.5 \cdot 10^{-3}$	5266	1041
méthode des 8 points	$2.89 \cdot 10^{-14}$	$2.7 \cdot 10^{-12}$	46	12

TABLE 2.1 – Tableau comparatif des méthodes 5 et 8 points en précision et temps de calcul

On peut donc vérifier que la méthode des 8-points donne des résultats au moins aussi satisfaisants que la méthode des 5-points pour des temps de calculs largements inférieurs.

En pratique, les méthodes d'estimation de matrice essentielles sont appliquées dans une procédure de robustification telle que le *RANSAC*, voir la partie 2.9. Dans ce cas le nombre mininal de 5 appariements offre l'avantage de nécessiter moins d'essais. Le coût de la mise en oeuvre du *RANSAC* combiné au nombre d'essais plus important pour la méthode des 8 points rend la méthode des 5 points plus rapide. La moindre précision de celle-ci n'est pas problématique, puisque les paramètres de poses tirés de la matrice essentielle sont toujours affinés selon une procédure de minimisation d'erreurs. Enfin, l'absence de configurations dégénérées est un avantage. En particulier au regard des configurations planaires, qui peuvent être fréquentes dans certains environnements urbains.

2.3.4 Estimation de pose depuis une matrice essentielle

Dans [Hartley and Zisserman, 2004] Hartley et Zisserman présentent la procédure de décomposition d'une matrice essentielle E en quatre matrices de pose $\begin{pmatrix} R_1 & T \end{pmatrix}$, $\begin{pmatrix} R_1 & -T \end{pmatrix}$ $\begin{pmatrix} R_2 & T \end{pmatrix}$ $\begin{pmatrix} R_2 & -T \end{pmatrix}$. Les translations de ces matrices de pose, comme la matrice essentielle, ne sont définies qu'au facteur d'échelle près.

La matrice E se décompose en valeur singulières comme $E = U \cdot \text{diag}\begin{pmatrix}1 & 1 & 0\end{pmatrix}^\top \cdot V^\top$. Alors on note $E = S \cdot R$ et elle se décompose de deux, et seulement deux, manières possibles :

$$E_1 = S \cdot R_1 \quad , \quad E_2 = S \cdot R_2 \tag{2.12}$$

avec :

$$\begin{gathered} S = U \cdot Z \cdot U^\top \qquad R_1 = U \cdot W \cdot V^\top \qquad R_2 = U \cdot W^\top \cdot V^\top \\ W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad Z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{gathered} \tag{2.13}$$

On note u_i la i-ième colonne de U. La matrice S correspond à la matrice $\begin{bmatrix}T\end{bmatrix}_\times$, alors $S \cdot T = \mathbf{0}$. U étant orthogonale, la partie $Z \cdot U^\top$ est celle qui doit annuler $S \cdot T$. Donc

$$\begin{gathered} Z \cdot U^\top \cdot T = 0 \\ \begin{pmatrix} u_2 \\ u_1 \\ \mathbf{0} \end{pmatrix} \cdot T = 0 \end{gathered} \tag{2.14}$$

Ainsi :

$$T = u_3 \tag{2.15}$$

Les quatres configurations se désambiguïsent facilement, théoriquement une seule place les appariements $m^1 \leftrightarrow m^2$ devant les deux caméras. En pratique et pour de faibles mouvement il est préférable d'utiliser plusieurs appariements et de sélectionner la pose plaçant un maximum de points devant les deux caméras.

2.4 Cartographie par triangulation

La géométrie épipolaire et les algorithmes de calcul de pose introduits précédemment nous permettent d'adresser la question de la localisation du système. La question de cartographie est adressée en reconstruisant dans l'espace les positions de points dont les observations sont disponibles dans

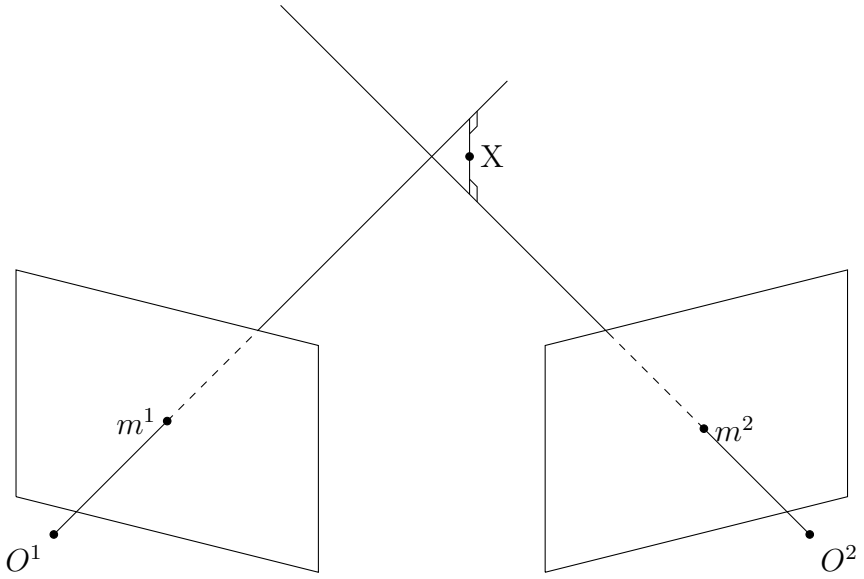


FIGURE 2.3 – Illustration de la méthode dite du *point milieu* pour la triangulation.

deux caméras ou plus. Ces procédures portent le nom de méthodes de triangulation. Nous présentons d'abord un peu plus formellement la tâche puis présentons la méthode de triangultion dite du point milieu.

Soit un point X de position inconnue dans $\mathbf{P}^4$ observé en m^1 et m^2 dans les images respectives de deux caméras C^1 et C^2. L'objectif de la triangulation est de retrouver les coordonnées du point X à partir des points m^1, m^2 et des poses C^1, C^2.

2.4.1 Triangulation par point milieu

Une méthode simple et peu coûteuse de triangulation est celle dite du *point milieu*.
A partir des observations de X le principe est de tracer les rayons, r^1 et r^2, partant des centres optiques des caméras, O^1 et O^2, passant par les points image et de calculer leur intersection. Les rayons sont paramétrés comme suit :

$$\begin{array}{rcl} r^1 & = \{X^1 & | \quad X^1 = O^1 + \alpha \cdot \overrightarrow{O^1 m^1} \quad \} \\ r^2 & = \{X^2 & | \quad X^2 = O^2 + \alpha' \cdot \overrightarrow{O^2 m^2} \quad \} \end{array} \tag{2.16}$$

Cependant, en pratique, les rayons s'intersectent rarement. La solution du *point milieu* consiste à trouver le point le plus proche des rayons.

On détermine d'abord les coefficients α et α' tels que la distance entre $O^1 + \alpha \cdot \overrightarrow{O^1m^1}$ et $O^2 + \alpha' \cdot \overrightarrow{O^2m^2}$ est minimale. On note :

$$\begin{array}{rccl} f: & \mathbb{R}^2 & \rightarrow & \mathbb{R}^3 \\ & (\alpha, \alpha') & \mapsto & O^1 + \alpha \cdot \overrightarrow{O^1m^1} - (O^2 + \alpha' \cdot \overrightarrow{O^2m^2}) \end{array} \tag{2.17}$$

Cette fonction est linéaire en α et α', on peut donc écrire $f((0,0)+(\alpha,\alpha')) = f(0,0) + J_f \cdot (\alpha,\alpha')^\top$ où $J_f \in M_{3,2}(\mathbb{R})$ est la jacobienne de $f()$. On a $J_f = \begin{pmatrix} \overrightarrow{O^1m^1} & -\overrightarrow{O^2m^2} \end{pmatrix}$. On note $D = O^1 - O^2$, on recherche α et α' tels que $f(0,0) + J_f \cdot (\alpha,\alpha')^\top = 0$. La solution est obtenue en résolvant l'équation connue sur le nom d'équation *normale* $J_f^\top \cdot J_f \cdot (\alpha,\alpha')^\top = -J_f^\top \cdot D$. Ce qui donne :

$$\begin{pmatrix} \alpha \\ \alpha' \end{pmatrix} = \frac{\begin{pmatrix} \overrightarrow{O^2m^2}^\top \cdot \overrightarrow{O^2m^2} & \overrightarrow{O^1m^1}^\top \cdot \overrightarrow{O^2m^2} \\ \overrightarrow{O^1m^1}^\top \cdot \overrightarrow{O^2m^2} & \overrightarrow{O^1m^1}^\top \cdot \overrightarrow{O^1m^1} \end{pmatrix} \cdot \begin{pmatrix} -\overrightarrow{O^1m^1}^\top \cdot D \\ \overrightarrow{O^2m^2}^\top \cdot D \end{pmatrix}}{(\overrightarrow{O^1m^1}^\top \cdot \overrightarrow{O^1m^1}) \cdot (\overrightarrow{O^2m^2}^\top \cdot \overrightarrow{O^2m^2}) - (\overrightarrow{O^1m^1}^\top \cdot \overrightarrow{O^2m^2})^2} \tag{2.18}$$

Une fois les coefficients α et α' optimaux obtenus le point est reconstruit comme le point au milieu de X^1 et X^2 :

$$X = \frac{1}{2}\left((O^1 + \alpha \cdot \overrightarrow{O^1m^1}) + (O^2 + \alpha' \cdot \overrightarrow{O^2m^2})\right) \tag{2.19}$$

Cette méthode est essentiellement géométrique. D'autres méthodes, algébriques, sont présentées dans la partie E. Ces autres méthodes offrent l'avantage de pouvoir prendre en compte plus de deux observations, mais à des coûts calculatoires légèrement plus importants. Nous rappelons également dans cette partie les travaux d'Hartley et Sturm [Hartley and Sturm, 1997] traitant d'un préconditionnement des observations afin d'obtenir une reconstruction optimale en prenant en compte la géométrie épipolaire reliant les deux vues. Nous jugeons néanmoins ce préconditionnement non nécessaire dans notre contexte puisque les positions des points reconstruits sont régulièrement affinées par ajustement de faisceaux.

2.5 Estimation de pose à partir de correspondances 3D-2D

Nous avons vu comment calculer la pose relative d'une caméra à partir de correspondances entre des observations de points dans deux images. Nous avons également vu comment à partir de deux observations d'un point dans les images de deux caméras dont la pose relative est connue il est possible d'en reconstruire la position dans l'espace.

L'étape suivante consiste à estimer la pose d'une caméra à partir de correspondances entre les positions de points dans l'espace et leurs observations dans l'image. Les méthodes résolvant ce problèmes sont communément appelées *PnP* pour *Perspective-n -Points*. Lorsque $n = 3$ on parle de méthode *P3P*. L'article [Haralick et al., 1994] d'Haralick *et al* offre un aperçu de plusieurs méthodes résolvant ce problème, dont la première apparue est la méthode de Grunert. Dans [Lepetit et al., 2009] Lepetit *et al* ont introduit la méthode *EPnP* actuellement la méthode de référence pour résoudre ce problème. Cette méthode offre des résultats très précis pour des coûts calculatoires particulièrement intéressants car linéaires avec le nombre de correspondances utilisées. Les auteurs observent que les estimations de poses sont toujours améliorées par l'application d'un ajustement de faisceaux.

2.6 Ajustement de faisceaux

Nous avons à présent vu plusieurs méthodes d'estimation de pose de caméra ainsi que de triangulation de points. Pour plusieurs raisons ces méthodes sont imprécises. En effet, les mesures sont souvent bruitées et les calculs, de pose ou de triangulation, ne peuvent pas être d'une précision infinie. Des erreurs apparaissent, et comme le *SLAM* est un processus incrémental elles se propagent. Il est alors nécessaire de réduire ces erreurs.

En vision par ordinateur la méthode de référence de réduction d'erreurs porte le nom d'ajustement de faisceaux. L'ajustement de faisceaux est une application des méthodes de moindres carrés non linéaires au cas d'une ou plusieurs caméras observant des points dans l'espace.

Ces méthodes raffinent localement un ensemble de paramètres selon une fonction de coût minimisée. Elles nécessitent pour cela de connaître une estimation initiale de bonne qualité. La fonction de coût est formée des erreurs de reprojection, c'est à dire des distances entre les mesures de points et les

reprojections de leurs reconstructions selon les paramètres de pose de caméra estimés. Dans un processus d'ajustement de faisceaux les mesures sont usuellement considérées de bonne qualité. L'influence du bruit, s'il n'est pas trop important et équiréparti, peut être annulée par la masse des données. Ainsi, seuls les paramètres de pose des caméras et de position des points peuvent être modifiés dans le but de réduire les erreurs de reprojections.

L'annexe A présente ces notions et quelques méthodes courantes avec plus de détails ainsi que quelques bibliothèques logicielles.

Usuellement les données raffinées par ajustement de faisceaux ont subi un filtrage afin d'éliminer de mauvaises associations. Il est cependant difficile de garantir que celles-ci ont toutes été supprimées. Ces mauvais associations peuvent avoir des effets importants si elles ne sont pas prises en compte. Ce problème est adressé en faisant usage de fonctions de coût robustes. Cette notion est abordée dans l'annexe D.

2.7 Estimation de pose par ajustement de faisceaux

Comme évoqué dans la partie 2.5 l'une des méthode d'estimation de pose à partir de correspondances 3D-2D les plus performantes est probablement celle introduite par Lepetit *et al* dans [Lepetit et al., 2009]. Dans cet article les auteurs comparent leur méthode à plusieurs autres méthodes de référence et montrent que la leur est la plus efficace et que l'estimation fournie gagne toujours à être raffinée par un processus d'ajustement de faisceaux. La méthode sert donc à trouver une très bonne estimation initiale.

Lorsque l'on cherche à estimer la pose d'une nouvelle caméra, puisque nous faisons du *SLAM* nous savons que celle-ci est proche de la dernière estimée. Cela nous fournit donc une bonne initialisation pour effectuer un ajustement de faisceaux. Il est toutefois nécessaire d'avoir préalablement filtré les *outliers*, voir la partie 2.9, par exemple au travers d'une procédure *RANSAC* basée sur l'estimation de la géométrie épipolaire, par exemple via la matrice essentielle, partie 2.3.

Il est aussi possible d'estimer la géométrie épipolaire sur l'ensemble des *inliers*, mais l'ajustement de faisceaux présente deux avantages sur cette démarche. La première est que la géométrie épipolaire est définie au facteur d'échelle près, ainsi l'ajustement de faisceaux permet de résoudre l'estima-

tion du facteur d'échelle. L'autre avantage de l'ajustement de faisceaux est que celui-ci n'est pas affecté par le mauvais conditionnement de la géométrie épipolaire dans le cas de translations infinitésimales.

De plus la complexité algorithmique est inférieure à celle d'une méthode d'estimation de matrice essentielle à partir de $N > 5$ points, voir 2.3.2.
Selon [C. Engels, 2006], la complexité algorithmique de l'ajustement de faisceaux est de $O(n^2 \cdot m)$,avec $n = 2$ le nombre de caméras impliquées et m le nombre de points impliqués, soit $O(m)$.
La complexité algorithmique de l'estimation de matrice essentielle par *SVD*, voir 2.3.4 est de $4 \cdot m^2 \cdot n + 8 \cdot n^3$ avec $n = 9$ le nombre de colonnes de la matrice décomposée et m le nombre de lignes de la matrice, c'est à dire le nombre de points impliqués. Au final la complexité est de l'ordre de $O(m^2)$.

Finalement, un autre avantage important de l'utilisation de l'ajustement de faisceaux pour l'estimation de pose est que contrairement à une méthode d'estimation absolue, comme présenté dans 2.5, l'étape d'initialisation de la pose permet, et nécessite, de prendre en compte l'historique des poses.

2.8 Points d'intérêt

Comme indiqué précédemment, nous modélisons l'environnement et estimons le mouvement du système grâce à des mises en correspondance de mesures de points dans différentes prises de vue de la caméra. Tous les pixels d'une image ne s'y prêtent pas de manière égale. La figure 2.4 illustre ce propos. On y voit une image et trois régions extraites. La première est assez uniforme, aucun point ne semble significativement différent de ses voisins. La seconde présente une arête, un ensemble de points caractéristiques de leurs voisins selon une direction, ici horizontale, mais similaires à leurs voisins selon la direction orthogonale. Enfin, la troisième présente un coin, un point différent de ses voisins dans toutes les directions.

On appelle détecteurs de points d'intérêt les procédures permettant de déterminer des points caractéristiques. On appelle descripteurs de points d'intérêt celles permettant d'associer une signature à un point. Les signatures sont employées pour mettre en correspondance plusieurs observations dans différentes images. Deux observations sont alors appariées lorsque la distance entre leurs signatures est suffisamment faible.

Dans cette partie nous présentons brièvement quelques détecteurs et descripteurs usuels.

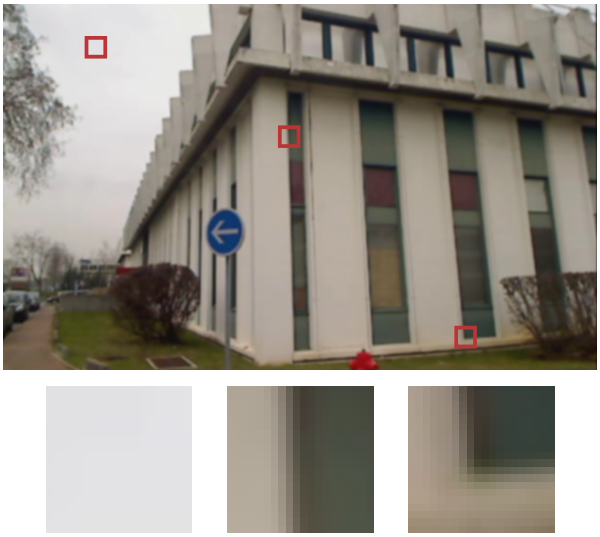


FIGURE 2.4 – Trois régions extraites d'une image. La première est assez uniforme, la seconde présente une arête, la troisième contient un coin.

2.8.1 Détecteurs de points

Les capteurs de couleurs pouvant s'avérer peu robustes aux variations d'illumination, les détecteurs de points d'intérêt opèrent généralement uniquement sur l'intensité lumineuse. L'image est alors composée de niveaux de gris.

La distinction entre zones uniformes, arêtes, et coins est à la base de l'un des premiers formalisme de détections de points d'intérêt. La distance entre deux régions de deux images, I_0 et I_1, peut être évaluée par la fonction de somme des différences quadratiques pondérées (*SSD*) suivante :

$$\mathrm{SSD}_{\{I_0,I_1\}} = \sum_i w(x_i) \cdot (I_1(x_i) - I_0(x_i))^2 \tag{2.20}$$

La recherche de la région de I_1 la plus proche de I_0 peut être effectuée en minimisant :

$$\mathrm{SSD}_{\{I_0,I_1\}}(u) = \sum_i w(x_i) \cdot (I_1(x_i + u) - I_0(x_i))^2 \tag{2.21}$$

où

- u représente un déplacement sur le support de I_1
- $w(x)$ est une fonction de poids
- l'ensemble des i indice les pixels dans la région
- x_i est la valeur du pixel d'indice i

Un point d'intérêt x_i précisément localisable doit présenter localement un minimum unique pour la fonction $\mathrm{SSD}_{\{I_0,I_0\}}$. Autrement dit : il ne doit pas pouvoir être confondu avec ses proches voisins. En appliquant un développement limité au premier ordre, la fonction $\mathrm{SSD}_{\{I,I\}}$ s'approxime comme :

$$\begin{aligned}\mathrm{SSD}_{\{I,I\}}(u) &= \textstyle\sum_i w(x_i) \cdot (I(x_i+u) - I(x_i))^2 \\ &\approx \textstyle\sum_i w(x_i) \cdot (I(x_i) + J_I(x_i) \cdot u - I(x_i))^2 \\ &\approx \textstyle\sum_i w(x_i) \cdot (J_I(x_i) \cdot u)^2 \\ &\approx u^\top \cdot A \cdot u\end{aligned} \tag{2.22}$$

avec

$$\begin{aligned}J_I(x_i) &= (\tfrac{\partial I}{\partial x}, \tfrac{\partial I}{\partial y})(x_i) = (I_x, I_y)(x_i) \\ A &= w * \begin{pmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{pmatrix}\end{aligned}$$

Les images I_x et I_y sont obtenues en convoluant I avec des filtres de gradient, par exemple $\begin{pmatrix}-1 & 0 & 1\end{pmatrix}$ et sa transposée. Pour une zone uniforme, les valeurs propres, λ_0 et λ_1, de A sont proches de 0. Pour une arête seule l'une des valeurs propres est proche de 0. Enfin, pour un coin les deux valeurs propres sont différentes de 0.

Dans [Harris and Stephens, 1988] Harris et Stephens constatent que le calcul des valeurs propres de A peut être coûteux et proposent de détecter les coins en déterminant les points maximisant la quantité suivante :

$$\det(\mathrm{A}) - \alpha \cdot \mathrm{trace}(A) = \lambda_0 \cdot \lambda_1 - \alpha \cdot (\lambda_0 + \lambda_1)^2 \quad \text{avec } \alpha = 0.06 \tag{2.23}$$

Ce détecteur est couramment appelé détecteur de Harris.

Dans [Shi and Tomasi, 1994] Shi et Tomasi observent que sous certaines conditions il est préférable de sélectionner comme points d'intérêt ceux maximisant la plus petite des valeurs propres, donc la quantité :

$$\min(\lambda_0, \lambda_1) \tag{2.24}$$

Ce détecteur est courammant appelé *KLT*.

D'autres types de points non détectés par des détecteurs de coins sont susceptibles d'être des points d'intérêt de qualité. Considérons un pixel, ou un amas de pixels, de la forme illustrée à la figure 2.5. Les valeurs propres de

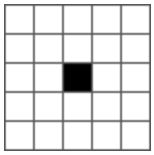


FIGURE 2.5 – Un point intéressant pourtant non détectable par les méthodes de recherche de coins

A calculées en ce pixel ne le rendent pas éligible au statut de point d'intérêt au sens de coins alors que ce pixel est clairement distinct de ses voisins. On peut démontrer que la différence entre un point et la valeur moyenne de ses voisins est proportionnelle à la valeur de l'opérateur laplacien en ce point :

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{2.25}$$

Dans [Lowe, 2004], en améliorant [Lowe, 1999], Lowe introduit un détecteur multi-échelle de point d'intérêt. S'appuyant sur des recherches antérieures, il utilise des filtres de convolution gaussiens pour flouter plusieurs fois l'image étudiée afin de simuler des observations à des échelles supérieures. Une pyramide d'image est également créée. Dans la pyramide la taille de l'image est successivement réduite pour limiter les calculs, voir la figure 2.6. Lowe remarque ensuite que l'opérateur laplacien appliqué sur les images floutées est égal à la dérivée de ces images. Les soustractions d'images floutées successives, DoG, fournissent alors des approximations d'images du laplacien de gaussienne. Finalement, les points d'intérêts sont sélectionnés comme les pixels extrema dans les images DoG par rapport à leur 8 voisins à la même échelle et leurs 9×2 voisins dans les images DoG aux échelles voisines. Les points de faibles contrastes ou situés sur des arêtes sont élagués. Ce détecteur est couramment appelé *SIFT*.

Dans [H. Bay and Gool, 2006] Bay *et al* reprennent les principes du *SIFT* en modifiant l'algorithme en profondeur de sorte à accélérer les calculs. L'image étudiée est en premier lieu convertie en image intégrale. L'intérêt de ce type d'image est de permettre de calculer très rapidement, en quelques

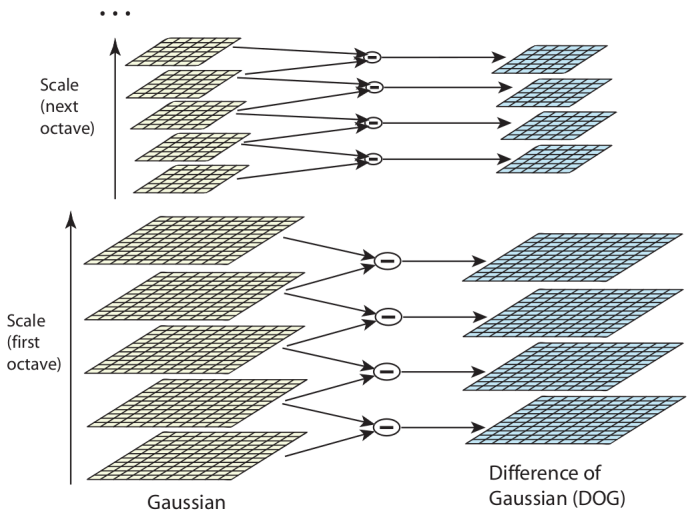


FIGURE 2.6 – Illustration du processus de création d'images de différences de gaussiennes. Image tirée de [Lowe, 2004].

opérations, la somme des intensités d'une région, quelque soit sa taille, de l'image d'origine. A la différence de [Lowe, 2004] les auteurs ne cherchent pas à approximer un opérateur laplacien, soit la trace de la matrice hessienne de l'image d'origine, mais le déterminant de cette hessienne. Le calcul des coefficients de la matrice est approximé en convoluant l'image avec des filtres simplifiés (à coefficient entiers notamment), se traduisant par des opérations rapides sur l'image intégrale. Une pyramide d'image est également émulée. Dans ce cas la taille de l'image n'est pas réduite, mais les tailles des masques de convolutions simplifiés sont augmentées. L'usage d'images intégrales garantissant la constance du nombre d'opérations, la méthode s'en trouve accélérée. Les finesses de facteur d'échelle obtenues croîssent de manière exponentielle au fil de la pyramide. Ce détecteur est couramment appelé *SURF*

Dans [Rosten and Drummond, 2006] Rosten et Drummond généralisent une méthode connue sous le nom de *critère de segment de test*. Ce test classifie un pixel p comme un coin s'il existe $n = 12$ pixels contigüs dans un cercle de périmètre 16 pixels autour de p tels qu'ils soient tous plus clairs ou plus sombres que p. La figure 2.7 illustre ce propos. Cette méthode peut être évaluée très efficacement selon une stratégie *diviser pour régner* pour $n = 12$. Les auteurs constatent que pour $n < 12$ il est difficile de deviser une stratégie efficace. Alors, pour un n fixé ils proposent une méthode d'appren-

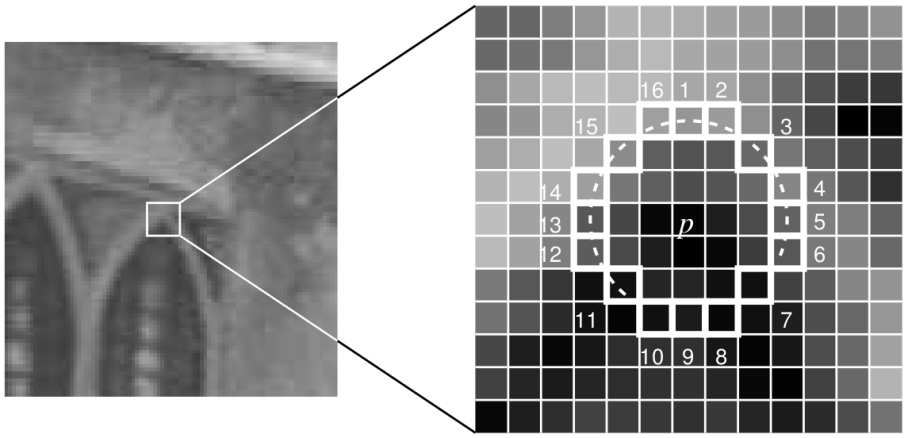


FIGURE 2.7 – Illustration de la méthode de *segment de test*. Image tirée de [Rosten and Drummond, 2006].

tissage d'un arbre de décision pour une classification rapide de pseudo-coins. Si plusieurs coins détectés sont trop proches, seul le plus saillant est conservé. Ce détecteur de coins est courammant appelé *FAST*.

Dans [Agrawal et al., 2008] Agrawal *et al* observent que les points d'intérêt détectés par les détecteurs *SIFT* et *SURF* perdent en précision lorsque le facteur d'échelle augmente. Selon les auteurs cela est dû aux schémas de pyramides employés. Il proposent alors de déterminer un moyen de détection qui soit suffisamment peu coûteux pour être appliqué sur un nombre d'échelles important tout en restant précis. Ils suggèrent alors d'approximer l'opérateur laplacien avec des filtres simples *center-surround*, comme ceux illustrés à la figure 2.8. Dans ces filtres simples les pixels d'une même région (intérieure ou

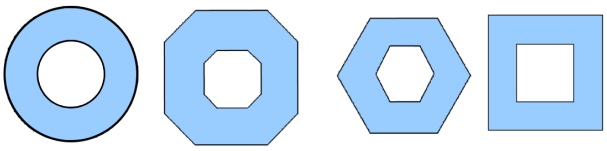


FIGURE 2.8 – Les filtres laplaciens *center-surround* de formes : circulaire, octogonale, hexagonale et carrée. Image tirée de [Agrawal et al., 2008].

extérieure) ont tous le même poids, la somme des coefficients d'un filtre doit être nulle. Le filtre de forme carrée est une généralisation du filtre laplacien

bien connu :

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

Le filtre choisi est appliqué en chaque point à différentes échelles. Les points extrema dans l'espace image et l'espace d'échelle sont des points d'intérêt potentiels. Les candidats situés sur des arêtes ou bien trop proches d'autres candidats plus saillants sont élagués. Selon les auteurs le filtre de forme carré simple à évaluer n'est pas invariant au changement d'orientation, le filtre de forme octogonale est plus approprié. Le premier peut s'évaluer rapidement à l'aide d'images intégrales, pour évaluer efficacement le second, les auteurs proposent une version modifiée d'image intégrale adaptée. Ce détecteur de points est appelé *CenSurE*, dans la bibliothèque *OpenCV* il porte le nom de *STAR*.

Il peut parfois être intéressant d'utiliser des arêtes. En effet, des effets de flous lors de la formation de l'image peuvent perturber la détection de points d'intérêts. Pour amoindrir ce phénomène, Klein utilise dans [G. Klein, 2008] des arêtes en plus de points. Les arêtes parallèles à la direction du mouvement sont robustes au flou.

Dans [Scaramuzza et al., 2009], Scaramuzza *et al* utilisent trois détecteurs de points pour évaluer leur propre formalisme de *SLAM*. Les trois détecteurs sont des points de Harris, des *KLT* et des *SIFT*. Les auteurs constatent que la répartition des *Harris* et *SIFT* peut être très localisée, alors que celle des *KLT* est beaucoup plus uniforme. Ils observent des estimations de mouvement dégradées par ces mauvaises distributions, donc plus fiables avec le *KLT* que les deux autres.

2.8.2 Descripteurs de points

La manière la plus simple de décrire un point d'intérêt consiste à considérer une région autour du point. Deux points peuvent alors être comparés en calculant le score de *SSD*, 2.20. Un inconvénient est que l'orientation des deux observations doit être la même pour obtenir un faible score.

Le détecteur de point *SIFT*, [Lowe, 2004], est également muni d'un descripteur. L'échelle de détection du point considéré définit la taille de la région l'entourant servant à calculer le descripteur. En calculant et pondérant les gradients de cette région une orientation dominante est calculée pour le point considéré, permettant d'obtenir une invariance du descripteur aux

changements d'orientation. Les gradients sont ensuite pivotés conformément à l'orientation calculée. L'ensemble des gradients, pondérés, est ensuite assemblé en un vecteur de taille 128, constituant le descripteur *SIFT*. Comme le détecteur, ce descripteur est couramment appelé *SIFT*.

Comme le *SIFT*, le détecteur *SURF* fournit aussi un descripteur. Toujours de manière similaire au *SIFT* une orientation dominante peut être calculée pour améliorer la robustesse du descripteur aux variations d'orientations. Les auteurs observent que même sans cela le descripteur est robuste à des variations de point de vue de $\pm 15°$. Les gradients calculés dans le *SIFT* sont approximés par des ondelettes de Haar, calculées efficacement grâce aux images intégrales. Le descripteur est formé des réponses aux ondelettes de Haar en x et en y ainsi que leurs valeurs absolues. Le vecteur résultat est à 64 dimensions. Comme le détecteur associé, ce descripteur est couramment appelé *SURF*.

Dans [Calonder et al., 2010] Calonder *et al* proposent de calculer un descripteur binaire. L'intérêt en est que l'évaluation de la distance entre deux tels descripteurs est réalisée en calculant une norme de Hamming, ce qui est très rapide. Les n bits du descripteur sont le résultat d'une comparaison d'intensité entre n paires de points situés dans une régions autour du point décrit. Les pixels de la régions sont au préalable adoucis par convolution avec un filtre gaussien pour atténuer les effets de bruit. Les auteurs préconisent d'utiliser $n = 128 \ldots 512$ bits. Ce descripteur est couramment appelé *BRIEF*.

Dans [Rublee et al., 2011] Rublee *et al* proposent d'améliorer le descripteur *BRIEF*. La première amélioration se base sur l'hypothèse que le point décrit est un coin et propose une méthode de calcul d'orientation. L'intérêt étant de rendre le descripteur robuste aux variations d'orientation. A partir de l'angle calculé, les n paires du *BRIEF* sont rotatées et le descripteur peut être calculé. Les auteurs observent cependant que ce *BRIEF* modifié est moins discriminant que l'original. La deuxième amélioration est basée une étude exhaustive des paires de test sur une large base de données, afin de déterminer les plus intéressantes. Ce descripteur est couramment appelé *ORB*.

Dans [Leutenegger et al., 2011] Leutenegger *et al* proposent également un descripteur binaire orienté. Autour du point décrit, $N = 60$ points et rayons sont échantillonnés de manière concentrique. Chaque couple de point-rayon définit une petite région et l'intensité de chaque région est évaluée suivant une pondération gaussienne. Pour l'ensemble des paires de points supérieures à une distance d fixée, un gradient est approximé, et la moyenne

de ces gradients fournit l'orientation du point. Le descripteur est formé à partir des 512 comparaisons des gradients pour l'ensemble des points dont la distance est inférieure à d. Ce descripteur est couramment appelé *BRISK*.

Dans [Alahi et al., 2012] Alahi *et al* proposent également un descripteur binaire. Les auteurs s'inspirent du système visuel humain pour échantillonner des régions autour du point décrit. A la différence du *BRISK* l'échantillonnage est plus dense près du point, et les régions peuvent se recouvrir. Les bits formant le vecteur de description sont obtenus comme précédemment. A la manière du *ORB* les auteurs réalisent un apprentissage pour déterminer les 512 paires les plus discriminantes. Ce descripteur est couramment appelé *FREAK*.

2.9 Robustification

Jusqu'à présent nous avons considéré des mesures parfaites ou faiblement bruitées. En pratique elles ne le sont pas et il est fréquent d'obtenir de mauvais appariements de points d'intérêt ou bien de sélectionner des ensembles de points d'intérêts mal conditionnés. Plusieurs stratégies existent afin de réduire l'influence de ces phénomènes.

La première de ces stratégies porte de le nom de *RANSAC*, nous la détaillons à présent. D'autres méthodes de robustification sont présentées dans la partie D.

RANSAC

L'algorithme *RANSAC*, pour *RANdom SAmple Consensus*, a été introduit dans la communauté de vision par ordinateur dans [Fischler and Bolles, 1981]. C'est une méthode probabiliste de sélection de modèle de transformation de paramètres robuste aux données erronées.

Initialement développée pour résoudre le problème *Perspective 3 points*, c'est à dire de l'estimation de position à partir de 3 points, cette méthode est très générale et s'adapte naturellement à tout type de transformation que l'on sait modéliser.
Usuellement les paramètres d'entrée de la méthode sont des appariements de points d'intérêt, mais d'autres formats de paramètres sont compatibles, par exemple dans le cas où l'on cherche à estimer un hyper-plan.
Les paramètres d'entrées sont considérés comme constitués pour part d'*inliers*

et d'*outliers*, c'est à dire de points vérifiant la transformation que l'on cherche à estimer et de points ne la vérifiant pas. L'objectif de la méthode est alors de déterminer conjointement quels sont les *inliers*, les *outliers* et la transformation. S'il existe plusieurs groupes d'*inliers*, chacun associé à une transformation différente, alors le plus important sera sélectionné.
Pour cela il faut savoir :

- générer une hypothèse de transformation à partir d'un nombre minimal de paramètres d'entrée
- calculer une erreur lorsque l'on applique la transformation à un paramètre
- déterminer un seuil d'erreur en deça duquel un paramètre est considéré comme *inlier* pour la transformation courante. C'est souvent la partie la plus délicate dans la mise en oeuvre de l'algorithme.

Si l'on sait faire cela le principe est simple : il consiste à sélectionner aléatoirement un nombre minimal de points pour déterminer la transformation associée à cet ensemble. Le support d'une transformation correspond au nombre de points total dont l'erreur est inférieure au seuil, et la meilleure transformation est celle ayant le plus grand support.

Nombre d'essais

Dans le cas général il est impossible d'effectuer une sélection exhaustive des paramètres générateurs de la transformation. Aussi, il est nécessaire de savoir fixer une borne au nombre d'essais. Pour cela, la méthode propose de fixer une probabilité p et de déterminer un nombre de tentatives minimal pour lequel au moins un essai n'a pas été contaminé par des *outliers* avec cette probabilité p.

Notons w le pourcentage *a priori* d'*inliers*, donc la probabilité qu'un point choisi aléatoirement soit *inlier*, s le nombre de paramètres nécessaires à l'estimation de la transformation et N le nombre d'essais à déterminer. La probabilité de n'avoir sélectionné aucun échantillon constitué uniquement d'*inliers* à l'issu de N essais est de :

$$\begin{aligned} & P(\text{échec}) = (1 - w^s)^N = 1 - p \\ \Leftrightarrow \quad & \\ & N = log(1 - p)/log(1 - w^s) \end{aligned} \tag{2.26}$$

Le tableau présente pour $p = 0.99$ et quelques valeurs de s et $1-w$ les valeurs

de N obtenues.

Echantillons	% *outliers*						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

TABLE 2.2 – Quantité d'essais nécessaires pour que l'algorithme *RANSAC* sélectionne avec une probabilité de 0.99 un ensemble d'échantillons ne contenant aucun *outlier* selon la proportion de ceux-ci et le cardinal de l'ensemble. Les valeurs sont reprises de [Hartley and Zisserman, 2004].

A chaque essai le support de la transformation courante est calculé et s'il excède celui de la meilleure transformation jusque là, alors celle-ci est mise à jour. Si le support est plus important que la quantité d'*inliers a priori* alors la quantité d'essais à effectuer est mise à jour. Il est ainsi possible d'initialiser la méthode avec une quantité *a priori* d'*outliers* élevée par précaution et la méthode s'adapte automatiquement à la quantité observée. L'algorithme s'arrête lorsque le nombre d'essais est atteint.

Si l'on cherche à estimer la meilleure transformation, il est recommandé de la re-calculer à l'issue de la procédure en prenant en compte tous les *inliers*. Cela n'est toutefois pas nécessaire si l'on cherche seulement à séparer les *inliers* des *outliers*. En effet, dans le cas de l'estimation d'une matrice essentielle si la variation de position est faible relativement à la profondeur de la scène, la géométrie épipolaire est mal conditionnée et les estimations de la matrice peu fiables. Les erreurs de mesure des points d'intérêt peuvent renforcer ce phénomène. Si le mouvement des points dans l'image est de l'ordre de l'imprécision du détecteur de points utilisé, il est peu vraisemblable d'estimer correctement la part de translation qui compose la matrice essentielle. C'est un cas de figure fréquent dans notre contexte. Cela suffit néanmoins très souvent à éliminer efficacement les *outliers*.

Seuil d'erreur

Selon [Hartley and Zisserman, 2004], sous réserve que les différentes composantes de l'erreur de mesure, $e = (e_0, \ldots, e_i, \ldots, e_k)^\top$, des paramètres suivent une loi normale centrée en 0 et d'écart type σ connu $N(0, \sigma^2)$ il est possible de déterminer une valeur du seuil d'erreur acceptable, t, tel qu'un *inlier* sera effectivement classifié comme tel avec une probabilité α donnée.

On note $e^\top \cdot e = e^2$, on cherche t tel que :

$$e^2 < t^2 \tag{2.27}$$

On note y :

$$y = \frac{e^2}{\sigma^2} = \sum_i \left(\frac{e_i - 0}{\sigma} \right)^2 \tag{2.28}$$

Puisque l'erreur e est une réalisation de k variables aléatoires suivant une loi normale, la variable aléatoire y suit une loi du χ^2 à k degrés de libertés.

La fonction de répartition de la loi du χ^2, $F_k(.)$ nous donne la probabilité p selon laquelle une variable X est inférieure à une valeur x donnée :

$$F_k(x) = P(X < x) = p \tag{2.29}$$

Nous cherchons à déterminer t^2 tel que pour "$\alpha\%$" des *inliers* leur distance e^2 associée y soit inférieure.

$$P(e^2 < t^2) = P(y < \frac{t^2}{\sigma^2}) = \alpha \tag{2.30}$$

Ainsi $\frac{t^2}{\sigma^2} = F_k^{-1}(\alpha)$, donc le seuil de l'erreur acceptable est égal à :

$$t^2 = F_k^{-1}(\alpha) \cdot \sigma^2 \tag{2.31}$$

Dans [Hartley and Zisserman, 2004], les auteurs préconisent de fixer α à la valeur 0.95. Dans le tableau 2.3 nous reprenons les valeurs calculées par Hartley et Zisserman.

Degrés de liberté	t^2
1	$3.84 \cdot \sigma^2$
2	$5.99 \cdot \sigma^2$
3	$7.81 \cdot \sigma^2$

TABLE 2.3 – Seuil d'erreur fonction de la variance des erreurs de mesures selon le nombre de degrés de libertés de la transformation pour $\alpha = 0.95$. Les valeurs sont reprises de [Hartley and Zisserman, 2004].

Dans notre implémentation en Matlab de notre *SLAM* nous avons utilisé l'implémentation du *RANSAC* de P.D. Kovesi [2]. Dans notre implémentation en C++ nous l'avons écrit sous forme de template.

2.10 Synthèse

Nous avons vu dans ce chapitre comment extraire et mettre en correspondance des mesures depuis les images acquises par la caméra, puis comment filtrer ces appariements afin d'en éliminer les *outliers*. Depuis ces correspondances nous avons présenté deux méthodes d'estimation de matrice essentielle. Nous avons montré comment à l'aide d'une matrice essentielle il est possible de déterminer la pose relative, *i.e.* sans facteur d'échelle, entre deux caméras. Nous avons vu comment construire la carte en triangulant les mesures images à partir des poses de la caméra. Nous avons esquissé le principe de l'ajustement de faisceaux pour affiner les estimations de la caméra et de la carte, et expliqué que dans le cadre du *SLAM* cela peut servir à estimer la pose de la caméra à partir de correspondances avec la carte.

Nous avons à présent tous les outils nécessaires à l'élaboration d'un *SLAM* visuel. Nous allons à présent examiner les capteurs que nous employons pour le *SLAM* monoculaire et le *SLAM* multi-capteurs.

2. `http://www.csse.uwa.edu.au/~pk/research/matlabfns/`

Chapitre 3

Capteurs et calibrations

Nous présentons dans ce chapitre les différents capteurs que nous avons employés dans nos implémentations de *SLAM*. Nous présentons également les méthodes de calibration que nous avons utlisées.

Nous commençons par examiner le processus de formation de l'image d'une caméra et les paramètres extrinsèques et intrinsèques le régissant. Nous résumons ensuite les grandes lignes de la méthode de calibration permettant de déterminer ces paramètres que nous employons. Puis nous présentons le capteur de profondeur que nous utilisons et développons une méthode de calibration pour estimer les paramètres reliant les repères de celui-ci et d'une caméra. Ensuite, nous présentons la centrale inertielle que nous employons et précisons quelles informations issues de celle-ci nous intéressent. Finalement, nous développons une méthode de calibration pour estimer la rotation reliant les repères de la centrale et d'une caméra.

3.1 Caméra RGB

Dans le cadre d'un *SLAM* monoculaire, nous n'utilisons qu'une seule caméra. Dans le cadre de *SLAM* multi-capteurs, la caméra restera le capteur principalement utilisé. Nous présentons dans un premier temps la modélisation la plus courante du mécanisme de formation d'une image. Puis nous décrivons brièvement une méthode de calibration usuelle. La calibration permet d'estimer les différents paramètres impliqués dans cette modélisation.

3.1.1 Modèle du sténopé pour une caméra

Le modèle du sténopé est une représentation assez simple de la formation d'une image dans une caméra. Dans ce modèle un point X de l'espace, $\mathbb{R}^3$, se projette linéairement sur le plan image de la caméra en un point m au travers d'un trou infinitésimal.

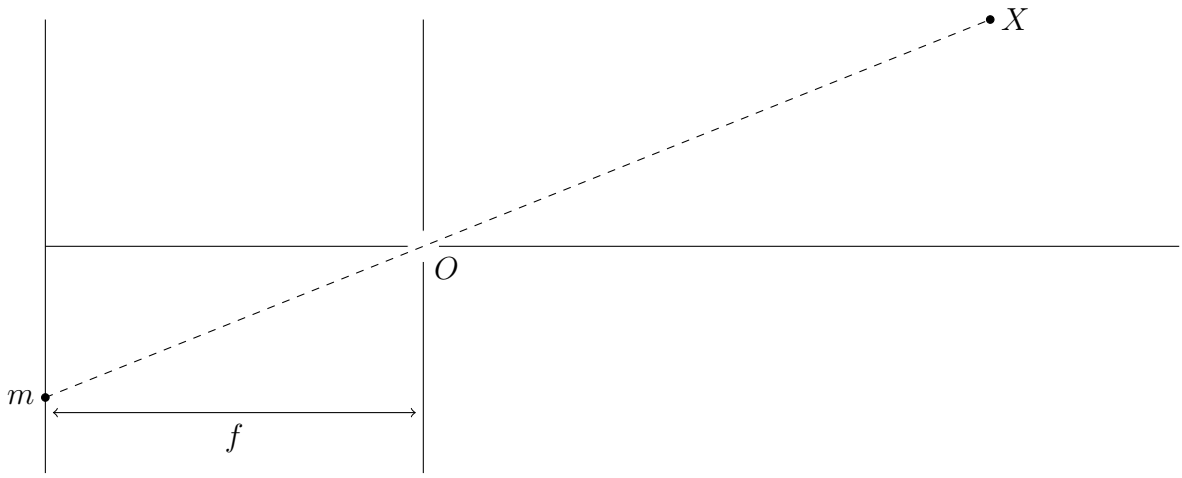


FIGURE 3.1 – Représentation de la projection d'un point de l'espace au travers d'un trou infinitésimal dans le plan image de la caméra.

Ce modèle physique de projection est équivalent, au signe près, à la modélisation mathématique suivante :

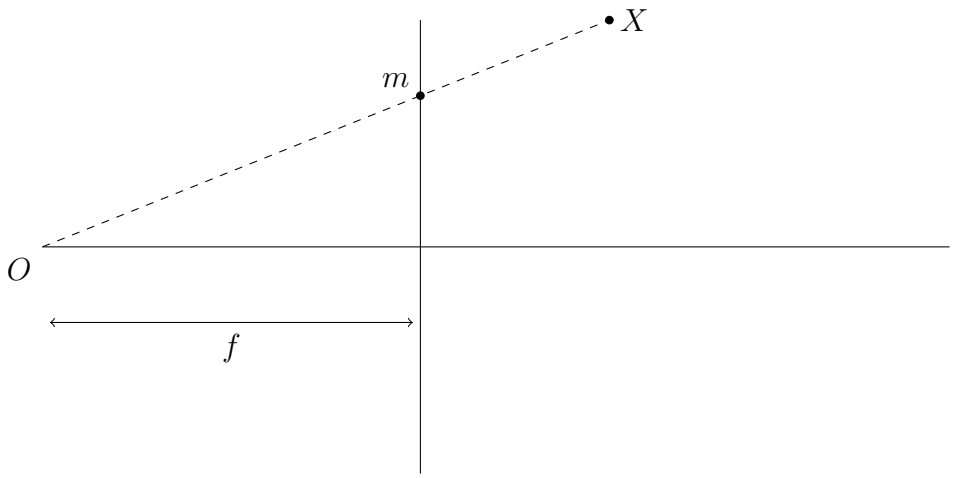


FIGURE 3.2 – Représentation de la projection de perspective d'un point de l'espace.

Selon ce modèle, un point de l'espace se projette donc dans l'image selon une projection de perspective.

Paramètres extrinsèques

Dans ce schéma de projection, le point de l'espace imagé doit être placé dans le repère de la caméra. Soit un point X^w de $\mathbb{R}^3$ défini dans un repère global W que l'on souhaite replacer dans le repère C de la caméra. On suppose que le déplacement de la caméra dans l'espace suit une transformation rigide. Alors la transformation des axes de W en ceux de C est caractérisée par un changement d'orientation et de position. Le changement de base associé au replacement de X^w dans C est donc paramétré par une rotation et une translation. On appelle **pose** de la caméra ou encore **paramètres extrinsèques** cette modification d'orientation et cette translation.
Soit R la matrice de rotation permettant de transformer les axes de W en ceux de C. Soit T la translation permettant de déplacer l'origine de W en celle de C, le centre optique de la caméra. On suppose que R est appliquée préalablement à T. Notons P la matrice réalisant les deux transformations :

$$P = \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} \tag{3.1}$$

La matrice P ainsi définie est la matrice de passage de la base W à la base C. La transformation que subit X^w lors de son expression dans C est donc

$$X^c = P^{-1} \cdot X^w = \begin{pmatrix} R^\top & -R^\top \cdot T \\ 0 & 1 \end{pmatrix} \cdot X^w \tag{3.2}$$

Dans la suite, pour des raisons de commodité, nous représenterons la pose d'une caméra par la matrice P_w^c permettant de transformer X^w en X^c, soit P^{-1} telle qu'explicitée dans 3.2 :

$$P_w^c = \begin{pmatrix} R_w^c & T_w^c \\ 0 & 1 \end{pmatrix} \tag{3.3}$$

On notera généralement C pour désigner P_w^c.

Paramètres intrinsèques

Une fois le point X exprimé dans C, il est projeté dans le plan image de la caméra. On note :

$$X^c = \begin{pmatrix} x & y & z \end{pmatrix}^\top \tag{3.4}$$

Le plan image se situant à la distance focale f du centre optique le point subit la transformation :

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} f \cdot x/z \\ f \cdot y/z \\ f \end{pmatrix} \tag{3.5}$$

Le point s'image donc en $\begin{pmatrix} f \cdot x/z & f \cdot y/z \end{pmatrix}^\top$.

En utilisant les coordonnées homogènes, on représente la transformation sous forme matricielle :

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} f \cdot x/z \\ f \cdot y/z \\ 1 \end{pmatrix} \sim \begin{pmatrix} f \cdot x \\ f \cdot y \\ z \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.6}$$

Une fois le point projeté dans le plan image, il reste à modéliser la transformation de ses coordonnées dans l'espace des capteurs photo-sensibles de la caméra pour obtenir ses coordonnées pixelliques. Les capteurs sont organisés en grille et nous nous plaçons dans le cas idéal où les axes de celle-ci

sont orthogonaux. Généralement l'origine de l'image pixellique se situe à l'extrémité supérieure gauche de la grille et les cases de la grilles sont rectangulaires. Ainsi, entrent en jeu deux transformations : une translation du centre de l'image et une mise à l'échelle anisotropique. Alors les coordonnées pixelliques sont obtenues par :

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} k_1 & 0 & u_0 \\ 0 & k_2 & v_0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \frac{1}{z} \cdot \begin{pmatrix} f \cdot x \\ f \cdot y \\ 1 \end{pmatrix} = \frac{1}{z} \cdot \begin{pmatrix} f \cdot k_1 & 0 & u_0 & 0 \\ 0 & f \cdot k_2 & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.7}$$

Usuellement on note la matrice des paramètres intrinsèques K :

$$K = \begin{pmatrix} f \cdot k_1 & 0 & u_0 & 0 \\ 0 & f \cdot k_2 & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.8}$$

Le processus complet de la formation d'image est :

$$s \cdot \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K \cdot P_w^c \cdot X^w \tag{3.9}$$

Distorsions

En réalité les ouvertures de caméras ne sont pas des trous infinitésimaux. En effet, de telles ouvertures ne laisseraient passer qu'une quantité infinitésimale de lumière que les capteurs ne pourraient pas percevoir. A la place les ouvertures sont constituées de lentilles pour concentrer les rayons lumineux afin d'en obtenir une quantité suffisante tout en pouvant contrôler leur point de convergence. La contre-partie est que la linéarité du modèle du sténopé n'est plus respectée et des distortions apparaissent lors de la formation de l'image. Ces distorsions sont de plusieurs types, notamment radial et tangentiel.
La prédiction de la projection d'un point dans l'image est donc erronnée si l'on ne prend en compte que le modèle du sténopé. Le redressement de l'image est le nom de la procédure qui consiste à transformer une image distordue en celle que le modèle du sténopé produirait. Dans le reste de ce

manuscrit, nous supposerons que les images sont redressées, ce qui peut être réalisé efficacement en utilisant la bibliothèque logicielle OpenCV [1].

3.1.2 Caractéristiques et limitations

Les caméras sont des capteurs peu chers et apportant beaucoup d'information. Néanmoins, celle-ci n'est qu'une projection de $\mathbb{R}^3$ dans $\mathbb{R}^2$, une dimension entière est perdue : la profondeur est inaccessible. Les fréquences d'acquistion sont élevées puisque les caméras produisent généralement au minimum 30 images par seconde. Pour une telle fréquence une image se forme en $\sim$ 33ms, pour des mouvements suffisament rapides des effets de flous peuvent se former.

3.1.3 Calibration de caméra

On appelle calibration une procédure permettant d'estimer les paramètres intrinsèques d'une caméra.
Plusieurs méthodes de calibration ont été proposées dans la littérature. L'une d'entre elles a été largement popularisée par son implémentation au travers d'une *toolbox* pour *Matlab* puis dans la bibliothèque logicielle *OpenCV*. Cette méthode est celle de Zhang proposée dans [Zhang, 2000]. Comme la plupart des méthodes elle s'appuie sur l'utilisation d'une mire planaire représentant un damier dont les dimensions sont connues. La mire est observée depuis plusieurs poses distinctes. Les coins du damier sont détectés à chaque prise de vue et sont les points utilisés dans l'estimation de la transformation d'une image à une autre. Bien que l'objectif soit avant tout de déterminer les paramètres intrinsèques, il est nécessaire de déterminer également les extrinsèques puisque différents points de vues sont utilisés.

Le caractère planaire de la mire permet de représenter la transformation des points de la mire réelle vers une des images sous la forme d'une homographie H, voir l'annexe C.1. Il permet également en toute généralité de fixer la troisième coordonnée des points de la mire à 0 et ainsi d'exprimer la chaîne de projection présentée dans 3.1.1 d'une manière simplifiée et de la mettre directement en relation avec H.

Cette relation entre les matrices formant la chaîne de projection entre les points de la mire et la matrice d'homographie permet d'exprimer deux rela-

1. http ://opencv.org

FIGURE 3.3 – La caméra couleur et profondeur *Xtion Pro Live* d'*ASUS*

tions dépendant seulement de l'homographie et des paramètres intrinsèques de la caméra. A partir de trois prises de vues distinctes il est alors possible de déterminer les paramètres intrinsèques. Une fois ceux-ci déterminés, il est possible de retrouver les paramètres extrinsèques liés à chaque prise de vue. Finalement les résultats sont affinés par ajustement de faisceaux, prenant possiblement en compte le phénomène de distorsion. Les paramètres le définissant étant considérés de faible grandeur, une initialisation à zéro permet à l'optimisation de les déterminer avec précision.

3.2 Caméra de profondeur

Pour pallier certaines limitations du *SLAM* monoculaire nous avons employé un capteur de profondeur dans un *SLAM* multi-capteurs. Dans la partie 5.1, nous nous en servons pour estimer la profondeur de points lorsque le mouvement de la caméra ne se prête pas au processus de triangulation.

3.2.1 Présentation

Il existe plusieurs types de caméras de profondeur. Nous nous intéressons aux capteurs récemment introduits par la société *PrimeSense* car à la différence de leurs prédécesseurs, ceux-ci sont relativement abordables financièrement. Le capteur de profondeur que nous utilisons est une caméra *Xtion Pro Live* de la société *ASUS*, sa fréquence d'acquisition est de 20 images par seconde. Une photographie de la caméra est produite à la figure 3.3.

Ces dispositifs sont équipés d'un projecteur infra-rouge, d'une caméra infra-rouge et éventuellement d'une caméra couleur. Le couple projecteur-

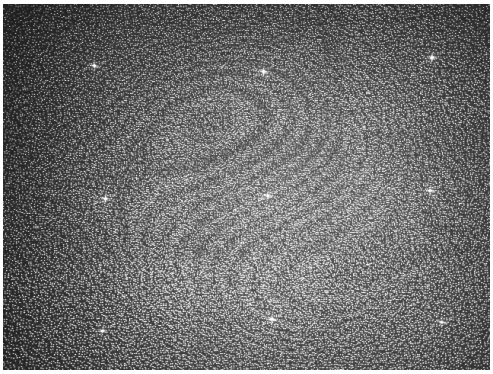


FIGURE 3.4 – Observation par la caméra infra-rouge du schéma structuré projeté

caméra infra-rouge peut être considéré comme une caméra capable de percevoir la profondeur de la scène observée. Le projecteur émet une image de référence structurée connue du capteur. L'observation de sa projection sur la scène et sa mise en correspondance par un circuit intégré avec la référence permet d'estimer la profondeur du pixel par triangulation. La figure 3.4 montre une image captée par la caméra infra-rouge avant son interprétation comme image de profondeur.

3.2.2 Limitations

La portée des caméras de profondeur est limitée selon le vendeur à des distances comprises entre 0.8 m et 3.5m. S'il est souvent possible d'obtenir la profondeur pour des distances approximativement comprises entre 0.4 m et 6 m, au delà de 3m les mesures s'avèrent fortement erronnées. Cette limitation est certainement dûe à la procédure de mise en correspondance, celle-ci étant réalisée localement autour de chaque pixel.

Par ailleurs, si la scène ne limite pas à un simple plan, le processus de mise en correspondance n'aboutit pas sur toutes les zones de l'image. Les profondeurs de ces zones ne sont alors pas accessibles. Sur presque toute surface, le reflet du projecteur bruite les observations, la profondeur de la zone de reflet n'est alors pas estimable. De même en est il pour les surfaces spéculaires : puisque réfléchissant le schéma projeté, ce dernier n'est pas visible. Enfin, la limitation la plus importante tient aux environnements favorables à l'utilisation du capteur. Celui-ci ne peut être généralement uti-

lisé qu'en environnements intérieurs, les rayonnements infra-rouges du soleil perturbant l'image projetée.

Les angles de vues annoncés sont de 58° horizontalement, 45° verticalement et 70° diagonalement. La précision annoncée en profondeur est de l'ordre du centimètre.

3.2.3 Calibration extrinsèque caméra de profondeur - caméra couleur

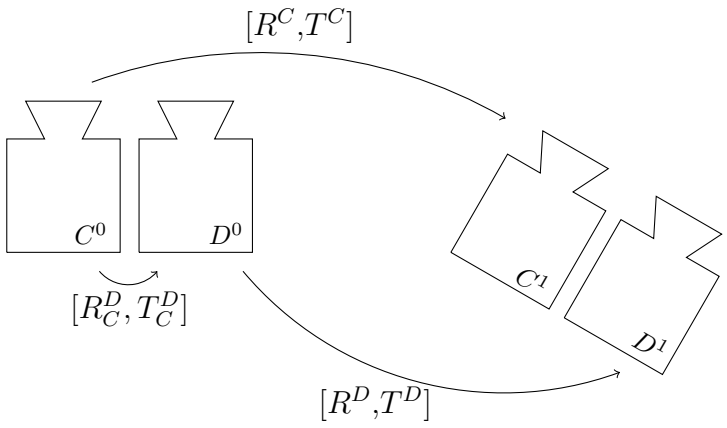


FIGURE 3.5 – Un couple de capteurs de couleur et profondeur.

Nous devisons à présent une procédure de calibration extrinsèque pour deux capteurs, profondeur et couleur, rigidement liés. Nous recherchons donc la matrice de rotation et le vecteur de translation reliant les repères des capteurs. Une fois ces inconnues déterminées il est possible pour chaque point d'intérêt de déterminer la mesure de profondeur correspondant. Pour un point de l'image de couleur cela peut être fait en recherchant le long de la ligne épipolaire définie par celui-ci le point dans l'image de profondeur dont l'erreur de reprojection est minimale.

La procédure de calibration que nous employons est une procédure de calibration stéréo basée sur la calibration de Zhang [Zhang, 2000]. La mire employée doit être détectable simultanément par le capteur de profondeur et la caméra couleur. Nous proposons pour cela d'utiliser une mire planaire de couleur claire percée de trous située à distance d'un fond noir. L'image de profondeur n'étant pas nette, un motif de type damier peut difficilement

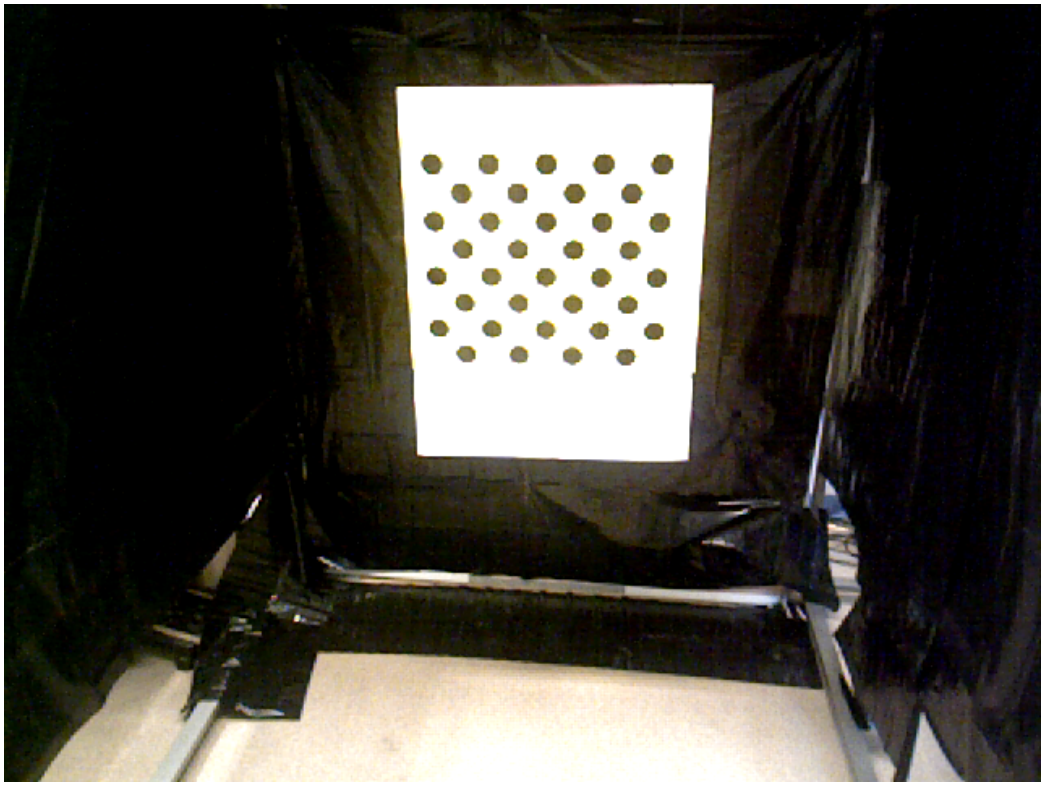


FIGURE 3.6 – Scène de calibration pour couple de caméra couleur - caméra de profondeur

être employé, la détection de coins et/ou d'arêtes ne pouvant être assurée avec précision. Nous proposons d'utiliser un motif composé de cercles et de considérer leurs centres comme points de calibration, voir la figure 3.6.

Calibration individuelle des capteurs

Pour l'image de couleur, la détection des points est faite avec la bibliothèque *OpenCV*. Elle est effectué par une détection de *blobs* dont sont extraits les centres.

Nous avons constaté que la détection de *blobs* de la bibliothèque fonctionne mal pour les images de profondeur, celles ci s'avérant trop bruitées. Nous avons alors développé notre propre procédure. Notre détection des points de la mire passe par une détection du plan. Celle-ci peut être réalisée par une sélection manuelle de valeurs de profondeur et une délimitation du

plan dans l'image de profondeur. Elle peut être aussi automatisée par une détection de plans via un algorithme *RANSAC*. Le nombre de plans à détecter dépend de la scène. Dans le cas de la figure 3.6 cinq suffisent. Le plan de calibration est sélectionné comme étant celui le plus proche de la caméra parmi les deux dont les composantes en z des normales sont les plus importantes. Une fois le plan détecté, nous binarisons l'image de profondeur en mettant à 0 les pixels n'en faisant pas partie et à la valeur maximale les autres. Sur l'image binarisée nous appliquons un algorithme de remplissage par diffusion pour détecter les blobs de la mire. Les points de contrôle pris en compte pour la calibration sont les barycentres des blobs.

Une fois les points détectés sur les images de couleur et profondeur pour plus de deux prises de vue on applique pour chaque capteur la méthode de Zhang, présentée dans la partie 3.1.3. A l'issue des calibrations individuelles nous avons identifié deux façons d'initialiser la pose relative des capteurs.

Initialisation de la pose relative par moyennage

Les résultats de calibration les poses de chaque capteur sont connues pour chaque prise de vue. Nous pouvons alors former pour chaque prise de vue la pose relative. On note $C = \begin{pmatrix} R^c & T^c \end{pmatrix}$ et $D = \begin{pmatrix} R^d & T^d \end{pmatrix}$ les poses des caméras couleur et profondeur. La pose relative est donnée par :

$$P_c^d = \begin{pmatrix} R^d \cdot R^{c\top} & T^d - R^d \cdot R^{c\top} \cdot T^c \end{pmatrix} \tag{3.10}$$

Puisque nous avons N prises de vues, nous obtenons N poses relatives. Nous initialisons alors la pose relative à la moyenne de ces poses. Pour moyenner les matrices de rotations nous les convertissons en quaternions.

Initialisation de la pose relative d'après une matrice d'homographie

A l'issue de la calibration les paramètres intrinsèques ainsi que les distortions de chaque caméra, couleur et profondeur, sont connus. Cela permet de redresser les images puis d'appliquer la matrice des paramètres intrinsèques inverse à chaque point de calibration. Deux ensembles, correspondant aux images couleur et images de profondeur, de points en coordonnées normalisées sont ainsi obtenus. Ces ensembles sont reliés par une matrice d'homographie. De cette matrice d'homographie nous pouvons extraire quatre poses relatives en utilisant la méthode présentée par Faugeras dans [Faugeras and Lustman, 1988]. Ces poses sont définies au facteur

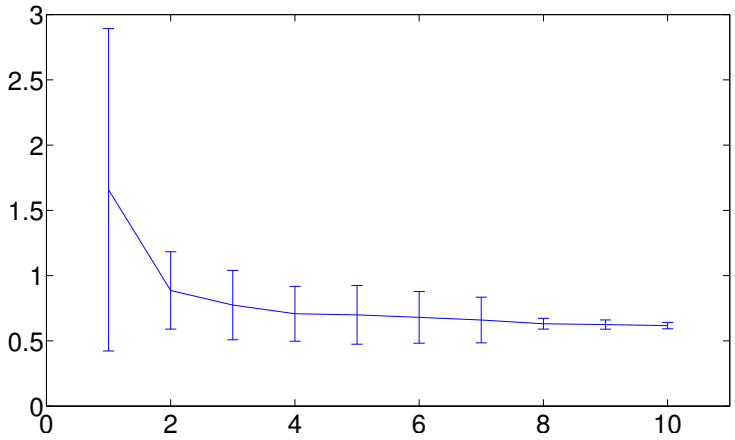


FIGURE 3.7 – Erreurs et écarts types de reprojection, en pixel, en fonction du nombre de paires d'images utilisées pour la calibration

d'échelle. Celui-ci est obtenu à partir d'une correspondance entre un point dans $\mathbb{R}^3$ de la mire dans le référentiel de la caméra de couleur et le point de l'image de profondeur correspondant. Parmi ces quatre poses nous choisissons celle dont le facteur d'échelle présente le moins de variation pour initialiser la pose relative des capteurs.

Raffinement de la pose relative

On note X_{cj}^{i} la position dans l'espace du point j dans le repère de la caméra de couleur i, et m_{dj}^{i} la mesure de ce point dans l'image de la caméra de profondeur. Une fois une initialisation de la pose relative obtenue nous l'affinons par un ajustement de faisceaux. La fonction de coût minimisée est :

$$\min_{P_c^d} \sum_i \sum_j \left\| m_{dj}^{i} - proj(X_{cj}^{i}, P_c^d) \right\|^2 \tag{3.11}$$

Nous avons constaté que les deux méthodes d'initialisation de pose relative convergent généralement vers la même pose.

Application

Nous avons acquis 33 paires d'images de couleur et profondeur avec les caméras de couleur et de profondeur de l'appareil. Pour chaque paire de vues $v = \{1 \dots 33\}$ nous avons :

- sélectionné aléatoirement $\{0 \dots 9\}$ paires d'images en plus de v, constituant un ensemble de $l = \{1 \dots 10\}$ paires.
- estimé la pose relative à partir des calibrations de chaque capteur effectuées avec les l images.
- évalué cette calibration sur les $r = 33 - l = \{32 \dots 23\}$ paires d'images restantes.

La qualité de la pose estimée est mesurée par l'erreur moyenne de reprojection des points de l'image couleur dans $\mathbb{R}^3$ dans l'image de profondeur. Soit :

$$\Delta z = \frac{1}{m \cdot n} \cdot \sum_{i=1}^{m} \sum_{j=1}^{n} \left\| m_{d_j}^{i} - proj(X_{c_j}^{i}, P_c^d) \right\| \tag{3.12}$$

Les résultats des calibrations sont disponibles à la figure 3.7. On y observe que la précision devient rapidement, à partir de 8 paires d'images, inférieure à 1 pixel. La matrice $P_c^d = [R_c^d, t_c^d]$, avec les translations données en centimètres, obtenue est :

$$P_c^d = \begin{pmatrix} 1.0000 & 0.0015 & -0.0010 & -0.0579 \\ -0.0015 & 1.0000 & 0.0014 & 0.0497 \\ 0.0010 & -0.0014 & 1.0000 & 0.1862 \end{pmatrix} \tag{3.13}$$

Les caméras de profondeur et couleur sont donc vraisemblablement calibrées d'usine afin de recaler les images. Ce recalage est assez précis puisque que nous avons constaté que les distances, sur des points saillants, entre les deux images sont de l'ordre de quelques pixels. Pour des considérations d'allègement des calculs, nous nous sommes donc basés sur cette hypothèse de recalage des images lors de nos utilisations des données de profondeur.

3.2.4 Paramètres intrinsèques : distorsion de profondeur

Plusieurs travaux ont observé des distorsions sur les mesures de profondeur, [Herrera et al., 2012], [Smisek et al., 2013], [Teichman et al., 2013].

Dans [Smisek et al., 2013], Smisek *et al* observent que le pas d'échantillonnage des profondeurs mesurées croît avec la distance de la scène au capteur. Ils constatent aussi la présence d'une distorsion des mesures dépendant de la position du pixel.

Dans [Herrera et al., 2012], Herrera *et al* proposent d'intégrer un modèle spatial, une image, de cette distorsion dans un processus visant à estimer la

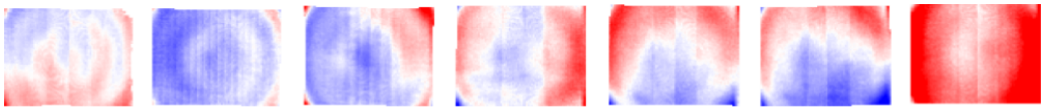


FIGURE 3.8 – Quelques images de coefficients de distorsion obtenue par Teichman *et al*, [Teichman et al., 2013].

profondeur à partir de mesures de disparité. Nous n'utilisons cependant pas le même pilote, le notre permettant d'accéder directement aux données de profondeur.

Dans [Teichman et al., 2013], Teichman *et al* proposent d'effectuer un *SLAM* dense avec profondeur puis d'alterner l'optimisation de la carte et l'optimisation de 5, correspondant à diverses profondeur, images de coefficients de distorsion de tailles 80×80 pixels. Les auteurs constatent que les distorsion croissent avec la distance du capteur à la scène. La figure 3.8 présente quelques images de distorsion obtenues par les auteurs pour différents capteurs.

Puisque nos scènes d'intérieur présentent une faible profondeur, d'après les travaux précédents les distorsions sont faibles également. Il ne nous est alors pas apparu crucial de corriger ces distorsions.

3.3 Centrale inertielle

Finalement, le dernier capteur que nous avons employé est une centrale inertielle. Dans la partie 5.4, nous nous en servons pour alléger le processus d'estimation de la géométrie épipolaire entre deux vues.

3.3.1 Caractéristiques et limitations

Une centrale inertielle est un capteur de mouvements mesurant la cinématique d'un système sans nécessiter de repère externe.

Le capteur que nous utilisons est une centrale *MTi* de la société *XSENS*, figure 3.9, dotée de gyroscopes, accéléromètres et magnétomètres. Ces capteurs permettent de mesurer :

- l'accélération tangentielle en $m \cdot s^{-2}$
- la vitesse angulaire en degrés par seconde
- le champ magnétique en milligauss

FIGURE 3.9 – La centrale inertielle *MTi* d'*Xsens*

Les données sont acquises à très haute fréquence, de l'ordre de 100Hz. Indiquant le nord magnétique terrestre, les magnétomètres permettent d'exprimer l'orientation dans un repère global.

La centrale *MTi* est dotée d'un processeur permettant d'appliquer un filtre de Kalman sur les données brutes afin d'en extraire une mesure d'orientation de qualité. Les accéléromètres et magnétomètres sont utilisés pour compenser la dérive d'intégration des données des gyroscopes. Nous avons constaté que les mesures d'orientations sont plutôt précises. Mais le caractère relatif des mesures employées la condamne à dériver, c'est une première limitation. Il nous est donc apparu préférable de ne pas utiliser ces informations de manière absolue, mais plutôt de manière relative sur des intervalles de temps restreints.

Une seconde limitation vient de ce que des objets générant des champs électriques importants peuvent perturber la mesure du nord magnétique terrestre, et donc l'orientation estimée par le capteur.

Il est difficile d'obtenir une position, ou translation, à partir des données d'accélérations tangentielles. Ceci nécessiterait d'estimer itérativement la vitesse de l'appareil par intégration des mesures accélérationnelles. Celles ci étant bruitées, le bruit se propagerait et s'accumulerait au cours de ces itérations, faussant grandement la vitesse estimée. C'est la troisième limitation que nous identifions.

Nous restreignons alors notre usage des données inertielles aux seules informations de rotation.

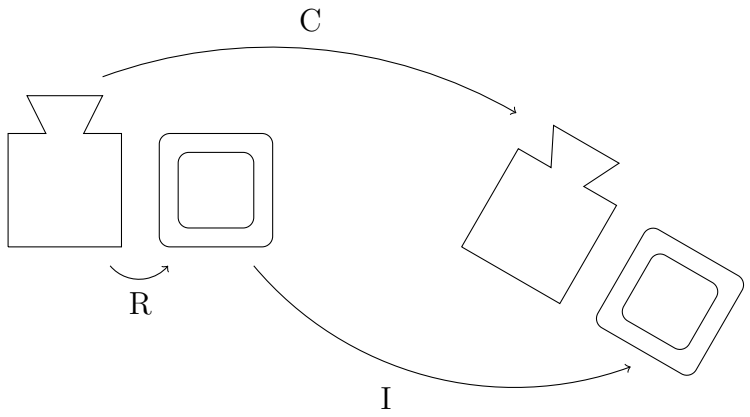


FIGURE 3.10 – La rotation entre la caméra et la centrale inertielle est estimée à partir des rotations de chaque capteur.

3.3.2 Calibration caméra-centrale inertielle

Pour utiliser les informations de rotation offertes par la centrale inertielle lors de l'estimation du mouvement de la caméra il est nécessaire de pouvoir les exprimer dans le repère de cette dernière. Nous présentons à présent la procédure de calibration que nous avons appliquée pour cela.

Méthode

Nous supposons le couple caméra-centrale inertielle rigidement lié. Comme nous souhaitons utiliser uniquement les mesures d'orientation de la centrale, estimer la seule rotation relative entre les capteurs nous suffit.

On note R la rotation entre la centrale inertielle et la caméra que nous cherchons à estimer. On suppose que cette matrice est la matrice de passage de la base de la centrale inertielle à celle la caméra. On note Q le quaternion associé. On rappelle que $**$ désigne le produit de quaternions.

On suppose les paramètres intrinsèques de la caméra connus. On imprime un mouvement au capteur hybride en observant une mire planaire. A chaque prise de vue de la caméra nous appliquons une méthode d'estimation de pose de type PnP, et enregistrons l'orientation donnée par la centrale inertielle. Pour deux prises de vues successives, pour chaque capteur nous formons alors une matrice de rotation relative : C pour la caméra et I pour la centrale

inertielle. On note Q_c et Q_i les quaternions associés. Nous avons la relation :

$$\begin{aligned} & C = R^\top \cdot I \cdot R \\ \Leftrightarrow \\ & Q_c = Q^{-1} ** Q_i ** Q \\ \Leftrightarrow \\ & Q ** Q_c - Q_i ** Q = 0 \end{aligned} \tag{3.14}$$

On pose :

$$Q = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \quad Q_i = \begin{pmatrix} a_i \\ b_i \\ c_i \\ d_i \end{pmatrix} \quad Q_c = \begin{pmatrix} a_c \\ b_c \\ c_c \\ d_c \end{pmatrix} \tag{3.15}$$

On développe 3.14 :

$$\begin{aligned} & Q ** Q_c - Q_i ** Q = 0 \\ \Leftrightarrow \\ & \underbrace{\begin{pmatrix} -a_i + a_c & b_i - b_c & c_i - c_c & d_i - d_c \\ -b_i + b_c & -a_i + a_c & d_i + d_c & -c_i - c_c \\ -c_i + c_c & -d_i - d_c & -a_i + a_c & b_i + b_c \\ -d_i + d_c & c_i + c_c & -b_i - b_c & -a_i + a_c \end{pmatrix}}_{\mathrm{S}} \cdot \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = 0 \end{aligned} \tag{3.16}$$

La matrice S ne peut pas être de rang plein, puisqu'il existe au moins une solution non nulle au système. On observe que la matrice est antisymétrique et a donc pour propriété d'être de rang pair, alors S est de rang 2.

Ainsi, il nous faut imprimer au moins deux mouvements au capteur hybride pour concaténer au moins deux matrices S. La matrice ainsi formée est de rang 4. La solution Q est obtenue comme la dernière colonne de la matrice $V^\top$ obtenue par décomposition en valeurs singulières.

Application

La figure 3.11 présente le capteur hybride que nous avons calibré. Nous avons acquis 38 images d'une mire de calibration et autant de mesures d'orientation de la centrale inertielle. Nous avons ensuite formé 37 paires de rotations relatives. Pour chaque paire $v = \{1 \dots 37\}$ nous avons :

– sélectionné aléatoirement $\{1 \dots 19\}$ paires de rotations en plus de v, constituant un ensemble de $l = \{2 \dots 20\}$ paires.

FIGURE 3.11 – Le couple de capteurs *Xtion Pro Live - MTi*

– estimé la rotation relative selon la méthode présentée.
– évalué cette calibration sur les $r = 37 - l = \{35 \dots 17\}$ paires de rotations restantes.

La précision de la calibration est évaluée selon l'erreur e définie comme la norme du vecteur formé de la distance, en degrés, entre les axes de la rotation relative de la caméra et sa prédiction à partir de celle de la centrale et de la rotation estimée reliant les capteurs. Notant C la rotation relative de la caméra, I celle la centrale inertielle, R la rotation reliant les capteurs et A_i la i-ème ligne de A, e est calculée comme :

$$e = \left\| \begin{pmatrix} \arccos\left(C_1 \cdot (R^\top \cdot I \cdot R)_1^\top\right) \\ \arccos\left(C_2 \cdot (R^\top \cdot I \cdot R)_2^\top\right) \\ \arccos\left(C_3 \cdot (R^\top \cdot I \cdot R)_3^\top\right) \end{pmatrix} \cdot \frac{180}{\pi} \right\|_2 \tag{3.17}$$

La figure 3.12 présente les résultats obtenus. On constate qu'à partir de 7 paires de rotations l'erreur devient proche de 1°. Elle n'y devient cependant jamais inférieure et l'écart type semble rester constant. Nous supputons que la dérive des orientations du capteur inertiel en est à l'origine.

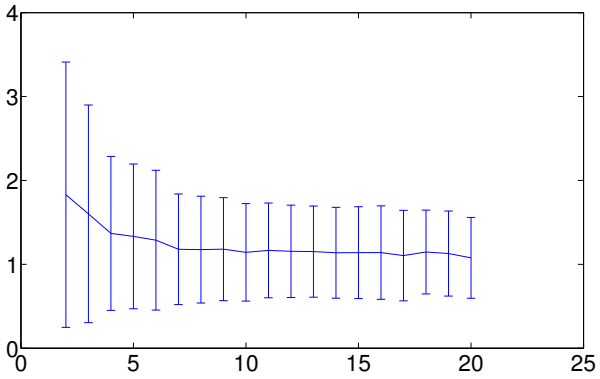


FIGURE 3.12 – Erreurs et écarts types, en degrés, de la distance entre les axes de la caméra et ceux prédits à partir de la centrale inertielle à l'aide de la transformation estimée, en fonction du nombre de paires de rotations relatives employées pour la calibration.

Ayant approximativement aligné les axes des deux capteurs nous obtenons pour R une matrice proche de l'identité :

$$R = \begin{pmatrix} 0.9993 & -0.0318 & 0.0202 \\ 0.0302 & 0.9969 & 0.0733 \\ -0.0225 & -0.0726 & 0.9971 \end{pmatrix} \tag{3.18}$$

3.4 Conclusion

Dans ce chapitre nous avons vu les différents capteurs que nous employons, une caméra, un capteur de profondeur et une centrale inertielle. Nous avons vu le processus de formation d'une image de caméra et une méthode de calibration pour estimer les paramètres qui le gouverne. Puis nous avons présenté le capteur de profondeur que nous employons et notre méthode de calibration servant à estimer la transformation rigide entre ce capteur et une caméra. Ensuite nous avons vu la centrale inertielle que nous utilisons et les raisons pour lesquelles nous nous limitons a en utiliser les seules informations d'orientation sur des intervalles de temps restreints. Enfin, nous avons présenté notre méthode de calibration de la transformation rigide entre la centrale inertielle et une caméra.

Nous pouvons à présent développer nos *SLAM* monoculaire et multi-capteurs.

Chapitre 4

Slam monoculaire

Nous présentons dans ce chapitre nos travaux sur le *SLAM* monoculaire. Nous commençons par présenter l'approche dont nous nous sommes inspirés, puis l'adaptation que nous en avons faîte. Nous en montrons ensuite deux applications. Puis nous en évaluons la précision de notre programme sur des séquences de mouvements *canoniques*. Suite à cela nous énumérons quelques unes des limites du *SLAM* monoculaire que nous avons observées. Enfin, en faisant l'hypothèse d'une reconnaissance de lieu, nous présentons nos contributions quant à la prise en compte d'une connaissance *a priori* pour réduire la dérive des estimations des poses antérieures à la relocalisation du système.

4.1 Le *SLAM* qui nous inspire

Les résultats présentés par Mouragnon *et al* dans [Mouragnon et al., 2006] apparaissent de très bonne qualité, comme en témoigne la figure 4.1. De plus l'approche de *SLAM* basée sur l'utilisation de caméras clés a été démontrée dans [Strasdat et al., 2010] plus intéressante que celles basées sur le filtrage. Nous avons alors choisi de reprendre ce schéma de *SLAM* et le décrivons à présent.

La méthode présentée suppose les paramètres intrinsèques de la caméra connus. Cela élimine des degrés de liberté dans la reconstruction de la scène et permet donc d'obtenir des résultats plus précis.

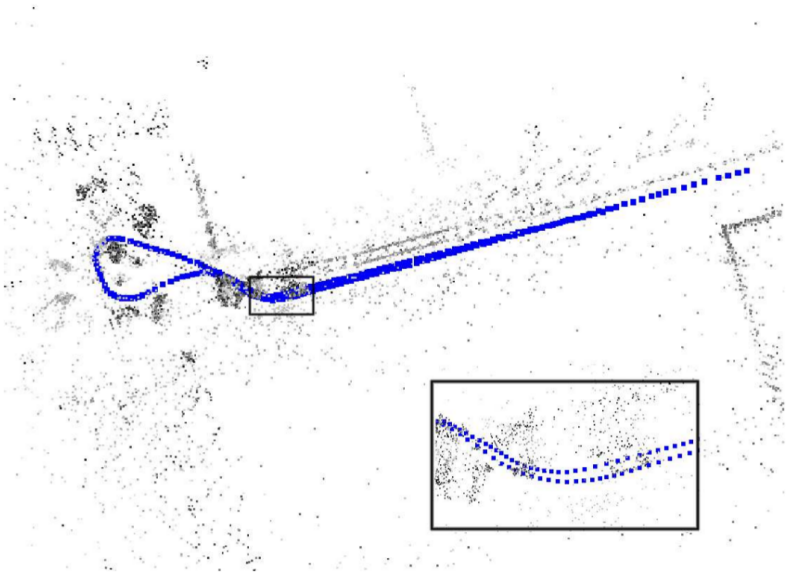


FIGURE 4.1 – Un résultat de la méthode présentée dans [Mouragnon et al., 2006], l'image en est tirée. La trajectoire est longue de 200m.

4.1.1 Initialisation

La première phase de l'algorithme est l'initialisation de carte et des trois premières caméras clés, elle est réalisée sans connaissance *a priori*.

La première caméra est la première caméra clé et définit le repère global dans lequel sont estimés les poses des caméras suivantes et les positions des points de la carte. Chacune des caméras est ensuite appariée avec elle. Lorsque pour la i-ème caméra la quantité d'appariements devient inférieure à un premier seuil, M, la $i-1$-ème caméra devient la deuxième caméra clé. Lorsque pour la j-ème caméra la quantité d'appariements devient inférieure à un second seuil, M', la $j-1$-ème caméra devient la troisième caméra clé.

L'estimation des poses des caméras clés 2 et 3 est réalisée simultanément au travers d'une processus *RANSAC*. Le processus d'estimation est le suivant : Un échantillon minimal de 5 appariements visibles entre les 3 caméras est sélectionné. A ces échantillons est appliqué un algorithme 5-points, [Nister, 2004], permettant d'estimer un ensemble, au maximum 10, de matrices essentielles correspondant à la pose relative entre les caméras clés 1 et 3. Pour chacune de ces matrices essentielles les quatre poses relatives potentielles sont

estimées ([Hartley and Zisserman, 2004], 2.3). Les 5 appariements sont triangulés selon les poses relatives obtenues. Celles qui ne placent pas les 5 points devant les deux caméras sont éliminées. La pose de la deuxième caméra clé est estimée à partir de 3 correspondances *3D-2D*, parmi les 5 échantillons de départ, selon l'algorithme de pose de Grunert, [Haralick et al., 1994].

A l'issue du *RANSAC*, tous les appariements sont triangulés puis les points et les poses sont raffinés par ajustement de faisceaux. Un schéma sous forme de graphe de la procédure d'initialisation est disponible à la figure 4.2.

4.1.2 Boucle de traitement

A la réception de chaque nouvelle caméra des points d'intérêts sont extraits. Pour chaque point d'intérêt de l'image de la caméra clé précédente une recherche d'appariement est effectuée dans une fenêtre de l'image courante.

Si le nombre d'appariements est supérieur au seuil M la pose de la caméra est estimée en ne prenant en compte que les points appariés dont la profondeur a déjà été estimée, c'est à dire ayant déjà été observés par au moins deux caméras clés. Ces points sont utilisés dans un *RANSAC* combiné à une estimation de pose de Grunert pour obtenir une première estimation de la pose de la caméra. Cette estimation est ensuite raffinée par ajustement de faisceaux.

Si le nombre d'appariements est inférieur au seuil M une nouvelle caméra clé doit être définie. Celle-ci est la caméra précédente. Les points appariés de cette caméra de profondeur encore inconnue sont triangulés, et un ajustement de faisceaux local est appliqué. Les paramètres ajustés sont ceux des n dernières caméras clés et ceux des points observés dans ces caméras, et $N - n$ caméras clés sont fixes, nous les appelons caméras observatrices. Toutefois dans le cas où moins de 20 caméras clés ont été estimées, l'ajustement de faisceaux est appliqué globalement. Les auteurs justifient ce choix en expliquant que le système est encore en cours d'initialisation et que la quantité de paramètres est encore suffisamment faible pour que le coût calculatoire de l'ajustement de faisceaux soit acceptable.

Un schéma sous forme de graphe de la boucle principale de l'algorithme est disponible à la figure 4.3

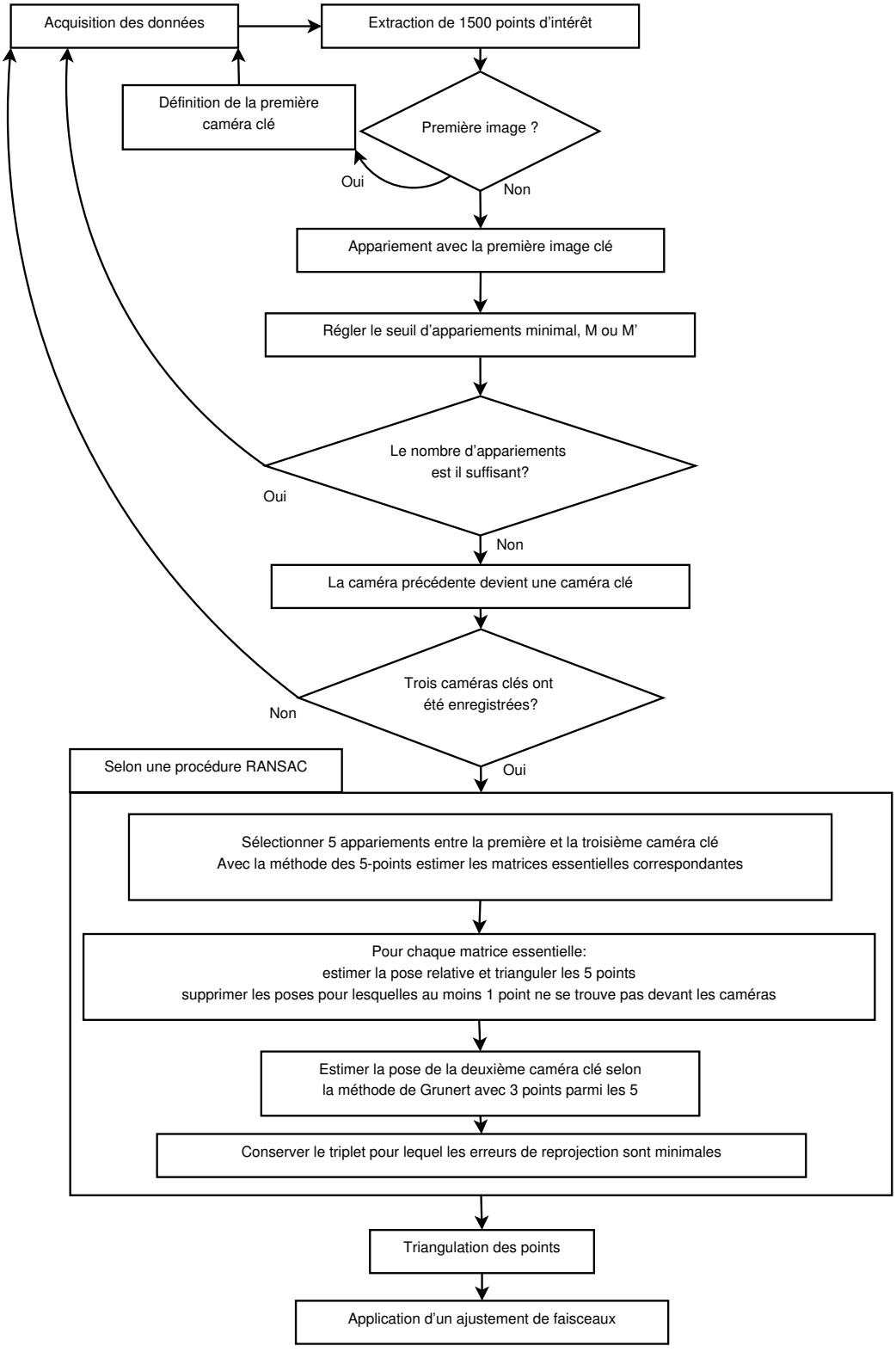


FIGURE 4.2 – La procédure d'initialisation de [E. Royer, 2005] et [Mouragnon et al., 2006]

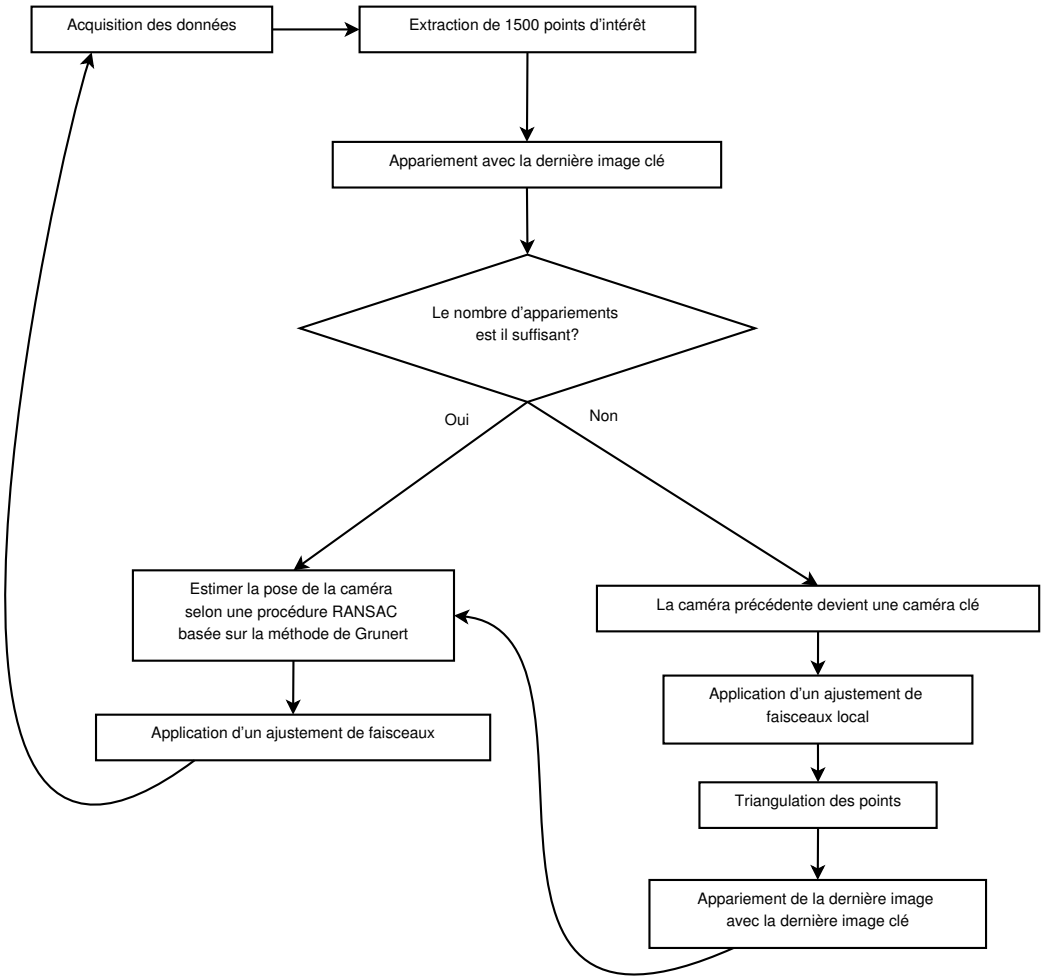


FIGURE 4.3 – La boucle principale présentée dans de [Mouragnon et al., 2006]

4.1.3 Ajustement de faisceaux local

Introduit dans la communauté par Mouragnon *et al* dans l'article en question, l'ajustement de faisceaux local est une application d'un ajustement de faisceaux dans lequel seule une partie de l'ensemble des poses de caméra est ajustée tout en prenant en compte des poses de caméra non ajustables dans l'évalution des erreurs des reprojections. Une illustration de l'ajustement de faisceaux local tirée de [Mouragnon et al., 2006] est visible à la figure 4.4. Selon [Mouragnon et al., 2006] les auteurs minimisent l'erreur quadratique

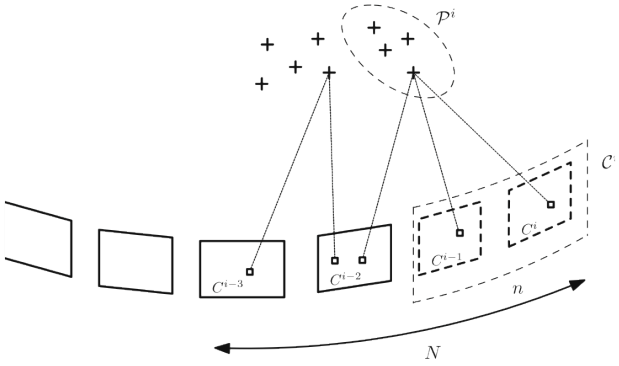


FIGURE 4.4 – Schéma de l'ajustement de faisceaux local sur fenêtre glissante. Les poses de n caméras sont optimisées selon les erreurs de reprojection observée dans N caméras. Illustration tirée de [Mouragnon et al., 2006].

de reprojection sans faire usage de fonction de coût robuste aux *outliers*. Une étude de l'erreur de l'estimation de pose comparée à des mesures GPS montre que l'augmentation du nombre n de caméras optimisées accroît la qualité des estimations. Par contre, l'effet de l'augmentation du nombre de caméras observatrices est plus difficilement caractérisable, des minima locaux semblant apparaître.

4.1.4 Points d'intérêt

Les auteurs utilisent des détecteurs de coins de *Harris*, [Harris and Stephens, 1988]. Selon Royer dans [E. Royer, 2005] 1500 points sont extraits. Les points sont appariés entre deux images par calul d'un score de corrélation croisée. En cas d'appariements multiples celui ayant le score le plus élevé est conservé. Les points de l'image 1 ne sont pas comparés avec l'ensemble de ceux de

l'image 2, mais seulement avec ceux faisant partie d'une région d'intérêt. La définition de cette région d'intérêt n'est cependant renseignée ni par Mouragnon ni par Royer. Les étapes de détection et appariements sont rendues temps réel par une implémentation basée sur l'utilisation d'instructions assembleur parallèles *SIMD*.

4.2 Notre *SLAM* monoculaire

Nous présentons maintenant les modifications que nous avons apportées à la méthode précédemment décrite. Notre algorithme est résumé en pseudo-code à l'algorithme 2. Pour effectuer les ajustements de faisceaux nous utilisons la bibliothèque *SSBA*, de C. Zach [Zach,]. Les fonctions de coût y sont robustifiées avec une fonction de *Huber*, voir la partie D.1.

4.2.1 Points d'intérêt

Alors que Mouragnon *et al* emploient leur *SLAM* sur des séquences vidéo d'une caméra montée sur un véhicule nous visons à utiliser le *SLAM* dans des applications de réalité augmentée. Par conséquent la caméra est dans notre cas susceptible d'effectuer des rotations autour de l'axe de visée, ce qui n'est pas le cas si elle est montée sur un véhicule. Il faut donc que les détecteurs et descripteurs de points d'intérêt que nous employons présentent un certain degré d'invariance aux rotations. Un descripteur de type *patch* dont la comparaison est effectuée par un calcul de distance avec un autre ne peut convenir.

Dans un premier temps nous avons employé des points *SIFT*. Ceux-ci offrent une grande répétabilité et une distribution convenable dans l'image. L'inconvénient est le temps de calcul nécessaire à leur extraction. Nous avons ensuite employé des points *SURF* en utilisant l'implémentation de Evans [1]. Comme pour le *SIFT* le détecteur-descripteur offre beaucoup de points d'intérêt et les appariements sont de très bonne qualité. Il est cependant assez coûteux, sur notre machine la détection et la description des points peut durer entre 100ms et 500ms pour des images de taille 640×480.

Nous avons testé plusieurs descripteurs binaires *FREAK*, *BRIEF*, *ORB* et *BRISK*, couplés au détecteur de coins rapide *FAST*. Les étapes de détections et description sont extrêmement rapides. Cependant nous avons observé à

1. `http://www.chrisevansdev.com/computer-vision-opensurf.html`

plusieurs reprises des appariements erronnés en quantité suffisante pour être tous filtrés efficacement. Le filtrage par *RANSAC* peut nécessiter un nombre important d'itérations, à tel point que dans notre première implémentation de *SLAM* en *Matlab* le temps gagné lors des détections-descriptions-appariements était perdu lors du filtrage. De plus, les points détectés sont souvent mal répartis ce qui est propice à la formation de configurations dégénérées et complique d'autant plus le filtrage des *outliers*.

Finalement, nous avons également essayé d'employer le détecteur *KLT* avec appariement réalisé par flot optique. Le détecteur est très rapide, les points sont détectés de manière assez uniforme dans l'image. Lorsque les caméras sont proches les appariements obtenus sont plus fiables que ceux fournis par les descripteurs binaires, mais cette fiabilité décroît très rapidement au cours du temps.

Nous nous sommes donc tenus à l'utilisation de points *SURF*.

4.2.2 Initialisation

Considérant que les points d'intérêt de la deuxième caméra clé fournissent des appariements supplémentaires au couple $\{\text{clé } 1, \text{clé } 3\}$ et donc permettent de trianguler de nouveaux points, il nous est apparu plus intéressant de la sélectionner par rapport à la troisième. En effet, l'ajout de points visibles par la troisième caméra clé améliore l'estimation de pose des caméras appariées avec celle-ci, c'est à dire des premières caméras suivant l'initialisation. Ainsi nous sélectionnons la troisième caméra clé comme dans [Mouragnon et al., 2006] mais la seconde est choisie en remontant l'historique des caméras de sorte qu'elle ait M appariements avec la troisième.

Bien qu'intuitivement la procédure d'estimation simultannée des poses des clés 2 et 3 apparaisse plus robuste, en pratique nous avons constaté peu de différence avec une estimation de chacune par la géométrie épipolaire. La lisibilité du code étant meilleure dans ce cas, nous avons privilégié cette façon de faire.

Ainsi nous détectons la première caméra clé, puis la troisième et estimons sa pose par la méthode des 5 points au travers d'un *RANSAC*. Nous recherchons ensuite la deuxième clé, et filtrons ses appariements avec la troisième. Sa pose est estimée directement par ajustment de faisceaux. Nous finalisons enfin l'initialisation par un ajustement de faisceaux sur l'ensemble des points et les poses des caméras clés 2 et 3.

Un schéma de notre procédure d'initialisation modifiée est visible à la figure 4.5.

4.2.3 Boucle de traitement

Chaque nouvelle caméra est initialisée selon les paramètres de pose de la précédente. Des points d'intérêt sont extraits, puis elle est appariée avec la dernière caméra clé. Les appariements sont filtrés par *RANSAC* et 5-points. Finalement, la pose de la caméra est mise à jour par un ajustement de faisceaux ne faisant varier que les paramètres de poses à partir des correspondances $3D$-$2D$.

L'usage des points *SURF*, par la qualité des appariements qu'ils fournissent, permet au *RANSAC* de s'exécuter en un faible nombre d'itérations. Appliquer la méthode des 5-points plutôt qu'une estimation de pose de Grunert, permet de filtrer tous les appariements, et pas seulement ceux pour lesquels un point a déjà été triangulé. Enfin, la pose étant raffinée par ajustement de faisceaux, une initialisation convenable est nécessaire. Puisque l'on fait du *SLAM*, la pose de caméra précédente remplit ce rôle convenablement.

Lorsque le nombre d'appariements *inliers* passe sous le seuil M ou lorsque le nombre de points reliés à un point de la carte passe sous le seuil M', la caméra précédente devient une caméra clé. Comme dans l'approche dont nous nous inspirons un ajustement de faiceaux local est alors appliqué et les appariements *inliers* sans $3D$ sont triangulés. Nous avons ajouté une étape de densification des liens entre la carte et la nouvelle caméra clé préalable à l'ajustement de faisceaux. Nous avons ajouté deux autres critères pour la détection de caméra clé, que nous présentons dès à présent, puis nous présentons l'étape de densification. Un schéma de la boucle principale de traitement est donné à la figure 4.6.

Détection de mauvaises estimations de pose

Les poses de caméra sont nécessairement similaires lorsque proches temporellement. Cela signifie que la pose estimée d'une caméra doit être proche de celle correspondant à la caméra clé avec laquelle elle est appariée.

Dans [Mikolajczyk et al., 2005] Mikolajczyk *et al* ont évalué la répétabilité de divers détecteurs de points d'intérêts, dont un détecteur Hessien. Le *SURF* étant un détecteur Hessien, nous nous basons sur les résultats obtenus pour ce détecteur. Ils observent que la répétabilité des points est égale à 50% pour

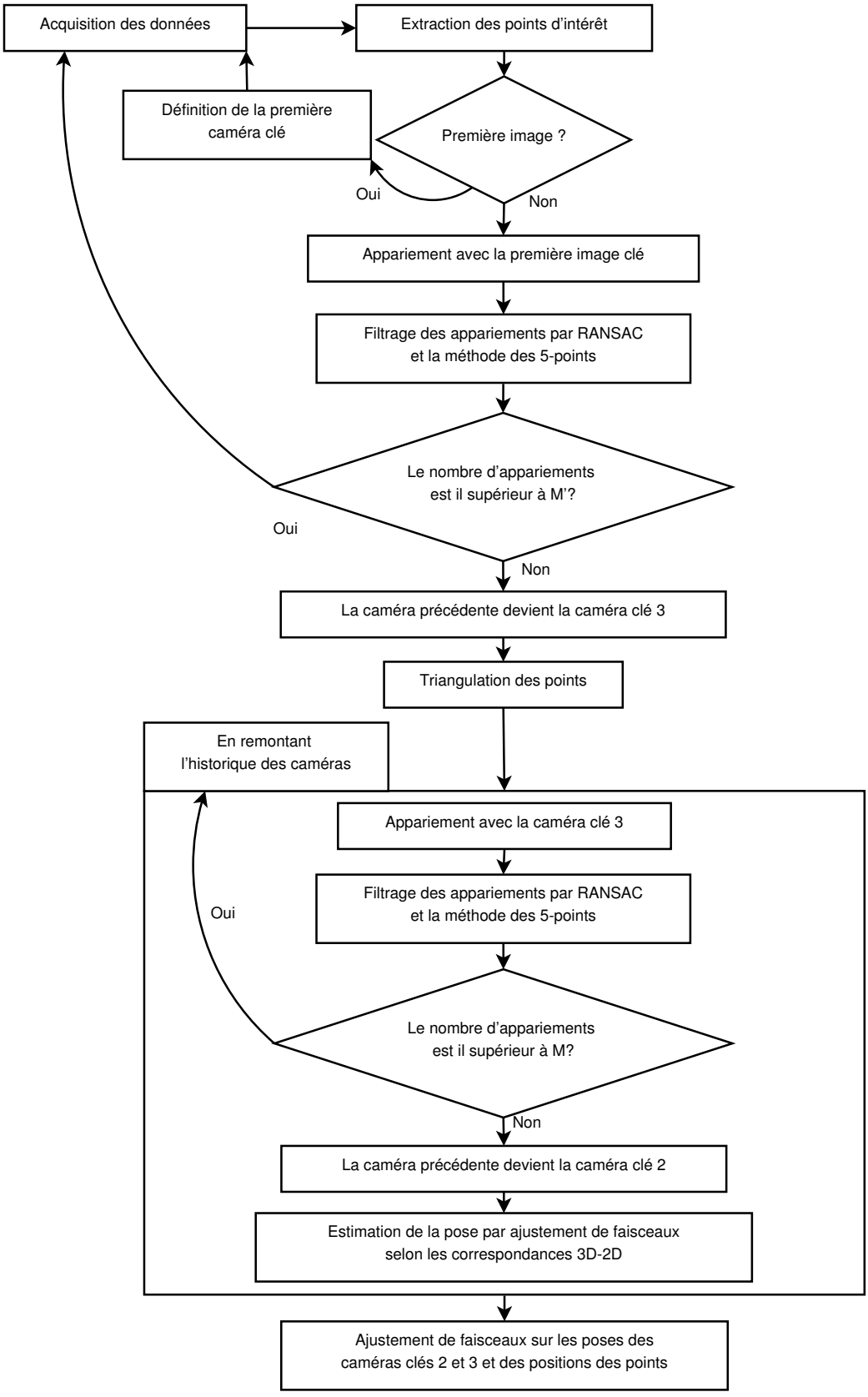


FIGURE 4.5 – Notre procédure d'initialisation modifiée

une variation de 40°. Cela fournit une mesure simple et assez pertinente pour détecter une mauvaise estimation de pose. Celle-ci n'est donc acceptée que si l'angle entre les axes de visée de la caméra et la caméra clé est inférieur à 45°. Si ce n'est pas le cas la caméra précédente devient une caméra clé.

Similairement l'évolution de la norme des translations doit être progressive. Sans connaissance préalable de la structure de l'environnement il est difficile de fixer théoriquement et *a priori* une mesure du déplacement maximal tolérable d'une caméra par rapport à la caméra clé précédente. Pour assurer cette continuité tout en permettant au système de s'adapter progressivement à l'environnement, nous imposons que la distance des centres optiques entre la caméra courante et la caméra clé associée ne doit pas dépasser 2 fois la distance entre cette caméra clé et la précédente. Sans quoi la caméra précédente devient une caméra clé.

Densification des liens

Les appariements et détection de points d'intérêt ne sont pas fiables à 100% : certains appariements sont erronés et des points ne sont pas détectés dans chaque image. Ainsi un point visible dans les images i et $i+2$ peut ne pas être détecté ou mal apparié dans l'image $i+1$. La conséquence en est que les observations dans les images i et $i+2$ d'un tel point ne sont pas reliées. Cette perte d'information conduit à une diminution des contraintes dans l'estimation de la pose de la caméra $i+2$, et donc potentiellement à une moins bonne estimation de ses paramètres ainsi que de la position du point dans la carte. De plus on risque d'obtenir une duplication du point reconstruit dans la carte.

Pour apporter un peu de robustesse à ces phénomènes, nous nous sommes inspirés de la solution de Klein et Murray, [G. Klein, 2007]. Cela est réalisé en recherchant dans l'image des points de la carte récemment créés mais non observés par la caméra. Cela ne demande que quelques étapes. La première consiste à parcourir les l derniers points ajoutés dans la carte non vus par la caméra, et de les reprojeter dans son image. Ces points sont ensuite appariés avec les observations n'ayant pas de point de la carte associé. Pour chaque appariement on crée un lien entre l'observation et le point reprojeté si la distance entre les deux est inférieure à un seuil d'erreur t. Le seuil d'erreur utilisé pour valider la création d'un lien entre un point d'intérêt et un point de la carte est de $t \sim 5.99 \cdot \sigma^2$, avec σ^2 la variance de la détection des points d'intérêt comme justifié dans [Hartley and Zisserman, 2004] et rappelé dans

la partie 2.9.

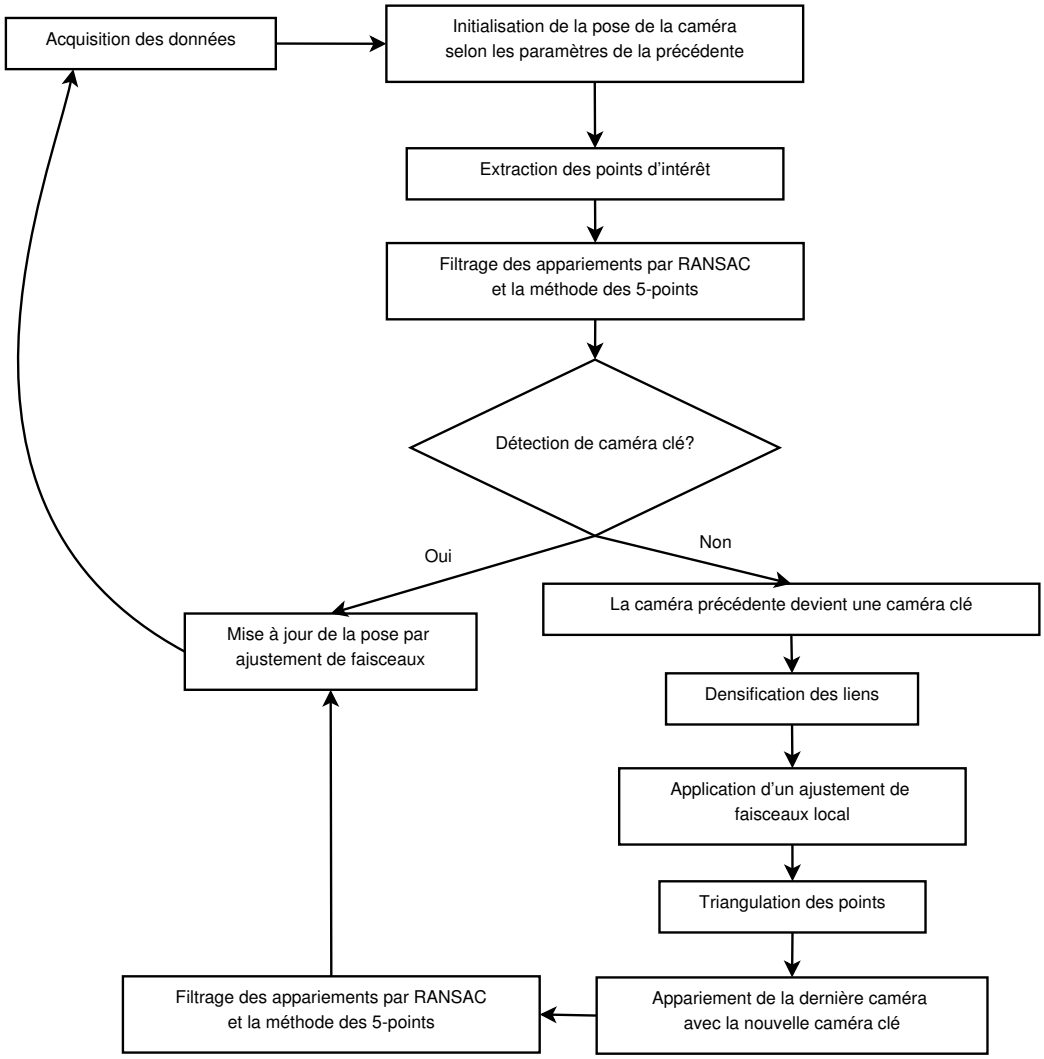


FIGURE 4.6 – Schéma de la boucle principale de traitement

Cette recherche de points de la carte peut être réalisée à l'issue de chaque estimation de pose, ou bien lorsqu'une caméra devient une caméra clé. Si l'on choisit la première option, cela permet de raffiner encore un peu l'estimation de pose en faisant un nouvel appel à l'ajustement de faisceaux.

Cependant la caméra a en général beaucoup d'appariements avec la précédente caméra clé, alors la précision gagnée est marginale. Si le nombre d'appariements est faible alors la caméra est destinée à devenir la prochaine caméra clé. Le coût de densification des liens n'étant pas nul, nous jugeons opportun de l'appliquer seulement lors du changement de statut de la caméra en caméra clé.

4.2.4 Résumé en pseudo-code

Notre *SLAM* modifié est résumé en pseudo-code dans l'algorithme 2.

Algorithme 2: Pseudo-code de notre *SLAM* modifié

Notations
- i indice de la caméra courante C^i
- j indice de la caméra clé courante K^j

tant que *nouvelle image* **faire**
- . détecter les points d'intérêt
- **si** $i == 1$ **alors**
  - . $C^1 \Rightarrow K^1 \overset{pose}{=} [Id|0]$
  - . passer à l'image suivante
- . $C^i \overset{pose}{=} C^{i-1}$
- . apparier avec la dernière caméra clé
- . $RANSAC$(5points) $\rightarrow$ [P : pose sans échelle, *inliers*]
- **si** $j < 3$ **alors**
  - **si** *détection de caméra clé* **alors**
    - . $C^{i-1} \Rightarrow K^3 \overset{pose}{=} P$
    - . trianguler les appariements entre K^1 et K^3
    - . remonter l'historique des poses :
      trouver C^l ayant $n \geq M'$ appariements *inliers* avec K^3
    - . $C^l \Rightarrow K^2$
    - . affiner la pose de K^2 par ajustement de faisceaux
    - . ajustement de faisceaux sur K^2, K^3 et les points de la carte
- **sinon**
  - . affinement de P^i par ajustement de faisceaux
  - **si** *détection de caméra clé* **alors**
    - . j = j+1
    - . $C^{i-1} \Rightarrow K^j$
    - . Densification des liens de K^j
    - . trianguler les appariements qui ne le sont pas déjà
    - . appliquer un ajustement de faisceaux local
    - . apparier C^i avec K^j
    - . $RANSAC$(5points) $\rightarrow$ [appariements *inliers*]
    - . affinement de C^i par ajustement de faisceaux

4.3 Applications

Nous montrons à présent deux applications de notre *SLAM* monoculaire.

4.3.1 Localisation et cartographie autour d'un bâtiment

Nous avons effectué une première application de localisation et cartographie autour d'un bâtiment pour un trajet dont la longueur est de l'ordre de 100m. Nous avons employé une webcam bas coût de marque *Logitech*. Les images capturées sont de taille 640×480, les paramètres intrinsèques de la caméra sont représentés par la matrice K suivante :

$$K = \begin{pmatrix} 532.8699 & 0 & 300.8525 \\ 0 & 532.8114 & 237.0369 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.1}$$

La figure 4.7 montre quatre images de la première façade observée et une vue de la reconstruction de cette façade avec les poses des caméras clés. Elles sont représentées par trois axes [vert, rouge, bleu] correspondant aux axes [X, Y, Z] de chacune.

La figure 4.8 montre des vues aériennes du bâtiment avec une approximation du chemin parcouru, et de la reconstruction. Les caméras clés sont toujours représentées par trois axes [vert, rouge, bleu].

On peut observer que la localisation est précise, certains points de la carte sont par contre fortement erronés, et la première façade n'est pas tout à fait plane.

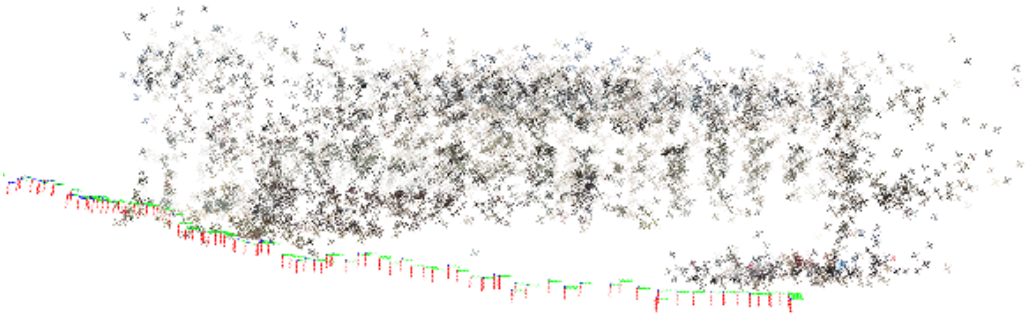
4.3.2 Réalité augmentée

Nous présentons une deuxième application de notre algorithme, cette fois en réalité augmentée.

La séquence sur laquelle nous faisons cette application est illustrée à la figure 4.9 par 8 images qui en sont tirées. Cette séquence contient 1254 images. Nous avons employé ici uniquement les données *RGB* d'une caméra *Xtion Pro Live* de résolution 640×480, dont les paramètres intrinsèques sont encodés dans la matrice K suivante :

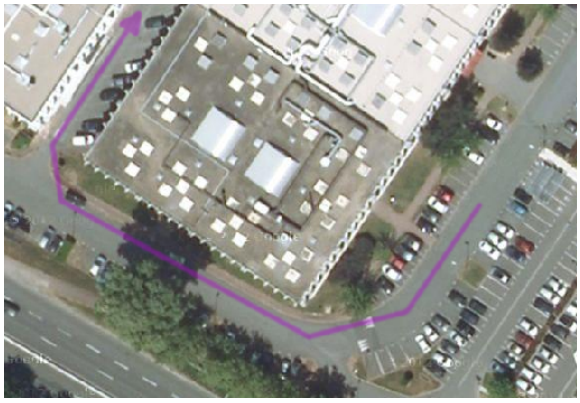
$$K = \begin{pmatrix} 546.04 & 0 & 316.66 \\ 0 & 546.05 & 234.71 \\ 0 & 0 & 1 \end{pmatrix}$$

(a) Quatre images de la première façade observée. Dans le flux vidéo elles apparaissent *"de la droite vers la gauche"*.

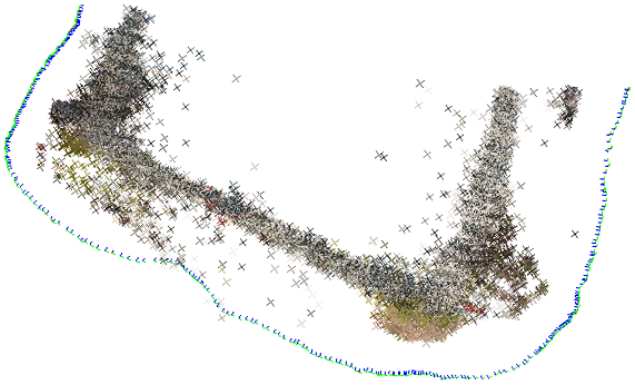


(b) Résultat du *SLAM* : nuage de points et poses de la caméra

FIGURE 4.7



(a) Vue aérienne du bâtiment et esquisse de la trajectoire de la caméra



(b) Vue aérienne de la reconstruction, légèrement de biais afin de voir les façades

FIGURE 4.8

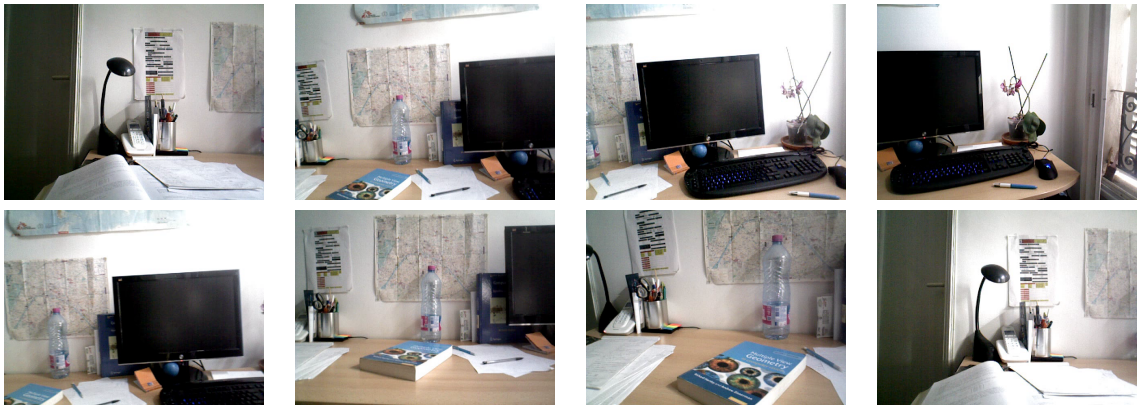
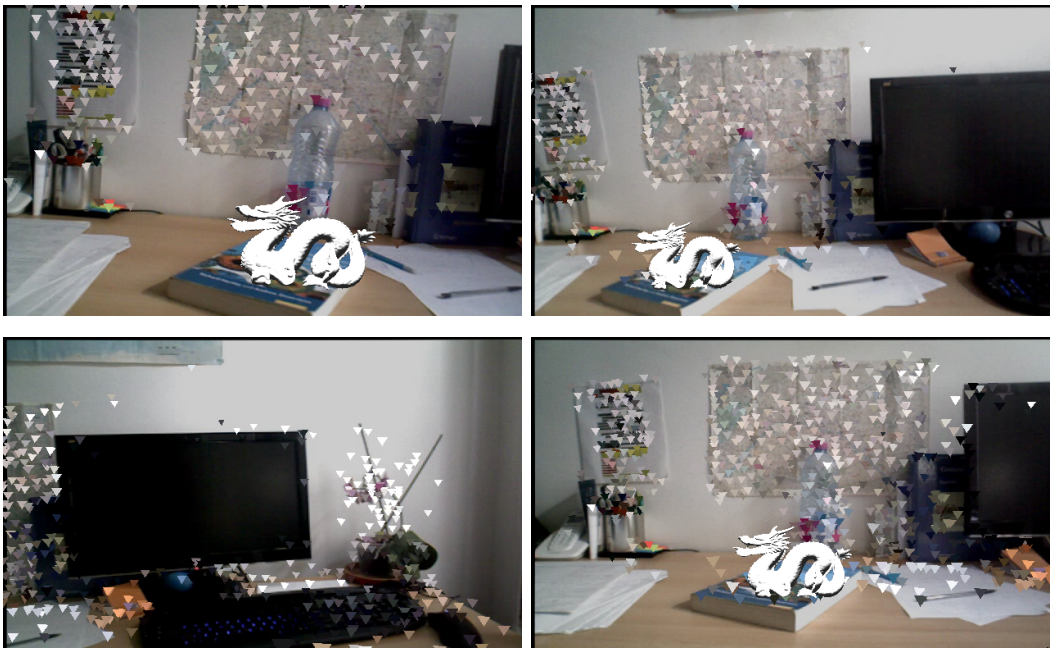


FIGURE 4.9 – Quelques vues de la séquence employée

Nous avons placé manuellement un objet virtuel, le dragon de Stanford, dans la carte créée par le *SLAM* de sorte qu'il apparaisse sur la couverture du livre. La figure 4.10 montre le résultats de la sur-impression de l'objet sur les images de la caméra. Nous affichons aussi les points triangulés de la carte sous la forme de triangles. La figure 4.11 montre la carte construite avec l'objet placé à l'intérieur.



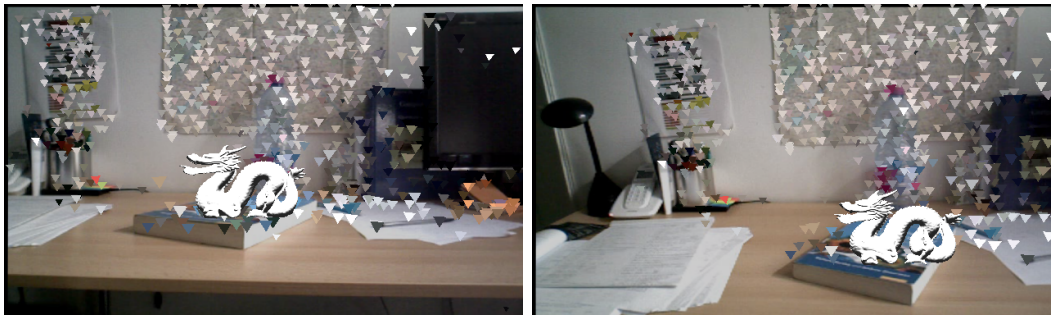


FIGURE 4.10 – Application de réalité augmentée, le dragon est virtuel. On affiche également les points de la carte sous forme de triangles

FIGURE 4.11 – Vue de la carte avec l'objet virtuel incrusté

On constate que les résultats sont satisfaisants, l'objet reste sur la couverture comme attendu.

4.4 Evaluations

Afin de mieux saisir quelles sont les forces et les faiblesses de notre *SLAM* monoculaire, nous procédons à présent à quelques évaluations.

Nous considérons une caméra sténopée *Xtion Pro Live* de résolution 640×480 calibrée dont les paramètres intrinsèques sont représentés par la matrice :

$$K = \begin{pmatrix} 546.04 & 0 & 316.66 \\ 0 & 546.05 & 234.71 \\ 0 & 0 & 1 \end{pmatrix}$$

Nous avons acquis six séquences de type ***boucle***, comprenant entre 700 et 1200 images. Dans la mesure de notre possible nous avons réalisé ces séquences en suivant des mouvements canoniques de la caméra. Ces mouvements canoniques correspondent soit à des translations pures selon un seul axe X, Y ou Z soit à des rotations pures autour d'un seul axe X, Y ou Z. On nomme ces séquences Seq_X, Seq_Y, Seq_Z et Seq_θ, Seq_ϕ, Seq_ψ. La figure 4.12 illustre les directions de ces mouvements.

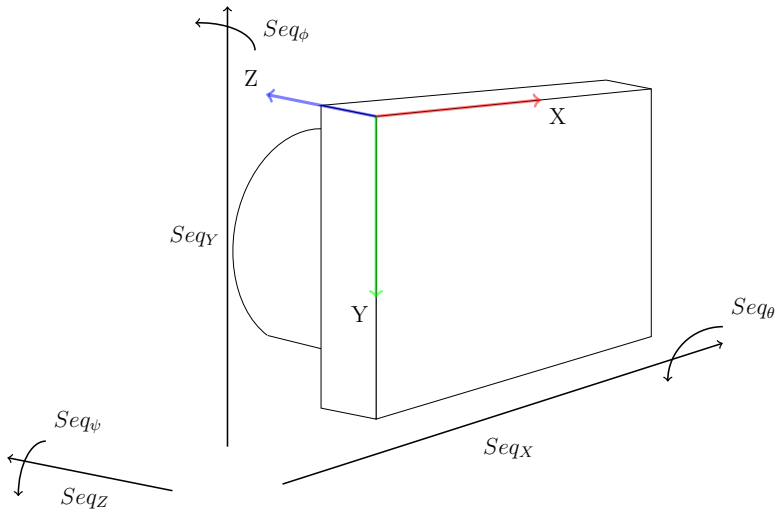


FIGURE 4.12 – Directions des mouvements de chaque séquence d'évaluation.

Pour chaque séquence nous avons effectué trois *allers-retours*. Pour Seq_X l'amplitude maximale d'un aller est de l'ordre de 100cm et la profondeur de la scène de l'ordre de 50cm, pour Seq_Y 60cm et 50cm, pour Seq_Z 100cm et la profondeur varie dans l'intervalle $[50, 150]$cm. Pour les séquences de rotation

l'amplitude du mouvement varie entre 0 et $\frac{\pi}{2}$ radians. Les profondeurs sont comprises dans l'intervalle $[50, 100]$cm pour Seq_θ, $[50, 150]$cm pour Seq_ϕ, et $[40, 50]$cm pour Seq_ψ.

Quelque soit la séquence, la pose de la dernière caméra est quasiment identique à celle de la première, modulo l'imprécision de l'expérimentateur. **Une légère imprécision a imprimé une faible translation au système dans les séquences Seq_θ et Seq_ψ.** Beaucoup de soin a été porté à la réalisation de Seq_ϕ. La comparaison de ces séquences se révélant porteuse d'information nous les avons conservées.

Nous avons appliqué sur les six séquences notre *SLAM* sans et avec recherche de points supplémentaires, partie 4.2.3. Dans la suite on nomme le premier *mono* et le second *densifié*. La figure 4.13 résume la forme sous laquelle nous présentons les résultats de chaque séquence.

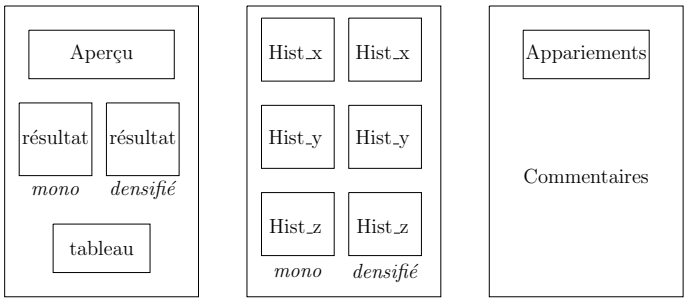
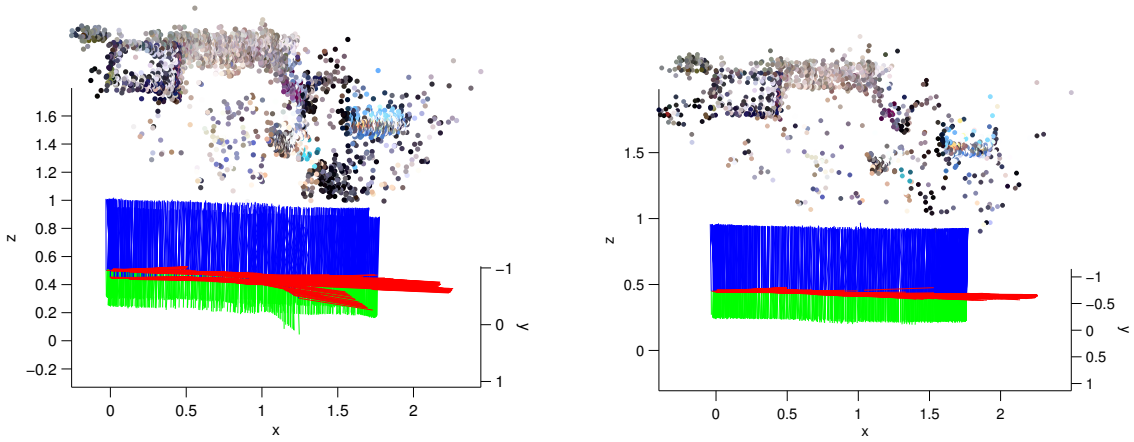


FIGURE 4.13 – Forme de la présentation des résultats d'évaluations pour une séquence

Les images "résultat" sont des vues de la reconstruction et des poses des caméras. On n'affiche que les points considérés *inliers*. Les critères de mesures d'*inliers* sont assez larges : on identifie les *outliers* selon un seuil de profondeur pour les séquences de translation, et selon un seuil de distance à la première caméra pour les séquences de rotation. Les quantités d'*inliers* données dans les figures "tableau" ne fournissent donc qu'un ordre de grandeur. Les figures "Hist" montrent l'évolution de la translation ou rotation, selon le type de séquence, pour chacun des axes de la caméra. Les mesures sont disponibles pour chaque caméra à partir de la troisième caméra clé. Les figures "Appariements" montrent en vert pour chaque caméra le nombre de points appariés avec la caméra clé associée. En rouge est affiché le nombre de point supplémentaires trouvés par l'étape de densification des liens. Celle-ci n'est appliquée que pour les caméras clés.

$\mathbf{Seq}_X$: Translation pure selon l'axe X de la caméra

FIGURE 4.14 – Tryptique de la scène



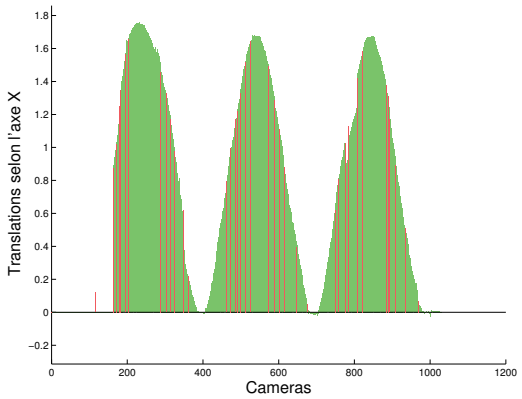
(a) Mono (b) densifié

FIGURE 4.15 – Reconstructions et caméras, vue de dessus. Les axes X, Y, Z sont en rouge, vert et bleu.

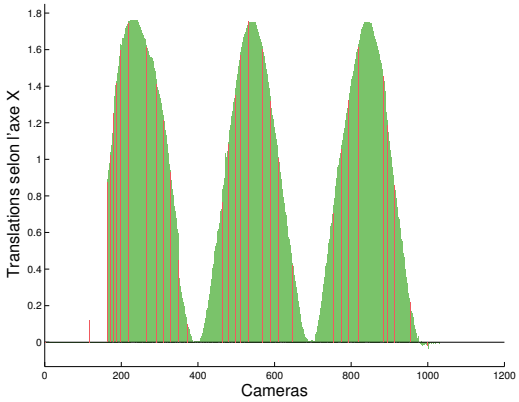
	Mono	densifié
Nombre de caméras :	1031	
Nombre de caméras clés :	39	32
Nombre de points :	3648	1502
Quantité d'inliers :	99%	98.5%

TABLE 4.1

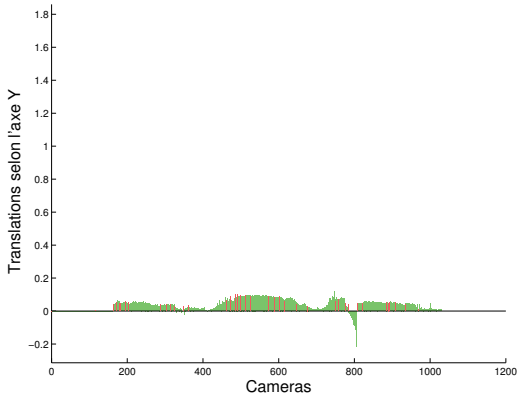
On mesure approximativement les points de la carte *outliers* comme ceux dont la profondeur est supérieure à la longueur du premier aller des six allers-retours effectués par la caméra.



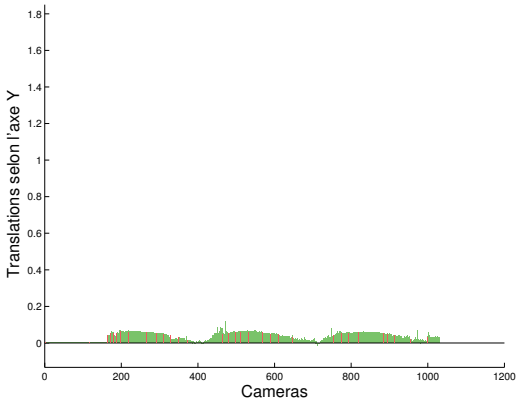
(a) Mono : translation le long de X



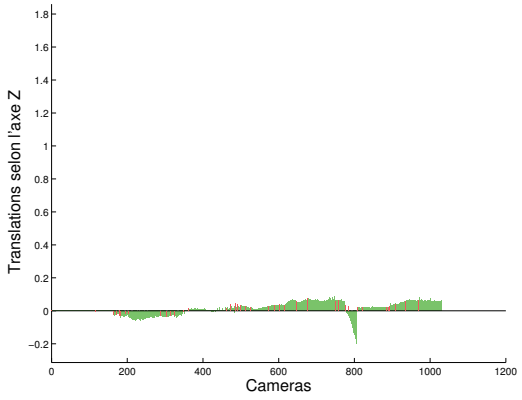
(b) densifié : translation le long de X



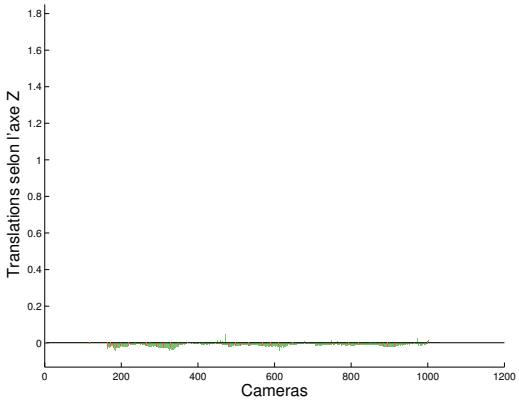
(c) Mono : translation le long de Y



(d) densifié : translation le long de Y



(e) Mono : translation le long de Z



(f) densifié : translation le long de Z

FIGURE 4.16 – Translations mesurées sur les trois axes de la caméra pour un mouvement de translation du système de long de l'axe X. En rouge sont signalées les données issues des caméras clés.

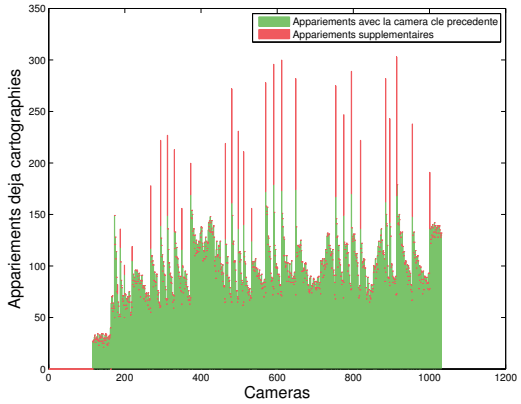


FIGURE 4.17 – En vert : quantité de points *inliers*. En rouge : quantité de points de la carte récupérés par densification des liens

On observe sur les reconstructions, figure 4.15, que la scène est assez bien cartographiée pour les deux approches. On y observe cependant un décrochage dans la trajectoire de l'approche *mono*, mais pas pour l'approche *densifié*.

Le tableau 4.1 et les figures de reconstruction montrent que la première carte contient environ deux fois plus de points que la seconde. Le tableau montre également une réduction non négligeable du nombre de caméras clés pour l'approche *densifié* par rapport à l'approche *mono*.

Les histogrammes, figure 4.16, montrent une estimation de trajectoire de la caméra en accord avec le mouvement imprimé. Pour l'approche *mono* on remarque que le décrochage vu sur la figure de reconstruction apparaît sur les trois histogrammes aux environs de la 800-ième caméra.

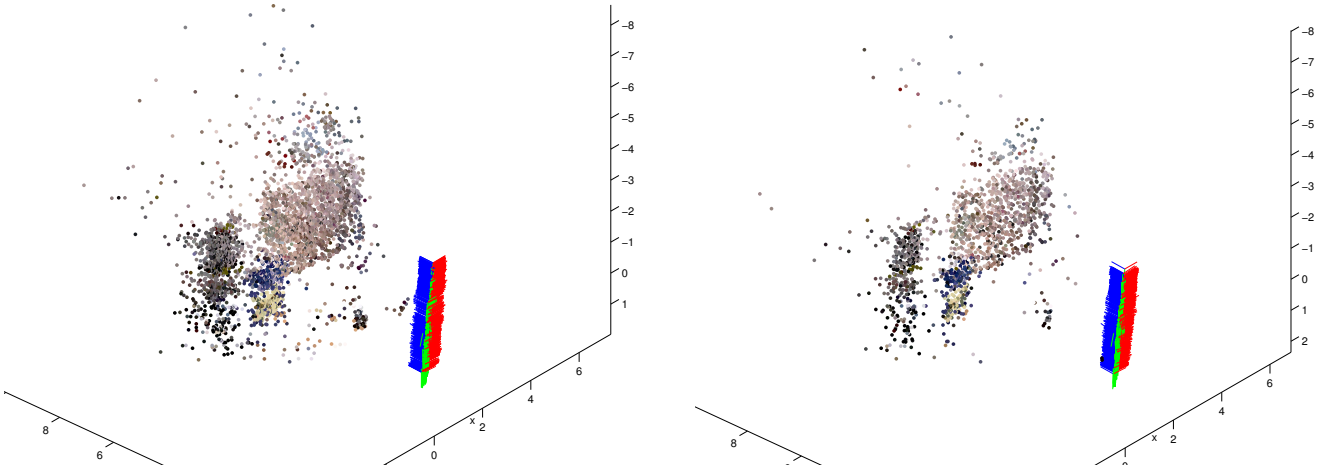
En regardant attentivement, on observe également que la hauteur des trois pics de translation en X décroît pour *mono* mais pas pour *densifié*.

La figure 4.17 montre que beaucoup d'appariements sont détectés lors de l'étape de densification des liens.

On en déduit donc que la quantité réduite de points de la carte pour l'approche *densifiée* est dûe aux appariements récupérés lors des densifications de liens. Les ajustements de faisceaux et estimations de poses s'en trouvent plus fortement contraints, ce qui permet à l'approche *densifiée* de ne pas subir le même décrochage que l'approche *mono*. Le nombre réduit de caméra clés en témoigne. La carte plus fournie de cette dernière est donc essentiellement peuplée de duplicata. La décroissance des pics pour *mono* est le signe d'une dérive du facteur d'échelle.

$\mathbf{Seq}_Y$: Translation pure selon l'axe Y de la caméra

FIGURE 4.18 – Tryptique de la scène



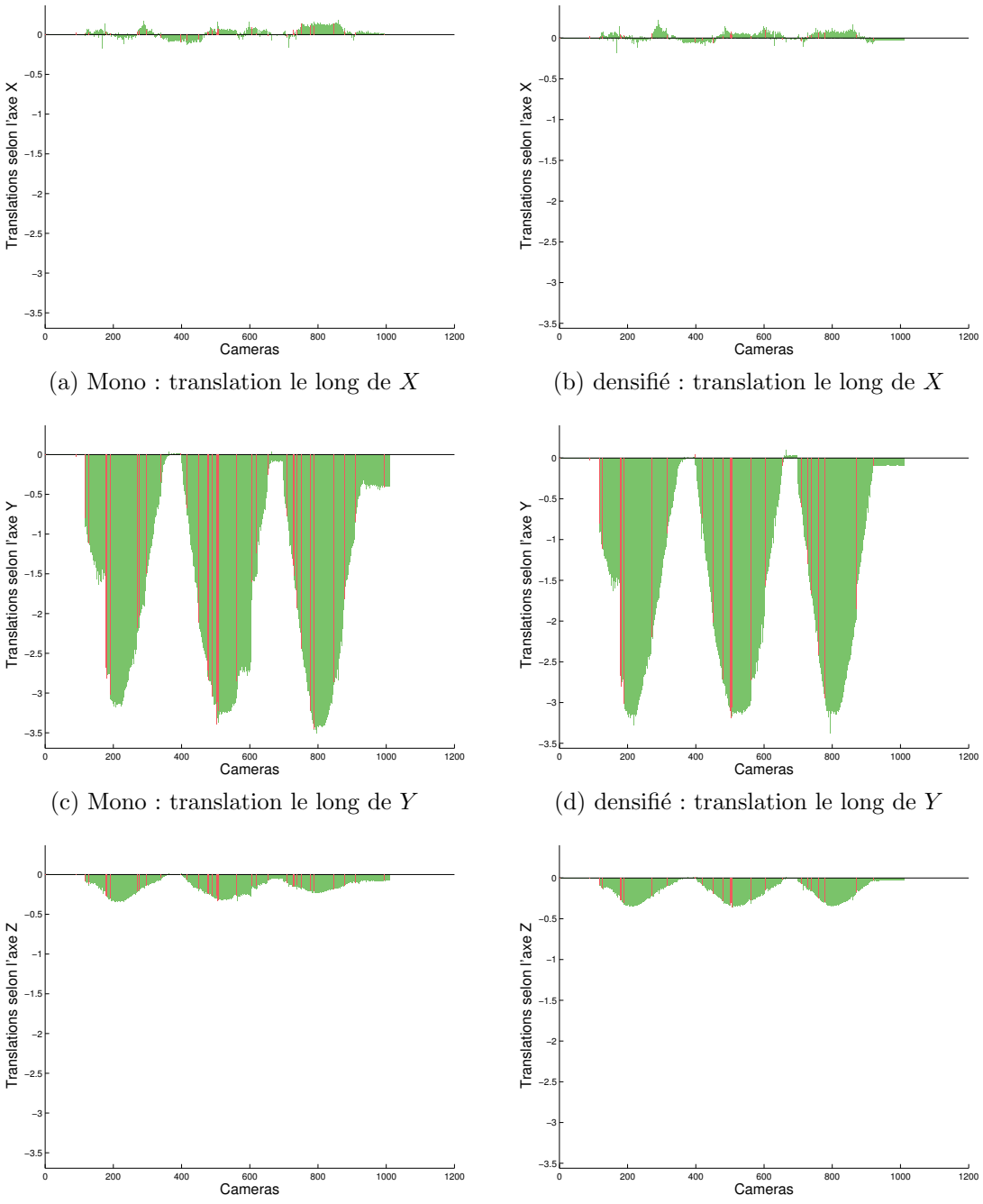
(a) Mono (b) densifié

FIGURE 4.19 – Reconstructions et caméras, vue de dessus et de biais. Les axes X, Y, Z sont en rouge, vert et bleu.

	Mono	densifié
Nombre de caméras :	1010	
Nombre de caméras clés :	37	27
Nombre de points :	4152	1791
Quantité d'inliers :	99.5%	98%

TABLE 4.2

On mesure approximativement les points de la carte *outliers* comme ceux dont la profondeur est supérieure à trois fois la longueur du premier aller de la caméra.



(a) Mono : translation le long de X

(b) densifié : translation le long de X

(c) Mono : translation le long de Y

(d) densifié : translation le long de Y

(e) Mono : translation le long de Z

(f) densifié : translation le long de Z

FIGURE 4.20 – Translations mesurées sur les trois axes de la caméra pour un mouvement de translation du système de long de l'axe Y. En rouge sont signalées les données issues des caméras clés.

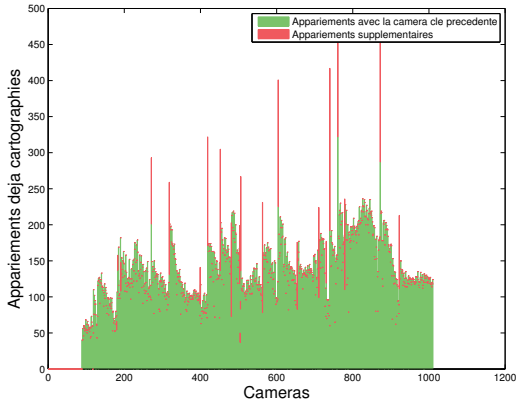


FIGURE 4.21 – En vert : quantité de points *inliers*. En rouge : quantité de points de la carte récupérés par densification des liens

Comme précédemment, la scène est assez bien cartographiée, figure 4.19. Les poses de caméras sont en accord avec le mouvement imprimé.

On observe sur le tableau 4.2, comme sur les reconstructions, la quantité réduite de caméras clés et points de la carte pour l'approche *densifié*.

Sur les histogrammes, figure 4.20, on observe que les pics de translation en Y croissent pour l'approche *mono*. On remarque également que pour cette approche la dernière position de retour est significativement différente de 0. Pour l'approche *densifié* on observe que la position de retour est également non nulle, mais plus faible, et qu'une caméra a décroché de la trajectoire, vers 800, mais sans emporter les suivantes avec elle.

La figure 4.21 montre un nombre important d'appariements récupérés.

On en déduit cette fois encore que la plus grande quantité de points dans la reconstruction *mono* est essentiellement dûe à des duplicata. La densification des liens a permis de mieux contraindre les ajustements de faisceaux et estimations de pose, ainsi la dérive du facteur d'échelle constatée pour *mono* n'est pas apparente pour *densifié*.

$\mathbf{Seq}_Z$: Translation pure selon l'axe Z de la caméra

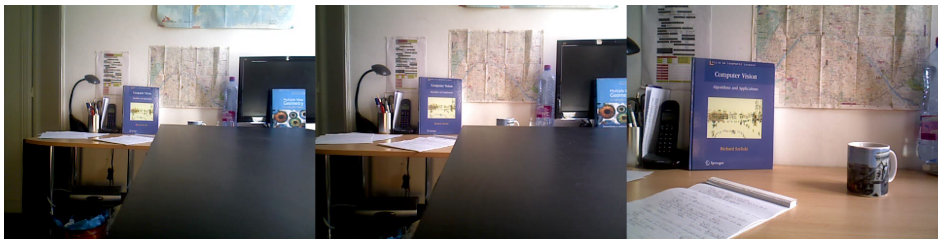
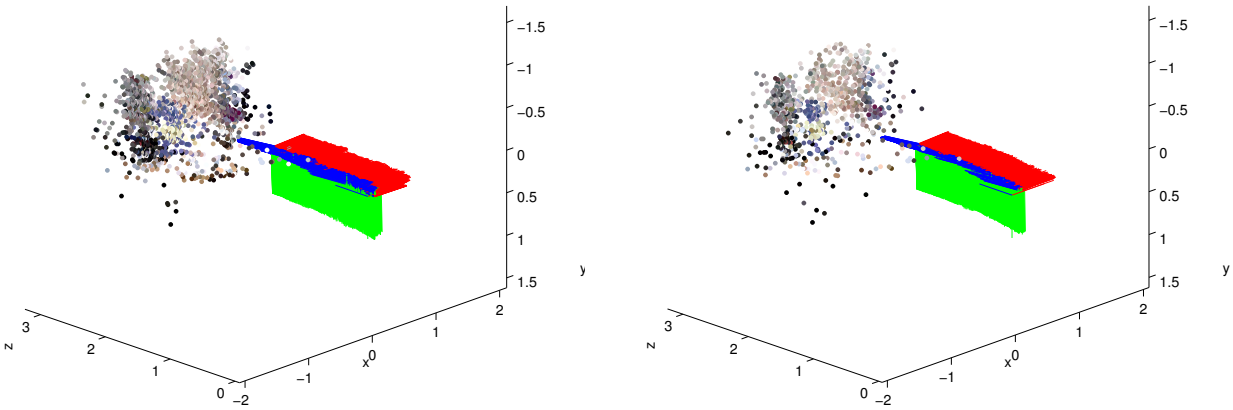


FIGURE 4.22 – Tryptique de la scène



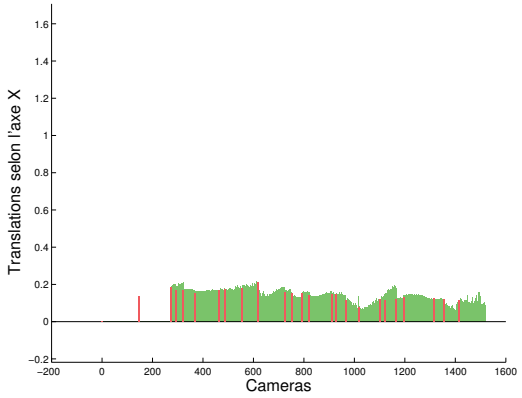
(a) Mono (b) densifié

FIGURE 4.23 – Reconstructions et caméras, vue de dessus. Les axes X, Y, Z sont en rouge, vert et bleu.

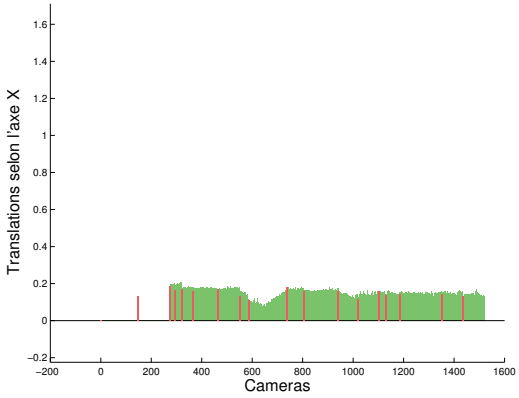
	Mono	densifié
Nombre de caméras :	1519	
Nombre de caméras clés :	27	18
Nombre de points :	2138	872
Quantité d'inliers :	96%	97.5%

TABLE 4.3

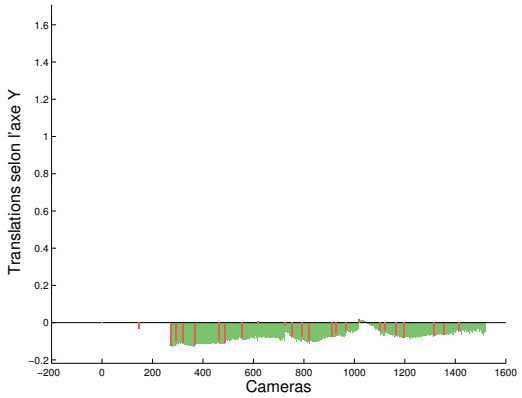
On mesure approximativement les points de la carte *outliers* comme ceux dont la profondeur est supérieure à deux fois la longueur du premier aller de la caméra.



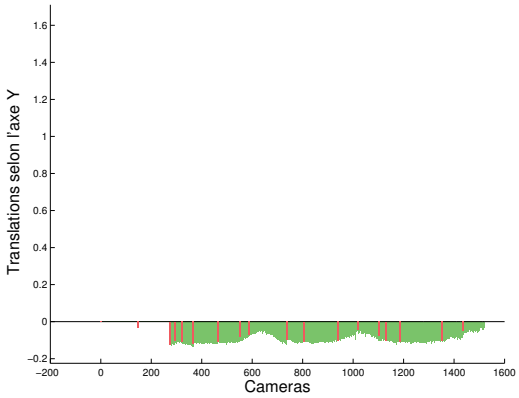
(a) Mono : translation le long de X



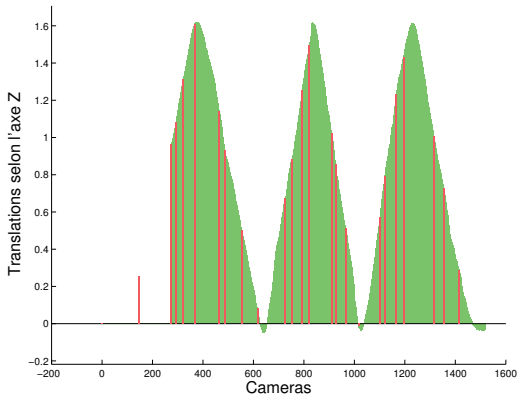
(b) densifié : translation le long de X



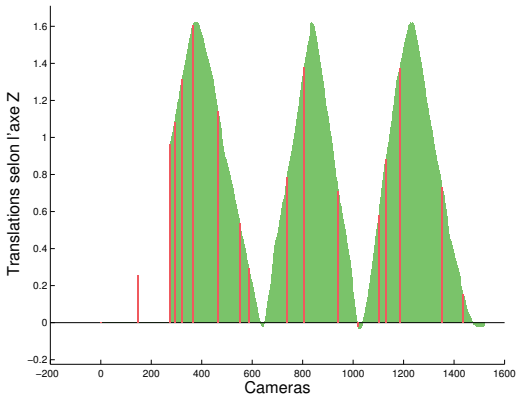
(c) Mono : translation le long de Y



(d) densifié : translation le long de Y



(e) Mono : translation le long de Z



(f) densifié : translation le long de Z

FIGURE 4.24 – Translations mesurées sur les trois axes de la caméra pour un mouvement de translation du système de long de l'axe Z. En rouge sont signalées les données issues des caméras clés.

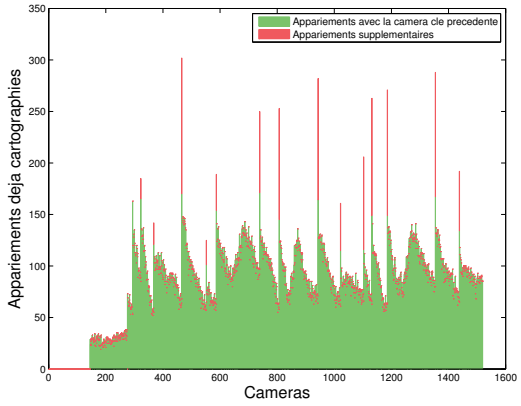


FIGURE 4.25 – En vert : quantité de points *inliers*. En rouge : quantité de points de la carte récupérés par densification des liens

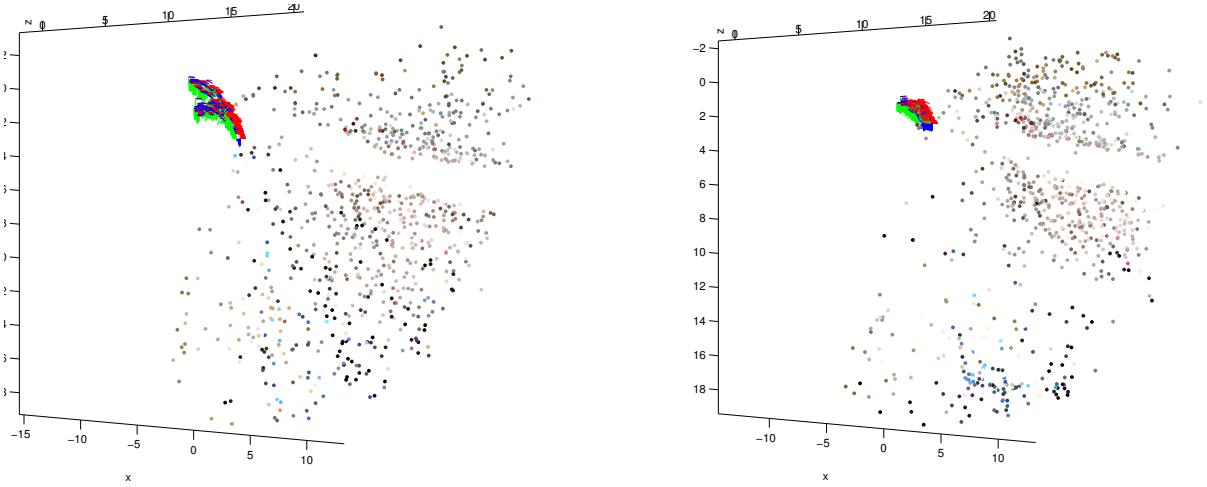
Les reconstructions, figure 4.23, montrent une cartographie assez satisfaisante et des poses de caméra en accord avec le mouvement imprimé. La carte semble être d'un peu moins bonne qualité que précédemment. Cela tient probablement à la configuration des caméras lors de la triangulation des points. En effet, les points au centre de l'image sont visibles plus longtemps que ceux en périphérie car ils présentent moins de mouvement. Puisqu'une caméra clé est ajoutée lorsque le nombre de points d'intérêt devient trop bas cela signifie que de nombreux points qui auraient pu être triangulés avec précision ont été perdus. On triangule donc ceux qui s'y prêtent le moins.

Pour les deux approches aucun histogramme, figure 4.24, ne présente ici de dérive du facteur d'échelle perceptible. Ceci est probablement dû à ce qu'une partie de la scène observée est visible par chaque caméra du début à la fin de la séquence.

Toutefois, on constate toujours un nombre significativement plus faible de points dans la carte, donc de points dupliqués, comme de caméras clés pour l'approche *densifié*.

$\mathbf{Seq}_\theta$: Rotation autour de l'axe X et légère translation

FIGURE 4.26 – Tryptique de la scène



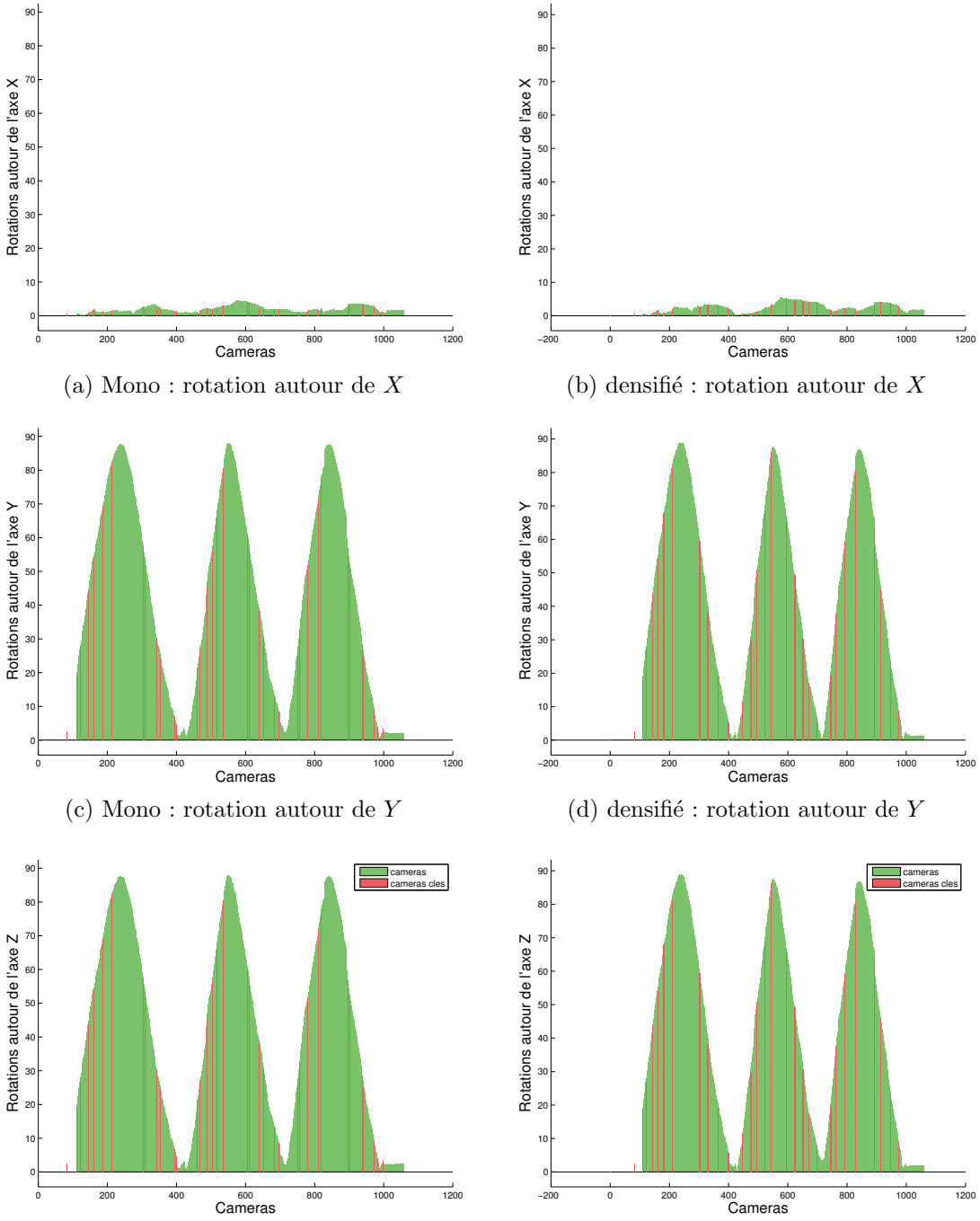
(a) Mono (b) densifié

FIGURE 4.27 – Reconstructions et caméras, vue de côté. Les axes X, Y, Z sont en rouge, vert et bleu.

	Mono	densifié
Nombre de caméras :	1058	
Nombre de caméras clés :	39	31
Nombre de points :	4503	1864
Quantité d'inliers :	20.5%	47.5%

TABLE 4.4

Pour l'estimation des *outliers* nous avons déterminé visuellement une valeur de distance à la première caméra pour laquelle la carte paraît la moins erronée.



(a) Mono : rotation autour de X

(b) densifié : rotation autour de X

(c) Mono : rotation autour de Y

(d) densifié : rotation autour de Y

(e) Mono : rotation autour de Z

(f) densifié : rotation autour de Z

FIGURE 4.28 – Rotations mesurées autour des trois axes de la caméra pour une rotation du système autour de l'axe X. En rouge sont signalées les données issues des caméras clés.

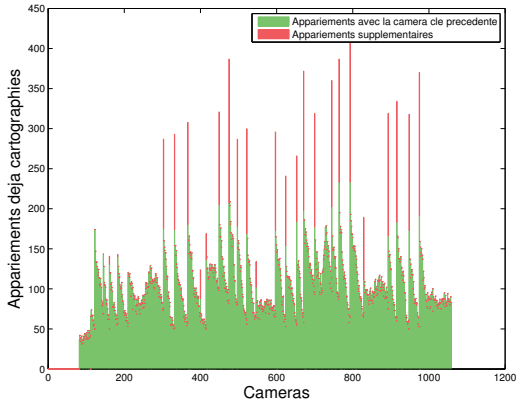


FIGURE 4.29 – En vert : quantité de points *inliers*. En rouge : quantité de points de la carte récupérés par densification des liens

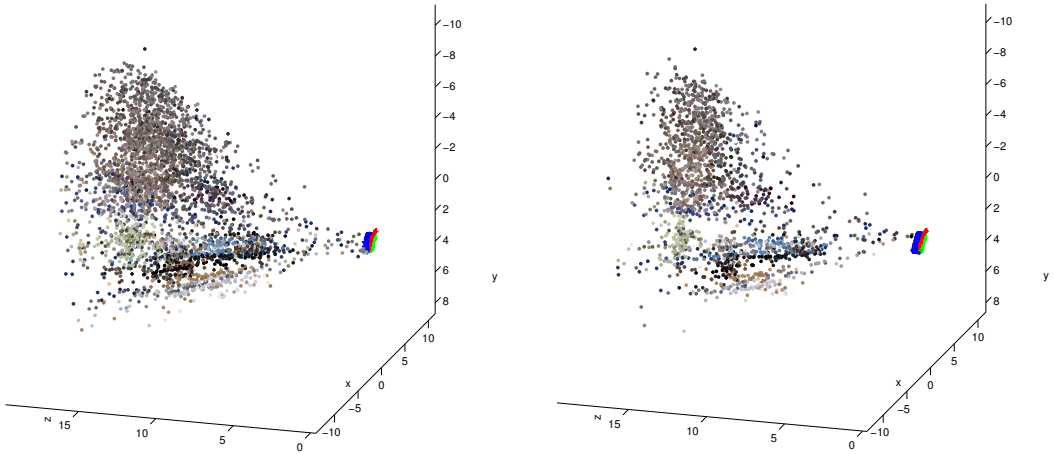
D'après les reconstructions, figure 4.27 on observe qu'une faible translation a été imprimée au système. Si la profondeur des points triangulés est très mal estimée, la localité du mouvement de la caméra rend son estimation relativement peu sensible à ce défaut.

Ainsi les histogrammmes, figure 4.28, montrent des rotations en accord avec le mouvement imprimé. Il ne fait cependant peu de doute que la carte ne serait pas utilisable si la caméra venait à effectuer de plus amples translations.

Les appariements récupérés par la méthode de densification sont nombreux, en témoigne la figure 4.29. Pour ce mouvement nettement plus critique que les précédents, le tableau 4.4 montre une différence non négligeable d'*inliers* entre les approches *mono* et *densifié*.

$\mathbf{Seq}_{\psi}$: Rotation autour de l'axe Z et légère translation

FIGURE 4.30 – Tryptique de la scène



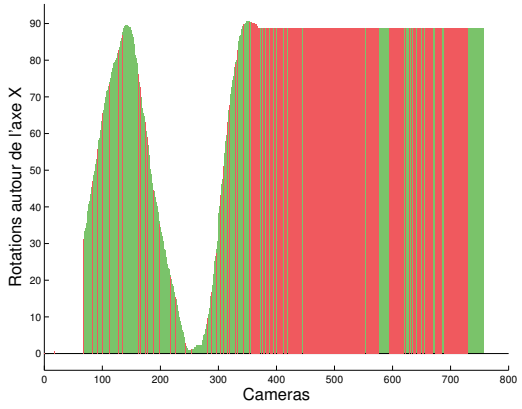
(a) Mono (b) densifié

FIGURE 4.31 – Reconstructions et caméras, vue de côté. Les axes X, Y, Z sont en rouge, vert et bleu.

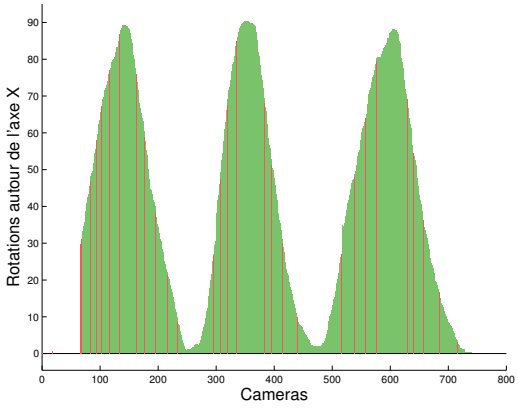
	Mono	densifié
Nombre de caméras :	756	
Nombre de caméras clés :	329	31
Nombre de points :	12355	1999
Quantité d'inliers :	29%	94%

TABLE 4.5

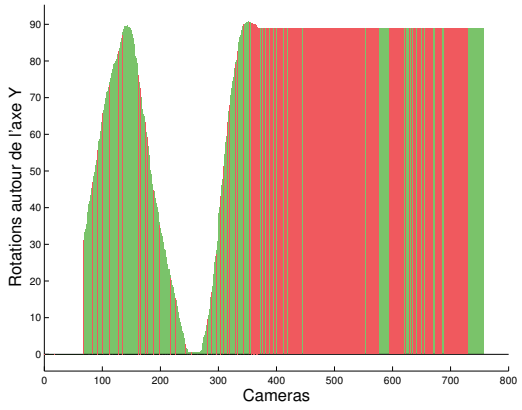
Pour l'estimation des *outliers* nous avons déterminé visuellement une valeur de distance à la première caméra pour laquelle la carte paraît la moins erronée.



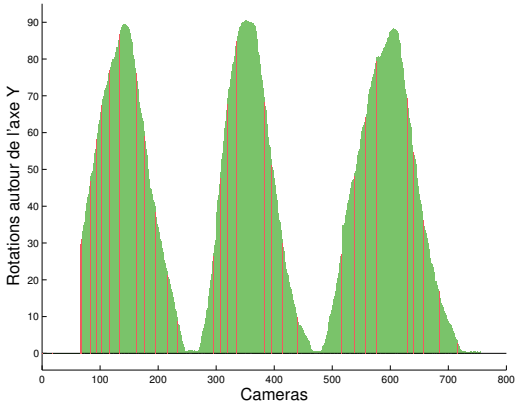
(a) Mono : rotation autour de X



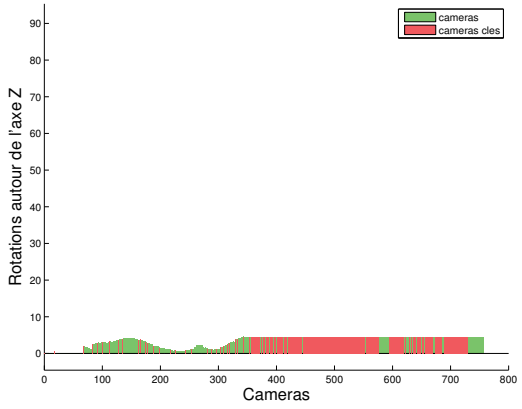
(b) densifié : rotation autour de X



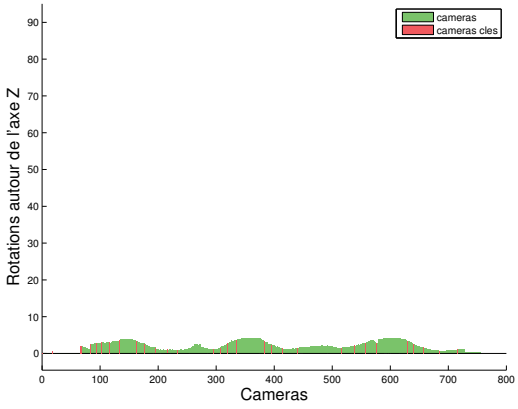
(c) Mono : rotation autour de Y



(d) densifié : rotation autour de Y



(e) Mono : rotation autour de Z



(f) densifié : rotation autour de Z

FIGURE 4.32 – Rotations mesurées autour des trois axes de la caméra pour une rotation du système autour de l'axe Z. En rouge sont signalées les données issues des caméras clés.

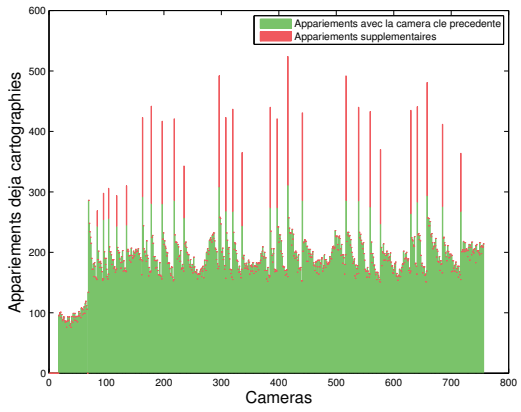


FIGURE 4.33 – En vert : quantité de points *inliers*. En rouge : quantité de points de la carte récupérés par densification des liens

On observe sur les histogrammes, figure 4.32, que l'estimation du mouvement de la caméra est erronéee pour l'approche *mono* mais juste pour l'approche *densifiée*. A partir de la 350-ème caméra environ le système entre dans une phase de création frénétique de caméras clés, il en résulte que chacune est initialisée selon la pose de la dernière caméra ayant observé suffisament de points d'intérêt. Cette création effrénée de caméras clés induit autant d'étape de triangulation, d'où le nombre élevé de points, comme indiqué sur le tableau 4.5.

On voit sur la figure 4.33 que l'étape de densification permet de retisser un nombre important de liens, en particulier aux environs de la 350-ème caméra, là où l'approche *mono* se perd.

La triangulation est de mauvaise qualité, l'estimation de profondeur est difficile, comme en témoigne les reconstructions en figure 4.31. La localité du mouvement diminue la sensibilité du processus d'estimation de pose à ces triangulations de mauvaise qualité. La très faible quantité d'*inliers* pour l'approche *mono* est dûe à la perte du système.

$\mathbf{Seq}_\phi$: Rotation pure autour de l'axe Y

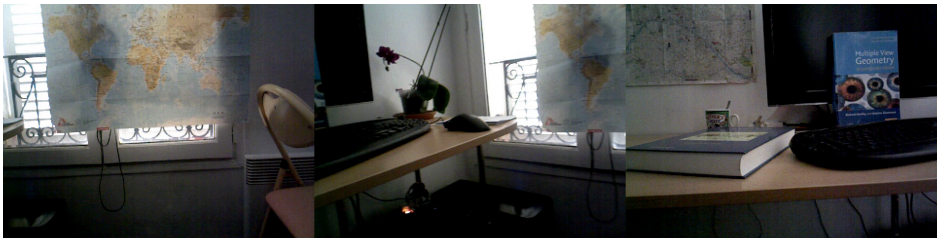
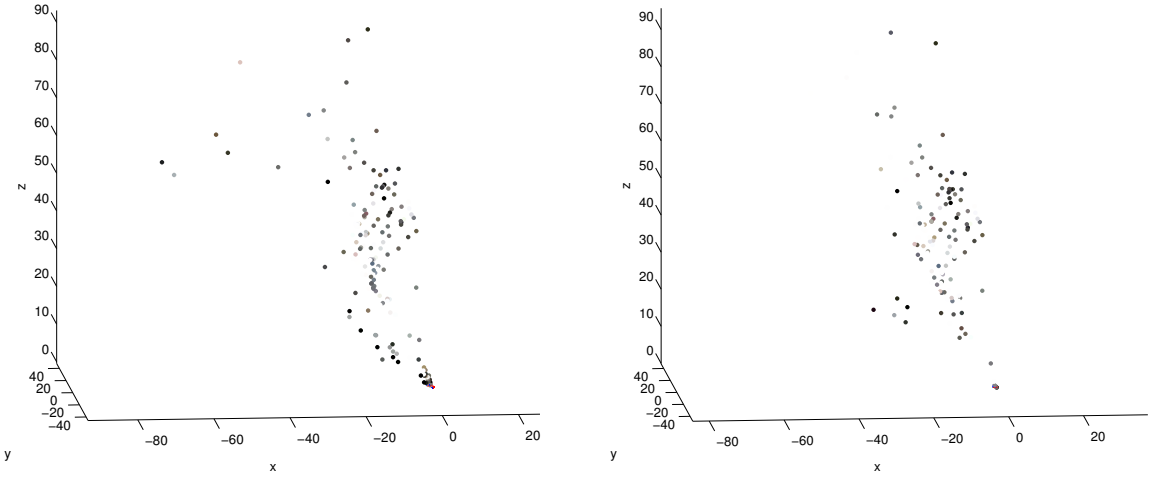


FIGURE 4.34 – Tryptique de la scène



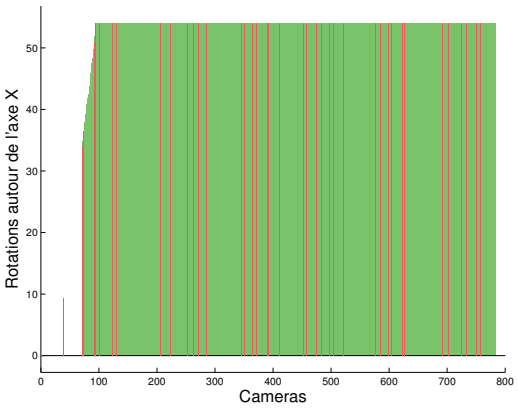
(a) Mono (b) densifié

FIGURE 4.35 – Reconstructions et caméras, vue de dessus. Les axes X, Y, Z sont en rouge, vert et bleu.

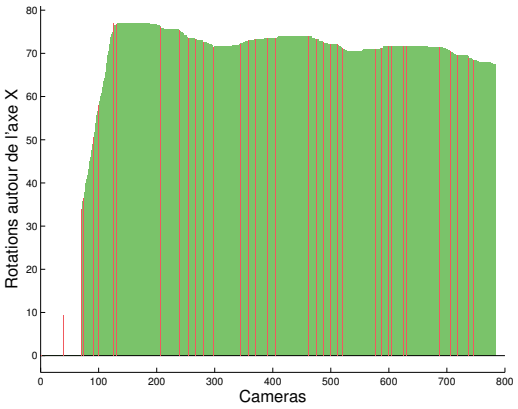
	Mono	densifié
Nombre de caméras :	783	
Nombre de caméras clés :	41	36
Nombre de points :	3550	2927
Quantité d'inliers :	4.5%	4.5%

TABLE 4.6

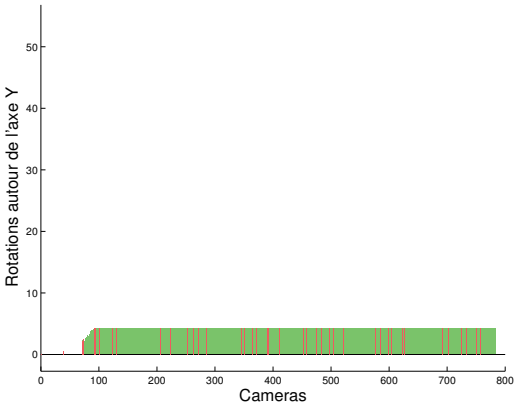
Pour l'estimation des *outliers* nous avons déterminé visuellement une valeur de distance à la première caméra pour laquelle la carte paraît la moins erronée.



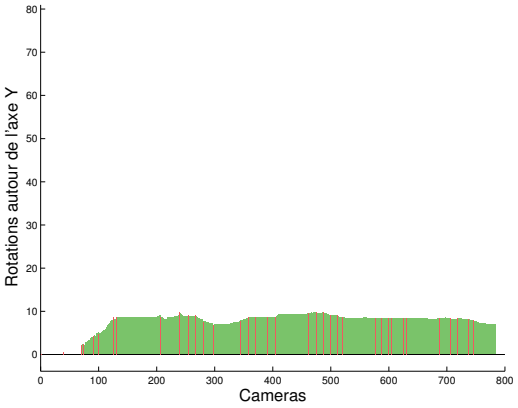
(a) Mono : rotation autour de X



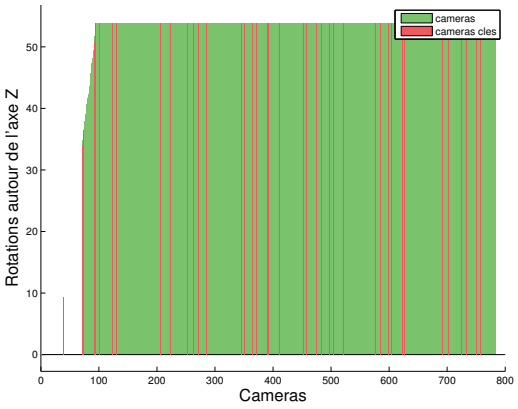
(b) densifié : rotation autour de X



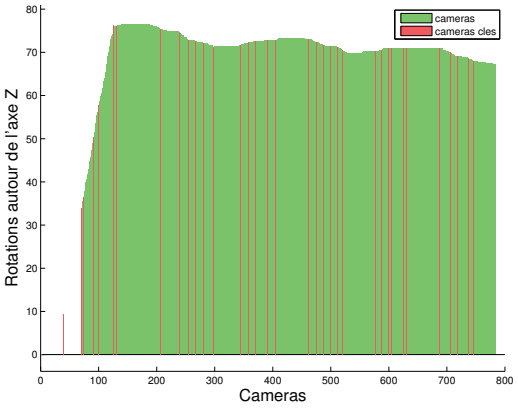
(c) Mono : rotation autour de Y



(d) densifié : rotation autour de Y



(e) Mono : rotation autour de Z



(f) densifié : rotation autour de Z

FIGURE 4.36 – Rotations mesurées autour des trois axes de la caméra pour une rotation du système autour de l'axe Y. En rouge sont signalées les données issues des caméras clés.

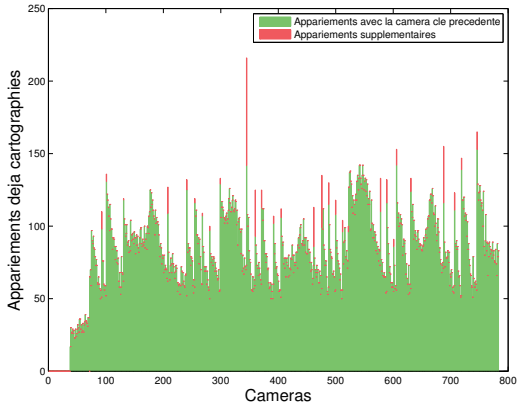


FIGURE 4.37 – En vert : quantité de points *inliers*. En rouge : quantité de points de la carte récupérés par densification des liens

Pour les deux approches les cartes, figure 4.35, sont totalement erronnées. Les orientations estimées, mieux appréciables sur les histogrammes de la figure 4.36, sont également en désaccord avec le mouvement imprimé.

On constate sur cette séquence que l'absence de translation rend toute cartographie impossible. L'étape de densification n'apporte alors aucune correction, la quantité de liens recréés est d'ailleurs très faible.

4.5 Les limites du *SLAM* monoculaire

Notre adaptation de *SLAM* monoculaire possède des limitations, les suivantes étant sans doute les plus pénalisantes.

Initialisation et suivi des points

L'initialisation du système est une étape délicate et est liée à la définition des seuils M et M' n'est pas évidente.

Une scène peu texturée peut donner peu de points d'intérêts et conduire le système à finaliser l'initialisation avant que le mouvement ne soit suffisament important pour conditionner la géométrie épipolaire raisonnablement.

Une scène très texturée peut au contraire donner beaucoup de points d'intérêt et conduire le système à finaliser l'initialisation après que la caméra ait effectué plusieurs allers-retours.

Dans [Mulloni et al., 2013] Mulloni *et al* proposent un schéma d'initialisation basé sur une initialisation à une profondeur constante des points. Cette profondeur est progressivement raffinée simultanément à l'estimation des nouvelles poses de la caméra au travers d'un ajustement de faisceaux. Bien que le schéma ne fonctionne pas pour tout type de mouvement les autres mènent une étude avec des utilisateurs et montrent que les mouvements naturels lors de la phase d'initialisation sont aussi ceux qui permettent la réussite de leur algorithme. Ils proposent également une méthode de vérification de la réussite de l'algorithme basée sur la géométrie épipolaire.

Mouvement de rotation et triangulation

Les évaluations de notre *SLAM* monoculaire sur des mouvements canoniques en translation et rotations ont mis en évidence la difficulté à prendre en compte les rotations. Il est en effet impossible de construire une carte par triangulation de points lorsque le centre optique de la caméra ne bouge pas ou très peu.

Dans le cas ou le centre optique ne bouge pas et où les mesures du point X, imagé en m^1 et m^2 dans les images I^1 et I^2 des caméras C^1 et C^2 de centres optiques O^1 et O^2, sont parfaites les rayons $\overrightarrow{r^1} = \overrightarrow{O^1 m^1}$ et $\overrightarrow{r^2} = \overrightarrow{O^2 m^2}$ sont confondus, ainsi calculer leur intersection n'a pas de sens. A plus forte raison, si les mesures de X sont bruitées et que le centre optique ne bouge pas, le calcul d'intersection des rayons ne peut que donner des résultats aberrants.

Enfin, c'est le cas problématique général, si le déplacement du centre optique n'est pas nul mais que les déplacements qu'il induit sur les images de X sont faibles devant le bruit des mesures, le processus de triangulation est presque certainement voué à échouer.

Dans [Pirchheim et al., 2013] Pircchheim *et al* proposent une solution partielle à ce problème en développant un système de *SLAM* basé sur une détection du type de mouvement de la caméra. Lorsque ce dernier présente suffisamment de translation, le système adopte une approche classique, *6Dof*, inspirée de [G. Klein, 2007]. Par contre lorsque pour une suite de caméra clés le mouvement est principalement rotatif, le système adopte une approche *3Dof* où seules les rotations sont estimées et plutôt que de trianguler les points observés forme une image panoramique. Le basculement entre les deux modes est réalisé lors de l'ajout de caméra clé. Dans ce cas pour chaque paire constituée de la nouvelle caméra clé et une caméra parmi un certain nombre des dernières caméras clés, une approximation de la parallaxe est réalisée pour la zone de carte mutuellement observée. S'il existe une de ces caméras telle que l'angle de parallaxe est trop faible et que l'angle entre son axe de visée et celui de la nouvelle caméra clé est suffisamment important, alors la nouvelle caméra clé est considérée comme purement rotative et le système bascule dans le mode $3Dof$.

Le système ne peut re-basculer vers le mode classique que si des zones précédemment cartographiées sont à nouveau observées.

Dérive

A chaque traitement de caméra des erreurs de mesures se produisent, les modélisations de capteurs et d'erreurs ne sont pas absolument précises, et la précision des calculs est limitée. Il en résulte que de petites erreurs d'estimation se produisent à chaque instant. Le processus de *SLAM* étant par nature incrémental, ces erreurs s'accumulent et conduisent le système à dériver.

Les évaluations, partie 4.4, ont montré l'apparition de ce phénomène, en particulier les séquences Seq_X et Seq_Y. La première application, partie 4.3.1, illustre également la dérive du facteur d'échelle, notamment concernant la distance de la caméra au bâtiment.

4.6 Utilisation d'une connaissance *a priori* pour réduire le phénomène de dérive

Puisque des erreurs sont destinées à apparaître à chaque instant et à se propager, une solution est de supposer connues des parties de la scène. Ce scénario paraît hautement plausible dans de grandes villes. En effet, dans [Agarwal et al., 2009] Agarwal *et al* ont démontré qu'il est possible d'obtenir des reconstructions automatisées de bâtiments historiques extrêment précises en utilisant seulement des photos touristiques tirées des bases de photos en ligne telles que *Flickr*. Le papier illustre notamment des reconstructions du Colisée à Rome ou encore de certaines parties de la ville de Venise.

Notons que ce scénario se rapproche des travaux de Irschara *et al* publiés dans [Irschara et al., 2009]. La différence avec l'approche d'Irschara est que celle-ci présuppose une modélisation préalable complète de la scène. Cette contrainte est très forte et peu d'environnements s'y prêtent. En revanche, nous proposons de tirer partie d'une connaissance éparse de la scène et de l'utiliser lorsqu'elle est disponible pour corriger la dérive de notre système. Une manière de réaliser ceci est d'ajouter une étape de reconnaissance effectuée péridodiquement, suivie en cas de succès d'une relocalisation du système et une propagation de la connaissance.

Nous pensons que la propagation de la connaissance est importante si l'utilisateur est amené à ré-explorer des zones ou encore si des augmentations ont précedemment été déposées dans la modélisation de l'environnement. En effet, dans ce dernier cas des augmentations réalistes nécessitent d'être fixées dans la carte modélisée par le *SLAM* donc relativement à des paramètres de poses passées. Dans le cas où des augmentations ainsi déposées sont visibles lorsque la pose du système est ajustée la connaissance nécessite d'être propagée au travers de l'historique et de la carte afin de pouvoir corriger les positions des augmentations. Sans quoi celles-ci sembleraient "sauter" brusquement.

Puisque notre *SLAM* est basé sur l'approche *SLAM-SFM* l'ajustement de faisceaux est un composant primordial du système. Nous proposons de propager la connaissance au travers de cette procédure. Comme dans l'approche de Mouragnon [Mouragnon et al., 2006] nous suggérons d'utiliser des paramètres fixes lors de l'ajustement de faisceaux, c'est à dire des paramètres contribuant à l'évaluation de la fonction de coût mais n'étant pas optimisés. On note :

- X les paramètres de l'ensemble des points de la carte à optimiser
- X_a les points *ancrés*, l'ensemble des points entrant en ligne de compte dans l'évaluation de la fonction de coût mais non optimisés
- C les paramètres des caméras à optimiser
- C_o les caméras observatrices, l'ensemble des caméras utilisées lors de l'évaluation de la fonction de coût mais non optimisées
- z_j^i la mesure du point X_j dans l'image de la caméra C^i

Dans ce cas la fonction de coût minisée est :

$$f(C,X) = \sum_{C^i \in \{C_o;C\}} \sum_{X_j \in \{X_a;X\}} d(h(C^i, X_j), z_j^i)^2 \tag{4.2}$$

Nous avons devisé plusieurs stratégies de propagation de connaissance. Ces stratégies sont basées sur la correction de la trajectoire de la caméra, mais les positions des points de la carte sont également corrigées au travers de l'ajustement de faisceaux.

Les travaux d'Irschara ayant démontré la faisabilité d'une reconnaissance de la scène modélisée par un nuage de points en temps réel nous supposons cette partie comme fonctionnelle et avons mis en place des protocoles simplifiés pour l'évaluation des stratégies de propagation. Ces protocoles nécessitent trois applications de notre *SLAM* pour une même séquence vidéo, celle montrée dans la partie 4.3.1 :

- Une première application de notre *SLAM* modifié est appliquée. La modification consiste en l'application d'un ajustement de faisceaux global à chaque étape de réduction d'erreur. Les résultats de ce processus sont de bonne qualité et nous les considérons comme la vérité terrain. Nous appelons la trajectoire issue de cette application la trajectoire de référence.
- Les résultats de notre *SLAM* non modifié sur cette séquence étant également de bonne qualité, c'est à dire prenant du temps pour montrer une dérive importante, nous dégradons volontairement ses performances de sorte à accentuer le phénomène de dérive. Cela est fait en modifiant l'étape de minimisation des erreurs de reprojection : seule la dernière caméra clé est variable et seules les deux caméras clés précédentes sont fixes. Une application de ce *SLAM* dégradé est donc réalisée.
- Finalement une troisième application de notre *SLAM*, dégradé, avec mise en oeuvre d'une stratégie de propagation est réalisée.

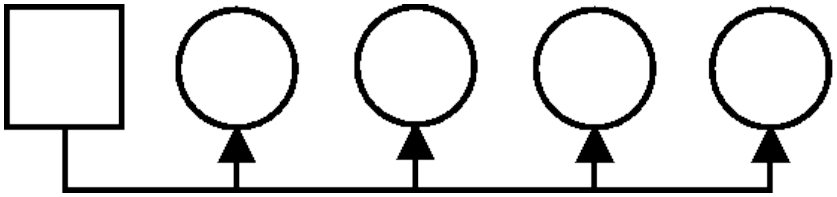


FIGURE 4.38 – Représentation de la première stratégie de propagation. La dernière caméra est fixe, elle est représentée par un carré, toutes les précédentes sont variables, elles sont représentées par des cercles.

FIGURE 4.39

La vérité terrain nous permet de simuler aisément la reconnaissance de scène puisque les paramètres de détection de caméras clé sont les même lors de la construction de la référence et lors des applications de notre *SLAM* dégradé.

La première stratégie a d'abord été présentée dans [Boucher et al., 2012]. Lorsqu'une simulation de reconnaissance de scène est effectuée, nous corrigeons la pose de la caméra clé courante selon ses paramètres dans la trajectoire de référence. Puis un ajustement de faisceaux est appliqué prenant en compte la caméra corrigée comme paramètre fixe et comme paramètres variables les poses des caméras clés précédentes ainsi que les positions des points de la carte. Une schématisation de la stratégie est visible à la figure 4.38.

Les résultats de la propagation de connaissance sont sauvegardés pour analyser l'efficacité de la méthode, mais ils ne sont pas pris en compte dans la suite de l'algorithme, laissant l'estimation diverger au cours du temps.

Lors des expérimentations nous avons utilisé des détecteurs de points d'intérêts *SIFT* avec des seuils assez élevés d'*inliers* minimum inter caméras clés. Le résultat de notre *SLAM* muni de ce détecteur avec ces seuils appliqué à la scène 4.39 est visible à la figure 4.40.

Nous prenons pour critères d'évaluation de l'efficacité des stratégies les réductions d'erreurs en orientation et en translation des caméras. Les erreurs

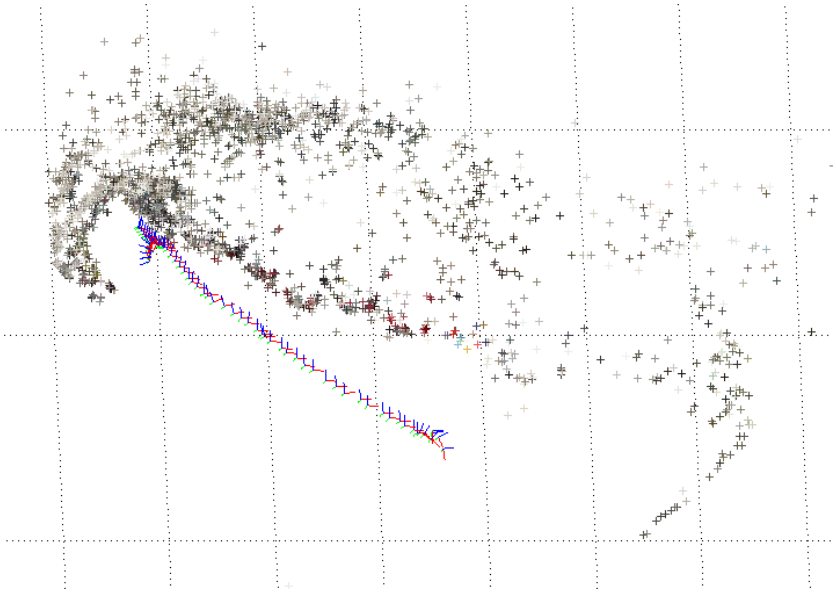


FIGURE 4.40

sont mesurées comme :

– la norme L_2 des différences pour la translation :

$$\varepsilon_T = ||T_{ref} - T_c||_2 \tag{4.3}$$

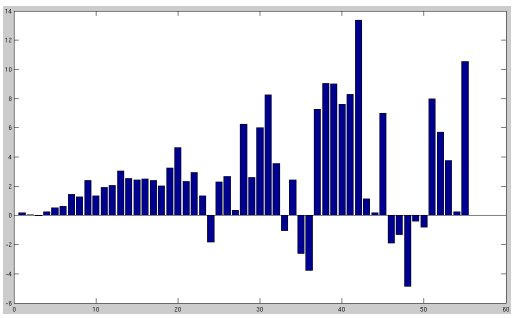
– l'angle entre les axes de visée normalisés pour l'orientation :

$$\varepsilon_O = \arccos(Z_{ref}^{\top} \cdot Z_c) \tag{4.4}$$

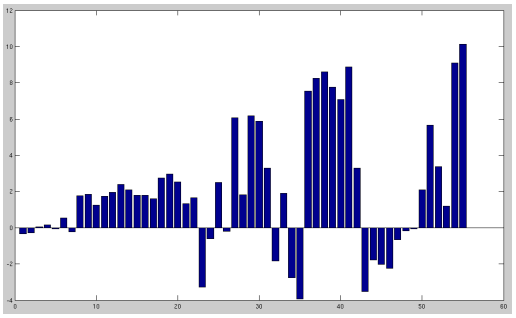
On note ε_T^d l'erreur en translation de la pose d'une caméra pour la trajectoire dégradée, ε_T^p l'erreur en translation de la pose d'une caméra pour la trajectoire avec application de la propagation de connaissance, et de la même manière ε_O^d et ε_O^p pour les erreurs en orientation. Les réductions d'erreurs en translation et en orientation sont respectivement mesurées comme :

$$r_T = \varepsilon_T^d - \varepsilon_T^c \quad \text{et} \quad r_O = \varepsilon_O^d - \varepsilon_O^c \tag{4.5}$$

Pour cette première stratégie nous avons mesuré les réductions d'erreur en translation pour les caméras clés $i-1$ et $i-2$ lorsque la reconnaissance de scène est simulée à la caméra clé i. La figure 4.41 illustre les réductions d'erreurs en translation mesurées sur ces caméras au long de la trajectoire.



(a) Réduction d'erreur apportée sur l'estimation des caméras clés lorsque la suivante est relocalisée.



(b) Réduction d'erreur apportée sur l'estimation des caméras clés lorsque le système est relocalisé deux caméras clés plus tard.

FIGURE 4.41

De ces figures il apparait que la réduction d'erreur est d'une efficacité similaire que les caméras soient corrigées en position $i-1$ ou $i-2$ tant que la distance à la caméra clé relocalisée est inférieure à $l = 20$ caméras clés. Cette distance est bien sûr dépendante de plusieurs facteurs, ceux-ci incluant notamment le nombre de caméras utilisées dans l'ajustement de faisceaux lors de l'ajout d'une caméra clé, le nombre moyen d'appariements entre chaque caméras clé successives ainsi que le nombre d'*outliers* dans ces appariements passés entre les mailles du filtrage. Dans la figure 4.42 nous avons tracé les erreurs pour chaque caméra lorsque corrigée en position $i-1$ et $i-2$.

Cette figure montre plus clairement que les erreurs ont été globalement réduites. On constate également qu'à la distance de $l = 21$ caméras clés la correction de la caméra clé $i-2$ n'est pas bonne. Cela signifie que la dérive était trop importante et que l'ajustement de faisceaux est tombé dans un minimum local. Cela signifie donc que la portée de l'introduction de connaissance est limitée dans l'historique des poses.

Les résultats obtenus nous donnent une distance, en terme de caméra clé, d'efficacité du premier schéma. Dans [Boucher et al., 2013] nous avons proposé deux autres schémas de propagation au travers de l'historique des poses. Le but de ces schémas est de réduire le coût calculatoire de la propation de connaissance. Celle-ci est toujours transférée d'une pose à l'autre via les positions des points de la carte. Les points sont donc toujours corrigés.

La deuxième stratégie de propagation de connaissance consiste en une application itérative d'ajustement de faisceaux en s'assurant que le poids des

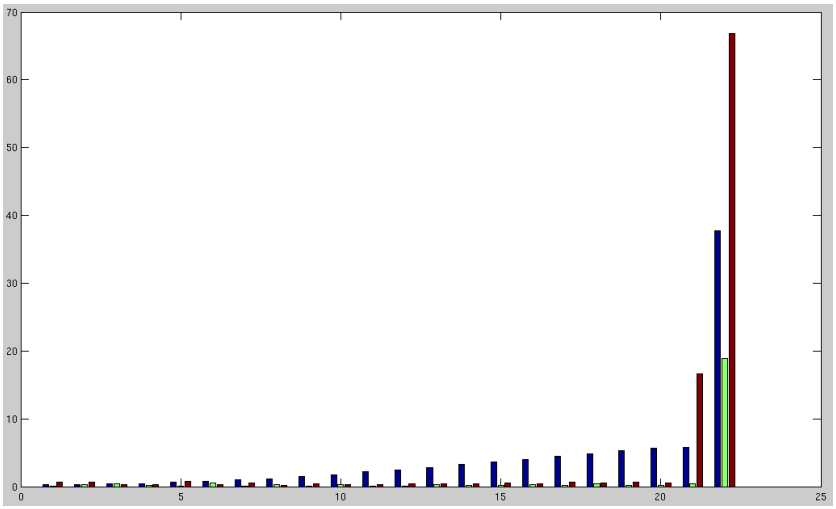


FIGURE 4.42 – Les erreurs d'estimation des positions de caméra. En bleu : l'erreur initiale, en vert : l'erreur lorsque le système est relocalisé une caméra clé plus tard, en rouge : lorsqu'il est relocalisé deux caméras clés plus tard.

caméras fixes supposées corrigées est plus important que celles à raffiner. En pratique une phase d'initialisation corrige la caméra clé $i-1$ par ajustement de faisceaux en fixant la caméra clé i. Ensuite, sur une longueur de caméra donnée, la trajectoire des poses est remontée en appliquant un ajustement de faisceaux avec deux caméras fixes et une caméra variable. Une représentation de cette stratégie est visible à la figure 4.43.

La troisième stratégie est similaire à la seconde excepté que l'initialisation est réalisée en corrigeant une pose sur deux par ajustement de faisceaux, en utilisant à chaque fois une caméra fixe. Ensuite toutes les caméras non corrigées le sont en utilisant deux caméras fixes. Une représentation du schéma est visible à la figure 4.43. Cette stratégie présente l'intérêt d'être parallélisable une fois l'initialisation effectuée.

On note n le nombre total de caméras clé à corriger. On note M le nombre de points observés par ces caméras. Le design de notre algorithme induit que le nombre de points observés entre les caméras i et $i-1$ est presque constant, de même qu'entre les caméras i et $i-2$. On fait donc l'approximation qu'ils le sont et on note ces constantes M_1 et M_2. On note $cba(n, M)$ le coût calculatoire d'un ajustement de faisceaux comprenant n caméras et M points, et donc $O(cba(n, M)) = O(n^2 \cdot M)$ sa complexité algorithmique. Les complexités

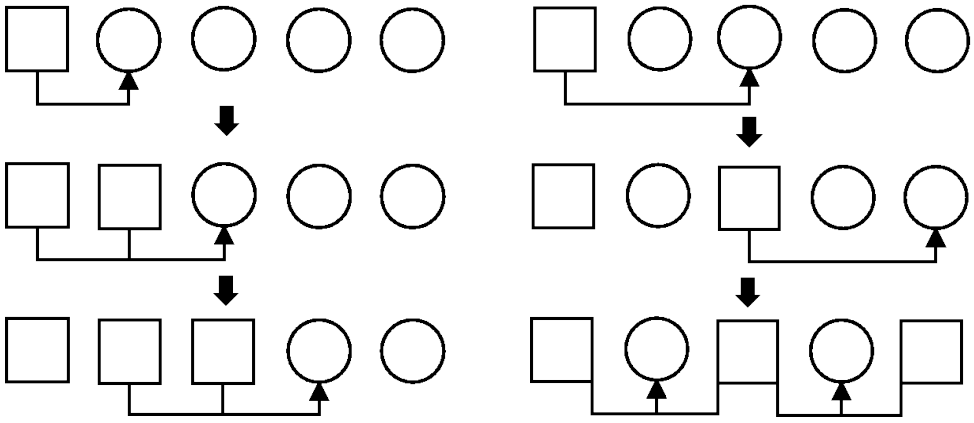


FIGURE 4.43 – Représentation des deuxième et troisième stratégies de propagation. Les caméras fixes pour l'ajustement de faisceaux sont représentées par des carrés, les caméras variables par des cercles.

algorithmiques des trois stratégies sont :

- Pour la première le coût de l'ajustement de faisceaux :

$$O(n^2 \cdot M) \tag{4.6}$$

- Pour la seconde le coût de l'initialisation puis des ajustement itératifs :

$$\begin{array}{rl} & O(cba(2, M_1) + (n-2) \cdot cba(3, M_1 + M_2)) \\ = & \\ & O(n \cdot (M_1 + M_2)) \end{array} \tag{4.7}$$

- Pour la troisième le coût des initialisations successives puis des ajustements de faisceaux finaux :

$$\begin{array}{rl} & O(\frac{n}{2} \cdot (2, M_2) + \frac{n}{2} \cdot cba(3, 2 \cdot M_1)) \\ = & \\ & O(n \cdot (M_1 + M_2)) \end{array} \tag{4.8}$$

- Il peut être intéressant de remarquer de la différence des complexités entre les stratégies deux et trois est de $O(n.M_2)$ en faveur de la stratégie trois.

Les trois stratégies ont été évaluées sur la même scène que précédemment en utilisant cette fois des détecteurs de points d'intérêt *SURF* avec des seuils d'*inliers* plus faibles, permettant d'obtenir une trajectoire plus longue (de

l'ordre de 250 caméras clés dans les expérimentations). La figure 4.44 montre les réductions d'erreurs mesurées pour des périodes de relocalisations de 4, 7, 11 et 17 caméras clés. Dorénavant le résultat des stratégies de propagation est conservé dans la suite de l'exploration.

Des résultats il ressort que la troisième, et moins coûteuse, stratégie est la moins performante. Elle présente des cas d'erreurs. La seconde stratégie présente également des cas d'erreurs, mais si on les excepte, la qualité de la réduction d'erreur est comparable à celle de la première stratégie pour une complexité calculatoire réduite. Finalement, les trois stratégies sont en général efficaces, bien que les deuxième et troisième présentent des cas d'échecs problématiques.

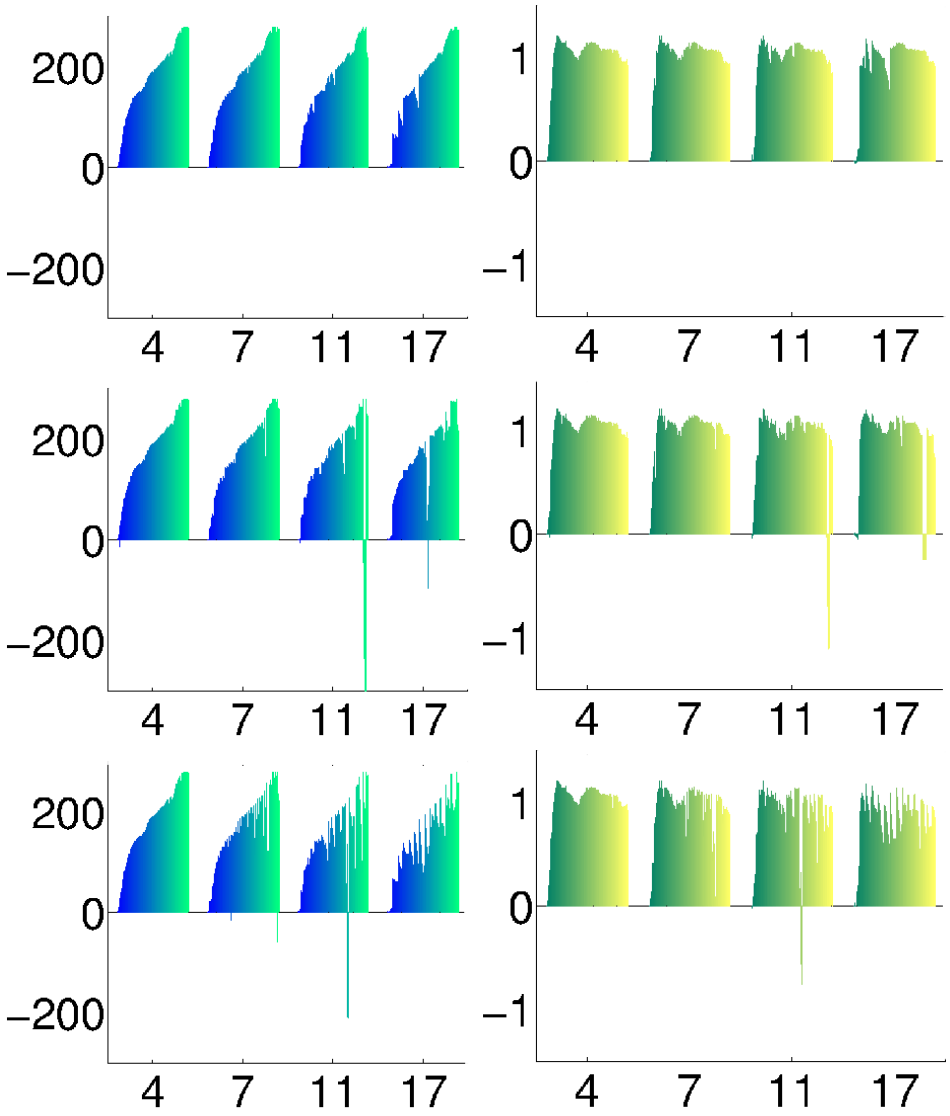


FIGURE 4.44 – Les réductions d'erreurs mesurées pour les stratégies de la première à la troisième stratégie, de haut en bas, pour quatre périodes de relocalisations indiquées sur l'axe des abscisses. La colonne de gauche montre les réductions d'erreur en translation et celle de droite en orientation. Les gradients de couleur indiquent les début et fin de trajectoire de la caméra. Pour les réductions d'erreurs en orientations l'axe des ordonnées est en radians, pour les réductions d'erreurs en translation l'unité de longueur est définie par le premier triplet de caméras clés.

4.7 Synthèse

Nous avons présenté dans ce chapitre l'algorithme de Mouragnon *et al*, [Mouragnon et al., 2009], d'après lequel nous avons modelé notre *SLAM* monoculaire. Nous avons détaillé les différences entre cette méthode et notre adaptation. En particulier nous avons introduit deux critères de vérification de la validité de la pose d'une caméra, afin de gagner de la robustesse à de mauvaises estimations. Nous avons également introduit une étape de densification des liens entre caméras afin de pallier les aléas de détection ou d'appariement des points d'intérêts. Nous avons montré deux applications de notre programme. La première est en milieu extérieur autour d'un bâtiment sur un longueur d'environ 100m, la trajectoire estimée est assez précise, mais la dérive du système est observable sur la carte de l'environnement. La seconde est une application de réalité augmentée où l'on dépose un objet virtuel dans la scène et l'on observe qu'il reste à sa position cependant que la caméra se déplace. Nous avons ensuite procédé à des évaluations du *SLAM* monoculaire sur 6 séquences correspondant à des mouvements presque *canoniques*, en translation et rotation, de la caméra. Nous avons constaté que les mouvements exhibant principalement une translation sont convenablement traités par l'algorithme. Les mouvements exhibant des rotations importantes au regard de la translation, se montrent en revanche plus difficiles à gérer, cette difficulté augmentant à mesure que la part de translation du mouvement s'amenuise. Nous avons également constaté que l'étape de densification est efficace et améliore les estimations de pose de la caméra tout en diminuant grandement les doublons de la carte. Nous avons suite à ces évaluations et applications énuméré les limites que nous avons constatées les plus importantes de notre *SLAM*. Dans une optique de correction de dérive du système nous avons fait l'hypothèse qu'un mécanisme de reconnaissance de lieu est disponible. Puisque ceci permettrait de relocaliser le système ayant dérivé, nous avons étudié la possibilité de corriger rapidement les poses de caméra précédant la relocalisation au travers d'un ajustement de faisceaux. Nous avons comparé trois stratégies et constaté que si elles ne sont efficaces qu'à un relativement court terme, il est possible d'obtenir des résultats raisonnables avec une complexité linéaire selon le nombre de poses à corriger.

Parmi les limites identifiées, celle concernant les mouvements de rotations est la plus contraignante. Elle est difficilement adressable dans le cadre du *SLAM* monoculaire. Il apparaît nécessaire d'ajouter un capteur permettant d'estimer la profondeur de la scène. C'est ce que nous allons voir à présent.

Chapitre 5

Slam multi-capteurs

Nous avons constaté que notre *SLAM* monoculaire offre des résultats satisfaisants tant qu'une carte de bonne qualité est disponible, mais aussi que cela ne peut pas être garanti. Dans les cas où la caméra suit un mouvement de rotation pure ou de rotation avec une translation suffisamment faible pour que son effet sur les mesures soit moins important que le bruit de détection, la triangulation de point ne peut que donner des résultats aberrants. Les mouvements de rotation pure sont détectables et il est possible de suspendre la création de caméra clé afin de prévenir les triangulations aberrantes, mais cela ne peut au mieux qu'éviter quelques cas d'échec en espérant que la quantité d'appariements reste suffisante pour contraindre le système ou que la caméra observe à nouveau une partie cartographiée, voir [Pirchheim et al., 2013] de Pirchheim *et al.* De manière plus générale, la triangulation nécessite d'avoir lieu fréquemment afin d'avoir toujours suffisamment d'information pour contraindre le système, diminuer l'influence du bruit de mesure et parce que l'apparence des points d'intérêt ne doit pas trop varier afin de pouvoir les apparier. Cependant la précision des points triangulés croît avec le déplacement des centres optique des caméras dans lesquelles ils sont mesurés. Ces deux objectifs étant contradictoires et la première contrainte étant la plus forte, cela conduit à la création d'une carte de mauvaise qualité dans certains cas. Ceci est renforcé par le fait que la pose relative entre deux caméras clés successives est elle même entachée d'erreurs. Nous estimons nécessaire d'ajouter un capteur permettant de capturer la profondeur.

Dans ce chapitre nous voyons comment modifier notre *SLAM* monoculaire pour faire usage de données de profondeur et présentons différentes modalités d'utilisation de ces informations. Nous montrons une application de

réalité augmentée pour une séquence exhibant des mouvements difficilement adressables par le *SLAM* monoculaire et voyons l'intérêt des données de profondeurs. Nous évaluons ensuite ces différentes modalités sur les séquences de mouvements *canoniques*. Nous présentons ensuite deux modifications de notre *SLAM* devenu multi-capteurs. La première s'appuie sur les résultats d'évaluation et vise à alléger le processus pour l'accélérer. La seconde, pour combler certaines limites du capteur, vise à rendre optionnelle l'utilisation de ces données. Finalement, nous voyons comment il est possible de faire usage de données d'orientations de la centrale inertielle pour alléger le processus d'estimation de pose utilisé pour filtrer les appariements *outliers*, et l'appliquons pour montrer les gains apportés en vitesse d'exécution et précision des estimations.

5.1 Slam visuel monoculaire avec profondeur

Des informations de profondeurs peuvent être obtenues avec plus de précision d'au moins deux manières. Il est possible d'ajouter une deuxième caméra ou bien une caméra de profondeur au système, par exemple de type *Kinect*. Ces dernières fournissent un accès direct à la profondeur en un pixel, mais ont un spectre limité. Certaines zones de l'image de profondeur peuvent ne pas contenir d'information et les environnements, en particuliers extérieurs, soumis à un rayonnement infra-rouge ne peuvent être cartographiés. Ajouter une deuxième caméra implique d'effectuer une mise en correspondance supplémentaire, ce qui nécessite des ressources calculatoires. Dans les deux cas les mesures de profondeur obtenues étant basées sur le principe de la triangulation elles sont entachées d'erreurs. Afin de limiter les besoins calculatoires de notre système nous avons choisi de faire usage d'une caméra de profondeur *Xtion Pro Live* d'*ASUS*.

Nous employons dans cette partie la bibliothèque *ceres*, [Agarwal et al.,], pour la minimisation des fonctions de coût.

5.1.1 Adaptation immédiate de l'approche monoculaire

Nous avons dans un premier temps simplement supprimé de notre algorithme de *SLAM* monoculaire l'étape de triangulation. Elle est remplacée par une lecture des données de profondeur pour tous les appariements *inliers* à chaque caméra. Nous ne conservons alors que les points d'intérêt pour les-

quelles une mesure de profondeur est disponible. Dans ce cas, en notant z_j^i et m_j^i les mesures de profondeur et d'image du point j dans la caméra i, sa position X_j^i dans le référentiel de la caméra i est

$$X_j^i = z_j^i \cdot m_j^i \tag{5.1}$$

A la différence de la bibliothèque de moindres carrés non linéaires *SSBA*, la bibliothèque *ceres* permet et nécessite de paramétrer la fonction de robustification, voir la partie D.1, employée. Nous utilisons toujours une fonction de *Huber*. On rappelle que cette fonction fait croître quadratiquement les valeurs inférieures à un seuil s prédéterminé, considérées comme *inliers*, et linéairement au delà. L'erreur étant à valeur dans $\mathbb{R}^2$, d'après les résultats du tableau 2.3 nous réglons s comme :

$$s = 5.99 \cdot \sigma_m^2 \quad \text{avec } \sigma_m = 1 \text{ pixel} \tag{5.2}$$

Contrairement au cas monoculaire dans lequel seuls les appariements entre caméras clés sont triangulés, nous pouvons ici créer un point de la carte pour chaque appariement entre la caméra courante et la caméra clé associée. La carte obtenue est plus riche, et l'estimation de pose de la caméra courante mieux contrainte, donc plus précise.

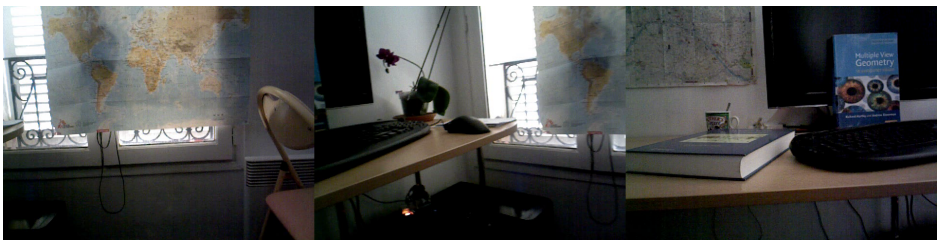


FIGURE 5.1 – Aperçu de la scène filmée dans la séquence Seq_ϕ.

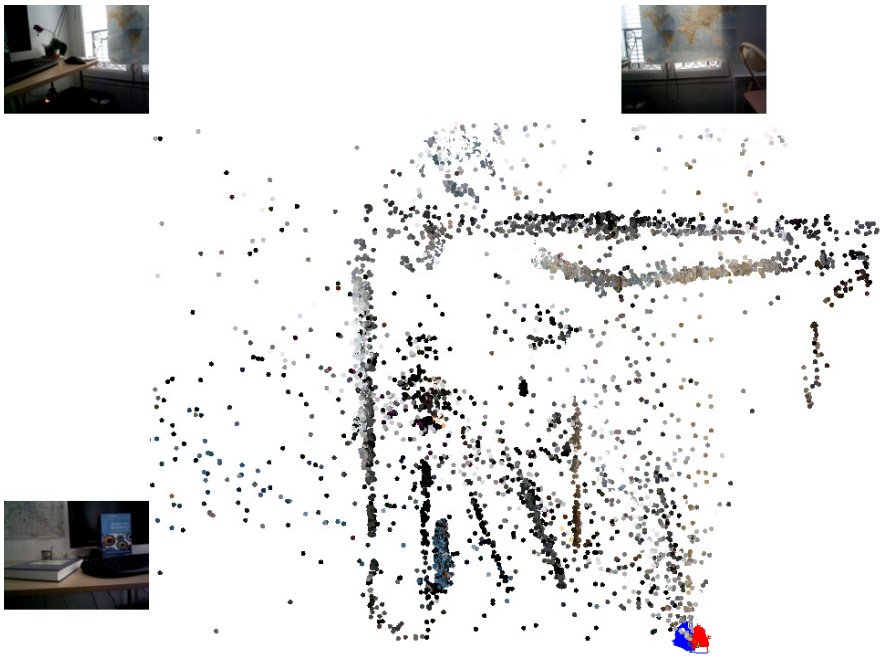


FIGURE 5.2 – Reconstruction et poses de la caméra pour la séquence Seq_ϕ, en utilisant les informations de profondeurs selon une approche monoculaire. La scène est vue de dessus.

Nous reprenons la séquence Seq_ϕ des évaluations du *SLAM* monoculaire, partie 4.4. On rappelle que la scène filmée correspond à un mouvement entre les trois images de la figure 5.1. Le résultat de l'application de cette modification est visible à la figure 5.2. La scène est vue d'en haut, c'est à dire selon $-Y$.

Nous pouvons observer que cette reconstruction aboutit, contrairement à ce qu'il se passait lorsque nous ne faisions pas usage de données de profondeurs. On constate toutefois que de nombreux points de la carte restent fortement erronés. Ceux-ci servant à estimer la pose de la caméra, c'est un problème. On peut par exemple voir qu'une caméra "en bas à droite" est mal estimée.

5.1.2 Modifications de l'approche

Fort de ce constat, il est apparu nécessaire de préserver les informations de profondeur qui sont trop souvent perdues lors des étapes de minimisation

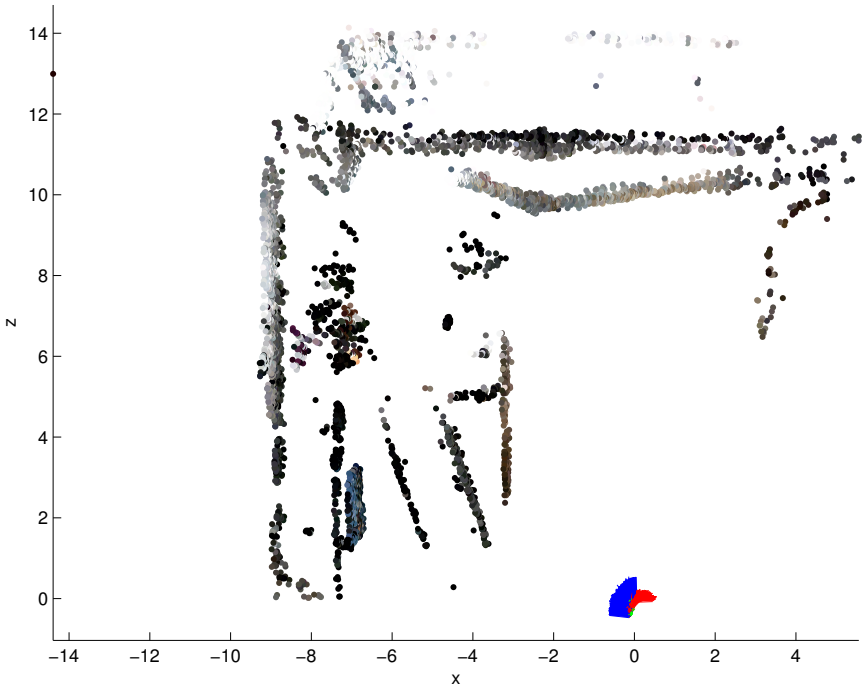


FIGURE 5.3 – Reconstruction de la séquence Seq_ϕ, en utilisant les informations de profondeurs. Les positions des points sont fixes lors des ajustements de faisceaux. La scène est vue de dessus.

d'erreurs.

Points fixes

Une première solution consiste à modifier l'ensemble des paramètres optimisables lors des applications d'ajustements de faisceaux de sorte que seuls ceux correspondant aux poses des caméras puissent varier. La figure 5.3 présente la reconstruction obtenue pour la séquence Seq_ϕ.

Comme attendu, on observe que la carte est nettement plus propre. Les poses de caméra sont plus précises : le décrochage précédemment observé ne se produit pas.

Puisque les points de la carte sont initialisés relativement à des poses de caméras, et que celles-ci sont modifiées lors des ajustements de faisceaux, nous nous sommes posé la question de la nécessité de ré-estimer ces positions lorsque les poses sont mises à jour. Nous avons constaté qu'en pratique les

différences produites sont marginales.

Fonction de coût dans $\mathbb{R}^3$

Il nous est donc apparu que l'adaptation directe de l'approche monoculaire dégrade les informations de profondeur puisque les ignorant lors des ajustements de faisceaux. Nous avons observé des reconstructions sensiblement plus précises lorsque nous forçons la prise en compte de la profondeur en désactivant l'optimisation des points de la carte. Cette solution n'est néanmoins pas idéale. Une approche plus satisfaisante est de prendre en compte les informations de profondeur lors des optimisations.

L'approche la plus directe consiste à définir la fonction de coût comme suit :

$$f(C^i, X_j) = C^i \cdot X_j - z_j^i \cdot m_j^i \tag{5.3}$$

Avec C^i la i-ème pose de la caméra, X_j la position du j-ème point de la carte, z_j^i et m_j^i les profondeur et coordonnées dans le plan images mesurées du point dans la caméra. Les erreurs minimisées sont donc définies dans $\mathbb{R}^3$.

Jusqu'à présent nous minimisions des erreurs exclusivement dans l'espace image de la caméra. Nous supposions que la variance des mesures était la même pour toutes. Il n'était alors pas nécessaire de l'inclure dans l'évaluation de la fonction de coût comme indiqué dans 1.3. Nous évaluions donc les fonctions de coût selon l'équation 1.15, puisque dans ce cas équivalente à l'équation 1.14. Cette hypothèse ne tient plus lorsque l'on utilise des données de profondeur. En effet, même en supposant que les données de profondeur ne sont pas bruitées, les positions des points de la carte étant évaluées selon l'équation 5.1, leurs variances doivent également être mises à l'échelle. Si une variable aléatoire x suit une loi $\mathcal{N}(\mu, \sigma^2)$, alors sa multiplication par un scalaire α suit une loi $\mathcal{N}(\alpha \cdot \mu, \alpha^2 \cdot \sigma^2)$. En faisant toujours l'hypothèse que les mesures de deux points distincts sont indépendantes, en notant σ_m l'écart type de l'erreur de mesure image, la variance dans $\mathbb{R}^3$ de la mesure du point j dans le référentiel de la caméra i devient proportionnelle à la valeur suivante :

$$(z_j^i \cdot \sigma_m)^2 \tag{5.4}$$

L'équation 5.3 est alors soit évaluée selon la formule 1.14 soit, c'est équivalent, selon la formule 1.15 en la modifiant comme suit :

$$f(C^i, X_j) = \frac{1}{z_j^i \cdot \sigma_m} \cdot C^i \cdot X_j - z_j^i \cdot m_j^i \tag{5.5}$$

La précision annoncée du capteur étant de $\sigma_z = 1$ cm, l'erreur étant évaluée dans $\mathbb{R}^3$, en utilisant les résultats du tableau 2.3, nous prenons pour seuil de modification de comportement de la fonction de *Huber* la valeur :

$$s = min\left(7.81 \cdot \sigma_z^2\ ,\ 7.81 \cdot (z_j^i \cdot \sigma_m)^2\right) \tag{5.6}$$

Il est bien sûr toujours possible de fixer les points de la carte en utilisant cette fonction de coût.

Fonctions de coût selon *Scherer et al*

Il a été observé dans quelques travaux, [Herrera et al., 2012] [Scherer et al., 2012] [Smisek et al., 2013] [Teichman et al., 2013], que les mesures de profondeur sont bruitées et que la modélisation de ce bruit n'est pas triviale. Il est en particulier fonction de la profondeur, et de la position du point dans l'image.

Parallèlement au travail que nous venons de présenter, dans [Scherer et al., 2012] Scherer *et al* ont proposé d'utiliser deux fonctions de coût afin de prendre en compte les différentes variances des mesures image et de profondeur. Ainsi, chaque point de la carte contribue à deux fonctions de coût. La première étant la fonction de coût usuelle d'erreur de reprojection :

$$f_m(C^i, X_j) = \frac{1}{\sigma_m} \cdot (p(C^i \cdot X_j) - m_j^i) \tag{5.7}$$

La seconde ne concerne que les mesures de profondeur :

$$f_z(C^i, X_j) = \frac{1}{\sigma_z} \cdot ((C^i \cdot X_j)_{[3]} - z_j^i) \tag{5.8}$$

Où l'écart type sur les mesures de profondeur est approximé par $\sigma(z^2) = 3.331 \cdot 10^{-3} \cdot z^2$.

Pour chaque fonction de coût, d'après le tableau 2.3, nous prenons pour seuil de modification de comportement de la fonction de *Huber* les valeurs :

$$s_m = 5.99 \cdot \sigma_m^2 \tag{5.9}$$

et

$$s_z = 3.84 \cdot \sigma_z^2 \tag{5.10}$$

5.1.3 Application

Nous présentons à présent une application de réalité augmentée bénéficiant de l'utilisation du capteur de profondeur. Nous avons acquis une séquence présentant un mouvement de rotation important afin de comparer les comportements du *SLAM* monoculaire et du *SLAM* multi-capteurs. Quelques vues de la séquence sont visibles à la figure 5.4.

FIGURE 5.4 – Quelques vues de la séquence employée

La rotation du capteur est effectuée entre les 3-ième et 4-ième images de la figure 5.4. Nous avons placé dans la carte un objet virtuel à la position correspondant à un point d'intérêt renseigné préalablement à l'exécution. La figure 5.4 montre deux captures d'écran, en début et fin de séquence, c'est à dire avant et après le mouvement de rotation du capteur, pour le *SLAM* monoculaire et le *SLAM* multi-capteurs, utilisant la fonction de coût de Scherer *et al.*

Sur cette figure, on observe que le *SLAM* monoculaire dérive fortement, ce qui est conforme aux résultats des évaluations de la partie 4.4. Les informations de profondeur s'avèrent ici précieuses, la dérive observable étant nettement plus modérée dans le cas du *SLAM* multi-capteurs.

FIGURE 5.4 – Application de réalité augmentée, le dragon est virtuel. Les images de gauche correspondent aux captures du *SLAM* monoculaire, celles de droite aux captures du *SLAM* multi-capteurs. La première ligne correspond à des images en début de séquence, la seconde à des images de fin de séquence.

5.2 Evaluations

Les séquences utilisées pour l'évaluation du *SLAM* monoculaire ayant été enregistrées avec un capteur de couleur plus profondeur, nous reprenons les mêmes séquences pour l'évaluation des différentes stratégies d'utilisation des informations de profondeur.

Les profondeurs des différentes scènes n'excédant pas un mètre, nous nous plaçons donc dans les conditions d'utilisation recommandées par le constructeur du capteur de profondeur. Dans ces conditions il a été observé, [Chow et al., 2012], [Teichman et al., 2013], que les mesures de profondeurs sont modérément erronées.

Nous évaluons les 3 formulations de fonctions de coût précédemment présentées. Nous nommons :

– $2D$ la fonction de coût pour l'erreur de reprojection usuelle.
– $3D$ la fonction de coût pour l'erreur évaluée dans $\mathbb{R}^3$ présentée dans la

partie 5.1.2.

– $2{+}1D$ l'aggrégat de la fonction l'erreur de reprojection et de la fonction d'erreur de profondeur de Scherer *et al* présentée dans la partie 5.1.2.

Pour chaque formulation, nous évaluons deux modalités de gestion des points de la carte. Selon que ceux soient optimisables ou fixes, on nomme ces modalités *opt* et fix.

		$2D$		$3D$		$2+1D$		*Moyenne*	
		opt	fix	opt	fix	opt	fix	opt	fix
Seq_X	cm	6.41	4.98	3.46	**2.06**	4.93	3.11	4.93	3.38
	◦	10.18	8.82	7.57	**6.52**	9.24	7.46	9.0	7.6
Seq_Y	cm	1.31	0.51	0.76	0.39	0.61	**0.34**	0.89	0.41
	◦	3.58	3.94	3.62	**3.51**	3.68	3.65	3.62	3.70
Seq_Z	cm	1.99	1.77	3.35	3.02	2.58	**1.59**	2.64	2.13
	◦	1.17	1.08	1.80	1.58	1.48	**0.99**	1.48	1.22
Seq_θ	cm	0.38	**0.15**	1.09	0.89	0.53	0.47	0.67	0.5
	◦	3.48	**3.14**	3.74	3.36	3.20	3.21	3.47	3.24
Seq_ϕ	cm	1.72	**0.87**	2.71	3.50	1.36	1.15	1.93	1.83
	◦	2.98	3.62	2.66	**2.59**	3.75	3.48	3.13	3.23
Seq_ψ	cm	4.50	0.41	0.58	0.30	0.63	**0.18**	1.9	0.29
	◦	4.43	1.31	0.68	**0.66**	1.17	0.84	2.09	0.94
Moy.	cm	2.72	1.45	1.99	1.69	1.77	1.14		
	◦	4.3	3.65	3.35	3.04	3.75	3.27		

TABLE 5.1 – Erreurs de pose entre la première et la dernière caméra pour les six séquences. Pour chaque séquence et chaque composante le meilleur résultat est affiché en rouge et en gras.

Comme précédemment, les trajectoires bouclent, il est donc possible d'estimer la dérive du système en mesurant la différence entre la première et la dernière caméra. Un tableau comparatif des erreurs pour ces six stratégies est disponible au tableau 5.1. Les erreurs sont évaluées en translation, par la distance entre les centres optiques, et orientation, par la norme du vecteur de $\mathbb{R}^3$ formé à partir des angles entre les axes des caméras.

On peut observer sur le tableau que la prise en compte de la profondeur lors des minimisations d'erreur est plus souvent bénéfique que l'inverse. D'après ces expérimentations la méthode $2+1D$ semble la plus précise.

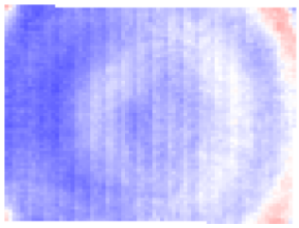


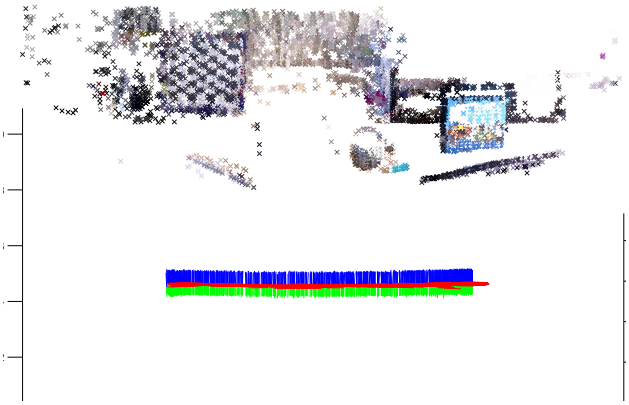
FIGURE 5.5 – Exemple d'une image de coefficients de distorsion de profondeur, image tirée de [Teichman et al., 2013].

On observe que les séquences Seq_X, Seq_Z et Seq_ϕ présentent plus d'erreurs que les autres. La forme des images, rectangulaire, implique qu'un point peut être observé relativement plus longtemps dans Seq_X et Seq_ϕ que dans Seq_Y ou Seq_θ. La nature du déplacement de Seq_Z induit le même phénomène. Nous conjecturons que ces différences de précisions sont dûes à la variation des distorsions de profondeur. Sous cette hypothèse, la précision obtenue pour la séquence Seq_ψ n'est pas impactée puisque la caméra tournant autour de son axe de visée la variation de cette distorsion est plus limitée. A fin de rappel, la figure 5.5, tirée de [Teichman et al., 2013], montre une image de distorsion de profondeur.

Afin de visualiser les gains apportés par l'ajout d'information de profondeur nous regardons à présent avec plus de détails les résultats obtenus pour chaque séquence en sélectionnant la meilleure méthode.

Seq_X : Translation selon l'axe X de la caméra

Méthode employée : $3d\_fix$

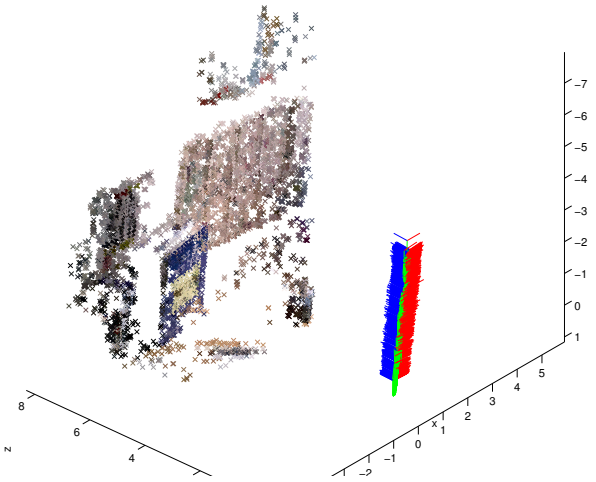


	$3d\_fix$
Nombre de caméras :	1031
Nombre de caméras clés :	40
Nombre de points :	9363
Quantité d'inliers :	99.99%

FIGURE 5.6 – Reconstruction et caméras

$\mathbf{Seq}_Y$: Translation selon l'axe Y de la caméra

Méthode employée : $2 + 1d\_fix$

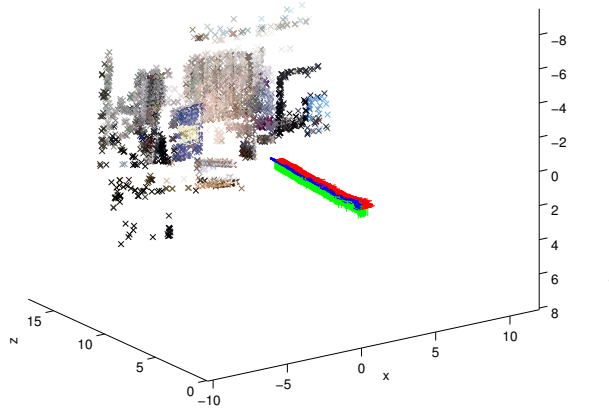


	$3d\_fix$
Nombre de caméras :	1010
Nombre de caméras clés :	28
Nombre de points :	9134
Quantité d'inliers :	100%

FIGURE 5.7 – Reconstruction et caméras

$\mathbf{Seq}_Z$: Translation selon l'axe Z de la caméra

Méthode employée : $2 + 1d\_fix$

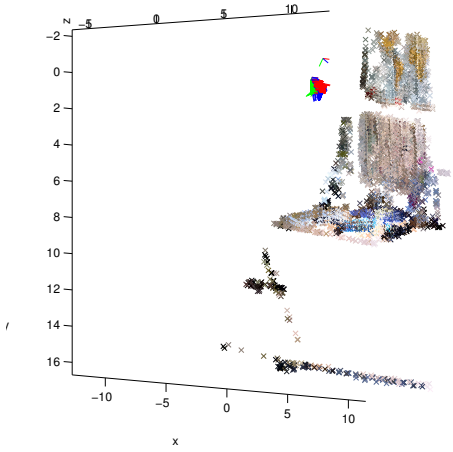


	$3d\_fix$
Nombre de caméras :	1519
Nombre de caméras clés :	14
Nombre de points :	4337
Quantité d'inliers :	99.58%

FIGURE 5.8 – Reconstruction et caméras

$\mathbf{Seq}_\theta$: Rotation autour de l'axe X et légère translation

Méthode employée : $2d\_fix$



	$3d\_fix$
Nombre de caméras :	1058
Nombre de caméras clés :	28
Nombre de points :	8247
Quantité d'inliers :	98.95%

FIGURE 5.9 – Reconstruction et caméras

$\mathbf{Seq}_\psi$: Rotation autour de l'axe Z et légère translation

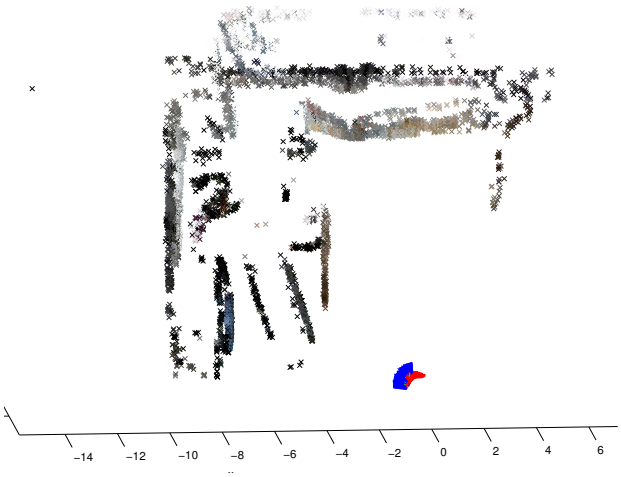
Méthode employée : $2 + 1d\_fix$

	$3d\_fix$
Nombre de caméras :	756
Nombre de caméras clés :	74
Nombre de points :	12810
Quantité d'inliers :	100%

FIGURE 5.10 – Reconstruction et caméras

$\mathbf{Seq}_{\phi}$: Rotation pure autour de l'axe Y

Méthode employée : $2d\_fix$



	$3d\_fix$
Nombre de caméras :	783
Nombre de caméras clés :	37
Nombre de points :	7431
Quantité d'inliers :	93.78%

FIGURE 5.11 – Reconstruction et caméras

Comme attendu les résulats sont de meilleure qualité que pour le *SLAM* monoculaire : les cartes sont plus détaillées, plus précises et les mouvements problématiques pour la cas monoculaire ne le sont plus. On peut toutefois

observer à la séquence Seq_θ qu'une caméra a décroché de la trajectoire. Les résultats ne sont donc pas exempts d'erreurs.

Il est intéressant de constater que fixer les points de la carte produit de meilleurs résultats que de les optimiser. Cela tient probablement au fait que les faibles profondeurs des différentes scènes permettent d'obtenir des mesures de profondeur de bonne qualité.

5.3 Extensions

Nous présentons à présent deux variations de notre *SLAM* avec profondeur. La première vise à diminuer le coût calculatoire de la méthode en se basant sur l'observation que fixer les points de carte permet d'obtenir des résultats de bonne qualité. La seconde vise à étendre les cas d'utilisation du *SLAM* avec profondeur aux cas où cette information n'est pas toujours disponible.

5.3.1 Un schéma de *SLAM* léger en environnement contraint

Nous avons constaté sur les séquences d'évaluation sur des environnements de faible profondeur des précisions comparables pour les approches *fix* et *opt*. Bien que le raffinement des estimations ne soit pas toujours la partie la plus consommatrice de ressources calculatoires de notre implémentation, cette observation est intéressante en ce qu'elle nous enseigne que dans ces environnements il est possible d'alléger le processus sans dégrader les estimations. Dans les expérimentations, le coût calculatoire d'une optimisation non linéaire avec points fixes est en moyenne entre 3 et 4 fois plus faible qu'avec des points optimisables.

Les modifications jusqu'à présent effectuées quant à la prise en compte des mesures de profondeurs n'ont porté que sur les optimisations sur fenêtre glissante appliquées à l'issue de la définition de nouvelles caméras clés. On rappelle que dans notre *SLAM* la dernière étape du processus d'estimation de pose de chaque caméra est un ajustement de faisceaux ne définissant que les paramètres de pose comme ajustables. Mais ce raffinement peut être également effectué selon l'une des fonctions de coût prenant en compte les informations de profondeur, présentées dans la partie 5.1.

Dans ce cas les optimisations locales post détection de caméra clé ne sont plus nécessaires. En effet, si l'on empêche les points de la carte de varier, et puisque la position de chaque point est définie d'après les mesures lues de la première caméra clé l'ayant observé, toute caméra se trouve à l'optimum lorsqu'elle est déclarée clé. Les optimisations sur fenêtre glissantes n'apportent donc pas d'amélioration. Ainsi cette étape peut être supprimée de l'algorithme.

En reprenant la séquence employée pour la deuxième application de *SLAM* monoculaire, partie 4.3.2, le temps de traitement de la séquence est de 558 s pour l'approche $3d\_opt$ et de 506 s pour ce schéma allégé. Le temps de traitement est donc réduit de 10%. Dans le premier cas, l'erreur de pose en translation de la dernière caméra est de 0.35 cm et dans le second de 0.53 cm. En utilisant la bibliothèque *SSBA*, la réduction de temps de traitement est encore plus significative puisque l'on passe de 490 s à 409 s, soit une réduction de 15%, pour des erreurs de pose, en translation, de 0.13 cm et 0.45 cm.

5.3.2 Un *SLAM* hybride tirant optionnellement partie de la profondeur

Nous savons que les motifs infra-rouges projetés par les capteurs de profondeur bas-coût sont perturbés par la lumière solaire. Ceci limite donc les conditions de leur utilisation. Une approche selon laquelle ces informations sont indispensables ne peut donc donner de bons résultats dans ce cas.

Pour tirer partie des informations de profondeur nous avons modifié au minimun notre algorithme de *SLAM* monoculaire. Nous n'avons pris en compte que les mesures images ayant une mesure de profondeur associée, remplacé la procédure de triangulation de points par la lecture des ces données, et modifié les processus d'optimisation. Pour pallier les éventuelles absences d'information de profondeur, nous proposons alors de prendre à nouveau en compte l'ensemble des mesures images, et de trianguler celles pour lesquelles aucune information de profondeur n'est disponible. Les points de la carte ainsi obtenus peuvent alors ne pas avoir de profondeur associée dans une, plusieurs ou toutes les caméras l'observant. Il faut donc mettre à jour les fonctions de coût évaluées lors des minimisation d'erreur. Si une donnée de profondeur est disponible pour chaque couple $\{C^i, P_j\}$ les optimisations appliquées sont effectuées d'après les fonctions de coût proposées par Scherer

et al, 5.7 et 5.8, et P_j est fixe. Sinon les optimisations sont effectuées d'après la fonction coût de reprojection, 5.7, et lors des ajustements de faisceaux sur fenêtre glissante P_j est optimisable.

Nous avons acquis une séquence en filmant un environnement partiellement ensoleillé et parfois vu de suffisamment près pour que le capteur ne puisse obtenir d'informations de profondeur. La figure 5.13 présente 12 vues de couleur représentatives de la séquence.

Nous présentons les images de profondeur correspondant aux quatre premières images couleur de la figure 5.13, à la figure 5.14. Les images sont seuillées, les zones blanches portent des informations de profondeur, les zones noires n'en portent pas. Nous observons toutefois que des données de profondeurs sont présentes en quantité raisonnable. Afin de simuler des situations plus difficiles nous avons supprimé les informations de profondeurs de certaines images. Les images affectées correspondent à la première observation de "l'extrémité droite" de la scène. La figure 5.15 montre les quatre premières images de profondeur de cette séquence modifiée.

Nous obtenons donc deux séquences, on les nomme *ensoleillée* et *modifiée*. On compare deux schémas de *SLAM*, *2+1d_fix* et on nomme le second, incorporant les modifications présentées, *hybride*. Le tableau 5.12 présente les erreurs en translation et orientation de la pose de la dernière caméra.

		ensoleillée	*modifiée*
2+1d_fix	cm	1.22	3.44
	◦	15.24	59.24
hybride	cm	0.4	1.6
	◦	7.16	18.58

FIGURE 5.12 – Tableau comparatif des schémas de *profondeur* et *hybride* pour les séquences *ensoleillée* et modifiée. Ces séquences présentent la particularité de contenir peu d'informations de profondeur.

Comme attendu, la séquence *hybride* est sensiblement plus robuste au manque d'informations de profondeur.

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

FIGURE 5.13 – Séquence d'expérimentation du schéma de *SLAM* hybride, images de couleur.

(a) (b) (c) (d)

FIGURE 5.14 – Séquence d'expérimentation du schéma de *SLAM* hybride, images de profondeur.

(a) (b) (c) (d)

FIGURE 5.15 – Séquence d'expérimentation du schéma de *SLAM* hybride, images de profondeur modifiées.

5.4 Utilisation de la centrale inertielle

Nous présentons à présent une utilisation d'une centrale inertielle pour l'estimation de pose relative de manière efficace en présence d'une quantité significative d'appariements *outliers*.

5.4.1 Théorie

Une centrale inertielle peut fournir des mesures d'orientation modérément affectées par le phénomène de dérive. Cela signifie que même si l'orientation absolue de la centrale est de moyenne qualité, la variation entre deux prises de vues successives est tout de même fiable.
Puisque l'on cherche à estimer des poses relatives, c'est à dire une orientation et une translation, il n'est plus nécessaire de rechercher l'orientation. Nous proposons donc la méthode suivante pour déterminer la translation connaissant l'orientation, représentée par la matrice R, et les appariements m^1 et m^2 des mesures d'un point X de $\mathbb{R}^3$ dans les images des caméras C et C^2. Les points m^1 et m^2 sont notés en coordonnées normalisées et homogènes.
D'après les contraintes épipolaires on sait que pour un couple (m^1, m^2) :

$$ {m^2}^\top \cdot t_\times \cdot R \cdot m^1 = 0 \tag{5.11} $$

Notant rm^1 le vecteur $R \cdot m^1$, on a :

$$ \begin{aligned} & {m^2}^\top \cdot t_\times \cdot rm^1 = 0 \\ \Leftrightarrow & \\ & \begin{pmatrix} m_x^2 & m_y^2 & m_z^2 \end{pmatrix} \cdot \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \times \begin{pmatrix} {rm^1}_x \\ {rm^1}_y \\ {rm^1}_z \end{pmatrix} = 0 \\ \Leftrightarrow & \\ & \left(\begin{pmatrix} {rm^1}_x \\ {rm^1}_y \\ {rm^1}_z \end{pmatrix} \times \begin{pmatrix} m_x^2 \\ m_y^2 \\ m_z^2 \end{pmatrix} \right)^\top \cdot \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} = 0 \end{aligned} \tag{5.12} $$

Puisque la géométrie épipolaire n'est définie qu'à un facteur d'échelle, la translation t que nous recherchons peut être en toute généralité considérée de norme égale à 1. Dans ce cas l'espace de recherche de la solution est un

espace à deux dimensions. A partir de R et des éléments de m^1 et m^2, on note l_i le produit vectoriel mis en évidence dans la relation 5.12.

$$l_i = \left(\begin{pmatrix} {rm^1}_x \\ {rm^1}_y \\ {rm^1}_z \end{pmatrix} \times \begin{pmatrix} m^2_x \\ m^2_y \\ m^2_z \end{pmatrix} \right)^\top \tag{5.13}$$

Ainsi à partir de deux appariements (m^1_i, m^2_i) et (m^1_j, m^2_j) il est possible de former des lignes l_i et l_j et de les concaténer dans une matrice

$$M = \begin{pmatrix} l_i \\ l_j \end{pmatrix} \tag{5.14}$$

Cette matrice M est alors de taille 2×3. Dans le cas général, l_i et l_j sont indépendants, alors selon le théorème du rang la dimension de l'image de M est égale à deux et la dimension du noyau est égale à un. La translation t recherchée est ainsi un vecteur du noyau de M. Une solution peut être obtenue par l'application d'une *SVD*.

Si l_i et l_j sont liés il existe α tel que $l_i = \alpha \cdot l_j$ et le rang de M est égal à 1. Ce cas potentiellement dégénéré est aisément mesurable.

Lorsque la quantité de mauvais appariements est importante, la connaissance de la matrice de rotation est particulièrement intéressante dans un schéma d'estimation de matrice essentielle robuste, par *RANSAC*. A titre d'exemple, selon [Hartley and Zisserman, 2004], afin d'obtenir une certitude égale à 99% que le modèle sélectionné est le bon la quantité d'essais nécessaire est égale à 26 avec 5 points et de 7 avec 2 points pour une quantité de 30% d'outliers. Pour 50% d'outliers ces nombres passent à 146 et 17.
Cela peut être particulièrement utile en conjonction de descripteurs de points d'intérêts binaires, voir 2.8, dont l'appariement est très rapide mais dont les résultats comportent beaucoup d'outliers.
C'est d'une manière générale bien adapté au *SLAM* basé sur des points d'intérêt. En effet, le caractère incrémental du problème conduit souvent à estimer des variations très faibles de poses, or dans ce cas la géométrie épipolaire est faiblement conditionnée, et des algorithmes efficaces tels que décrit dans 2.3 fournissent de mauvaises estimations de pose.

Notons par ailleurs, que la complexité de la méthode est raisonnable, selon [Hartley and Zisserman, 2004] le coût de la *SVD* est de 465 opérations flottantes pour la décomposition complète et de 264 pour la décomposition partielle qui est dans ce cas suffisante.

Cette méthode a également et indépendamment été devisée par Kneip *et al* dans [Kneip et al., 2011].

5.4.2 Application

Comme indiqué dans la partie concernant le *SLAM* monoculaire, nous avons privilégié l'utilisation de points *SURF* aux détecteurs et descripteurs rapides de points d'intérêt. Ayant dans un premier temps prototypé notre adaptation en langage *Matlab*, nous avions observé que la quantité importante d'appariements *outliers* induisait un plus grand nombre d'itérations dans la procédure *RANSAC*. A tel point que le temps gagné lors des détections et descriptions était perdu lors du filtrage des appariements.

A présent munis de la méthode présentée nous pouvons filtrer efficacement les appariements erronés. Nous utilisons conjointement un détecteur de *Harris* et *CenSurE* avec le détecteur *BRISK*. On nomme cet agrégat *HCB*. Nous les employons dans notre *SLAM* avec profondeur. Puisque nous recherchons dans le cas présent à améliorer la vitesse de traitement de notre algorithme, nous utilisons la bibliothèque *SSBA*, selon l'approche $2D\_fix$.

Dans le tableau 5.2 nous examinons l'influence de l'utilisation de données inertielles dans notre *SLAM* de profondeur, selon que l'on utilise le détecteur-descripteur *SURF* ou *HCB*.

		SURF	*SURF+imu*	*HCB*	*HCB+imu*
#points par image en moy.		658		287.7	
#itérations *RANSAC* en moy.		2.61	1.57	15.3	4.4
proportion d'*outliers* en moy.		5.54%	4.29%	28.38%	28.93%
framerate en moy.		2.65	2.88	5.08	8.9
précision	cm	0.39	0.27	0.96	0.66
de la dernière pose	°	3.3	1.70	7.96	5.53

TABLE 5.2 – Comparaison de l'influence des données inertielles selon deux types de points d'intérêt.

Alors nous ne l'attendions pas on observe un gain en précision pour les deux types de points d'intérêt. Les gains en vitesse de traitement sont par contre conformes à ce que nous attendions. Le gain est significatif pour l'agrégat *HCB*, produisant beaucoup d'appariements *outliers*, et anecdotique pour le *SURF*, dont les appariements sont de très bonne qualité.

5.5 Synthèse

Nous avons présenté une adaptation de notre *SLAM* monoculaire permettant d'utiliser des informations de profondeur issues d'un capteur *Xtion Pro Live*. Une application de réalité augmentée a montré l'intérêt de ce capteur pour une séquence comportant des mouvements difficilement adressables par le *SLAM* monoculaire. Nous avons vu plusieurs modalités d'utilisation des informations de profondeur et les avons évaluées sur les 6 séquences de mouvements canoniques. Ces évaluations ont montré qu'il est préférable d'optimiser séparément les fonctions de coût d'erreurs de reprojection et de profondeur. Parce que les scènes sont de faible profondeur il s'est également avéré plus intéressant de fixer les points de la carte lors des optimisations. Dans ce cadre de scène de faible profondeur, nous avons devisé une première modification allégée du *SLAM*. Nous avons présenté une seconde modification de l'algorithme rendant l'utilisation des données de profondeur optionnelles. En effet, celles-ci ne sont pas toujours disponibles, en particulier pour des zones directement illuminées par la lumière solaire. Nous avons décrit comment les informations d'orientation fournies par la centrale inertielle peuvent être utilisées pour simplifier le calcul de pose relative, et donc le filtrage d'appariements de points d'intérêt. Une application a montré des gains en précision et dans le cas de détecteurs et descripteurs rapides, fournissant des appariements erronés en quantité non négligeable, des gains en vitesse de traitement.

Conclusion et perspectives

Travaux

Au cours de cette thèse nous nous sommes intéressés à résoudre la tâche du *SLAM* visuel dans une optique d'application en réalité augmentée.

Nous nous sommes dans un premier temps intéressés au *SLAM* monoculaire. Nous nous sommes inspirés d'une méthode ayant fait ses preuves que nous avons modifiée afin de réduire les cas d'erreur. Nous avons montré au travers de deux applications, une de localisation et cartographie à grande échelle et l'autre de réalité augmentée, son efficacité. Nous avons ensuite procédé à une évaluation du système sur 6 séquences présentant des mouvements *canoniques* de la caméra. Nous avons observé que les mouvements de translation sont correctement pris en compte, mais que les rotations sont problématiques. Ayant constaté dans les applications et au travers des évaluations une dérive des estimations, nous avons étudié un moyen rapide, basé sur des ajustements de faisceaux, permettant de corriger des poses précédant une relocalisation du système.

Les évaluations ayant démontré que le *SLAM* monoculaire n'est pas en mesure de gérer des mouvements de la caméra présentant une rotation importante au regard de la translation, nous nous sommes ensuite intéressés au *SLAM* visuel multi-capteurs utilisant un capteur de profondeur en plus d'une caméra. Nous avons montré comment nous adaptons notre solution de *SLAM* monoculaire à l'utilisation de données de profondeur et avons décrit différentes modalités de prise en compte de ces informations. Une application de réalité augmentée sur une séquence présentant des mouvements importants de rotation a montré une précision sensiblement accrue de ce *SLAM* multi-capteurs par rapport au *SLAM* monoculaire. Nous avons ensuite évalué les différentes modalités de prise en compte de la profondeur. Nous avons vu que minimiser séparément erreurs de reprojection et de pro-

fondeur offre généralement de meilleures estimations. Nous avons également constaté que fixer les points de la carte sur ces zones de faible profondeur offre également de meilleurs résultats. S'appuyant sur ce constat, dans le cadre de zones de faible profondeur nous avons proposé une modification du *SLAM* multi-capteurs visant à alléger les besoins en ressources calculatoires. Une application a montré des temps de calculs diminués de 15%. Parce que le capteur de profondeur n'est pas toujours en mesure de fournir ses données, nous avons proposé une deuxième modification du *SLAM* multi-capteurs. Cette modification vise à rendre l'utilisation de ces informations optionnelle, et fonctionner en mode monoculaire lorsqu'elles ne sont pas à disposition. Une application a montré l'intérêt de cette approche. Dans un dernier temps, nous avons fait usage d'une centrale inertielle et, puisque celle ci permet de connaître la rotation relative, nous avons montré comment calculer la translation relative entre deux caméras à partir de cette connaissance et de 2 paires de points d'intérêt. L'intérêt résidant principalement dans l'accélération du filtrage des appariements une application à montré que lorsque ceux-ci contiennent une proportion non négligeable d'outliers les temps de calcul sont considérablement amoindris. Nous avons également constaté une amélioration de la précision des estimations.

Perspectives

SLAM monoculaire
Nous avons utilisé des caméras usuelles à faibles angles d'ouverture dans nos applications. La fiabilité des estimations du *SLAM* serait améliorée en faisant usage d'une caméra grand angle. Le suivi de points pouvant être alors mieux assuré, leurs estimations de positions seraient de meilleure qualité, et alors les poses de caméra pourraient être mieux estimées. Il se peut que l'ajout d'un modèle de mouvement accélère le filtrage des appariements de points d'intérêt. Enfin, notre étape de densification des liens ne s'applique que sur les caméras proches temporellement. Une adaptation aux caméras proches spatialement, probablement en prenant en compte les incertitude de position, pourrait s'avérer précieuse.

SLAM multi-capteurs
Une intégration plus fine des capteurs présenterait un grand d'intérêt. En particulier, une calibration des bruits de profondeur en fonction de la distance à

la scène en chaque pixel de la caméra de profondeur permettrait d'obtenir des estimations de l'environnement plus précises. Pouvoir estimer l'orientation et la position de la caméra à partir des données inertielles serait très utile. Cela pourrait permettre d'effectuer l'étape de robustification des appariements en une itération, ou encore de détecter des objets mobiles. L'utilisation de la centrale inertielle pourrait être étendue. Les données pourraient être utilisées pour détecter les mauvaises estimations de pose. Elles pourraient également être incluses dans les termes des fonctions de coût optimisées.

Réalité augmentée
Cela se voit peu sur des captures d'écran mais un effet de gigue des augmentations se produit parfois. Celles-ci apparaissent alors tremblotantes, ce qui nuit au réalisme. Il pourrait être intéressant d'effectuer une estimation de pose *en deux passes*. Considérons disponibles des points correspondant aux augmentations, probablement en les prenant comme les projections des sommets des objets virtuels, et appelons les "mesures". Suite à la procédure d'estimation de pose il est possible de former la pose relative à la précédente caméra. Nous pourrions alors affiner cette pose relative en minimisant les erreurs formées par la contrainte épipolaire entre les "mesures" dans l'image précédente et les projections des points correspondant dans l'image courante. Il pourrait même être souhaitable d'appliquer ce processus sur plusieurs images précédentes. Nous pensons que ceci aurait pour résultat de réduire les effets de tremblement.

Temps de calcul
Notre solution n'est pas temps réel. Quelques mauvais choix d'implémentation ont été commis. Par ailleurs, de nombreuses tâches parallèles bénéficieraient d'une adaptation à une architecture *multicore*, et celles massivement parallèles d'extraction et de description de points d'intérêt gagneraient à être exécutées sur *GPU*. Les capacités de notre machine se sont avérées trop modestes sur ces deux derniers points.

Appendices

Annexe A

Ajustement de faisceaux

Les paramètres que nous estimons lorsque nous résolvons le problème du *SLAM* sont inévitablement entachés d'erreurs. La manière classique de raffinement de ces paramètres consiste à minimiser l'erreur de reprojection des points de la carte dans les images de la caméra. L'erreur de reprojection est la distance $d()$ entre le point P_j reprojeté dans la caméra C^i en $h(C^i, P_j)$ et son observation m_j^i. Les erreurs de reprojection des points et paramètres de poses ajustés sont l'image de la fonction $f : \mathbb{R}^{n+m} \to \mathbb{R}$ de la forme suivante :

$$f\left(C^0 \dots C^i \dots C^n \; P_0 \dots P_j \dots P_m\right) = \frac{\left(\dots d\left(h(C^i, P_j), m_j^i\right) \dots\right)^\top}{2} \cdot \begin{pmatrix} \vdots \\ d\left(h(C^i, P_j), m_j^i\right) \\ \vdots \end{pmatrix} \tag{A.1}$$

Cette fonction n'étant pas linéaire, une méthode classique de minimisation au sens des moindres carrés n'est pas applicable.

Pour simplifier les notations nous notons à présent $X = \left(\dots C^i \dots P_j \dots\right)^\top$ et $f : \mathbb{R}^n \to \mathbb{R}$.

On fait l'hypothèse que $f()$ est différentiable. Si elle l'est au moins deux fois, un développement limité de $f()$ au voisinage de X à l'ordre deux, donne :

$$f(X + \delta) = f(X) + J_X \cdot \delta + \frac{1}{2} \cdot \delta^\top \cdot H_X \cdot \delta + \|\delta\|_2^{\,2} \cdot \epsilon(\delta) \tag{A.2}$$

Où J_X et H_X sont les matrices jacobiennes et hessiennes de $f()$ évaluées en X et $\epsilon(\delta)$ une fonction qui tend vers 0 lorsque $\|\delta\|_2$ tend vers 0.

La fonction $f()$ n'étant pas linéaire, il n'est généralement pas possible de déterminer son minimum global en temps fini. Pour résoudre ce problème il est nécessaire de connaître une estimation initiale X_0 des paramètres proche

du minimum global. La solution est ensuite recherchée par raffinements itératifs. A chaque étape i de nouveaux paramètres X_i sont estimés dont l'erreur $f(X_i)$ est inférieure à celle de l'étape précédente $f(X_{i-1})$.

A.1 Descente de gradient

La méthode de la descente de gradient ne requiert que $f()$ ne soit qu'une fois différentiable, elle consiste à modifier les paramètres dans le sens de l'inverse de la direction du gradient. La jacobienne J_X est aussi la transposée du gradient de $f()$, ∇_X, en X. L'itération $i+1$ donne alors pour résultat :

$$X_{i+1} = X_i - \lambda \cdot \nabla_{X_i} \tag{A.3}$$

Le paramètre λ est sélectionné de sorte que $f(X_{i+1}) < f(X_i)$. Si un tel λ n'existe pas alors un minimum est atteint. Ceci est réalisé par recherche linéaire : λ est initialisé une valeur fixée et itérativement minimisé, par exemple $\lambda \leftarrow \frac{1}{2} \cdot \lambda$, jusqu'à ce que $f(X_{i+1}) < f(X_i)$ ou $\lambda < \epsilon$.

La méthode de descente de gradient est connue pour souffrir de longs temps de convergence, en particulier près du minimum, [Triggs et al., 1999]. Une solution apportée à ce problème porte le nom de *gradient conjugué*. Dans cette approche les déplacements successifs sont orthogonaux.

A.2 Newton

La méthode dite de Newton est plus efficace mais nécessite que $f()$ soit deux fois différentiable. On reprend l'équation A.2. On suppose toujours que l'on est à proximité du minimum recherché, alors on cherche l'incrément δ annulant l'équation. On dérive puis annule A.2 par rapport à δ, on obtient :

$$\Rightarrow \begin{array}{l} f(X+\delta) \simeq f(X) + J_X \cdot \delta + \frac{1}{2} \cdot \delta^\top \cdot H_X \cdot \delta \\ \\ \frac{\partial f(X+\delta)}{\partial \delta} \simeq J_X + H_X \cdot \delta \end{array} \tag{A.4}$$

et

$$\Rightarrow \begin{array}{l} \frac{\partial f(X+\delta)}{\partial \delta} = 0 \\ \\ H_X \cdot \delta = -J_X \end{array} \tag{A.5}$$

A.3 Gauss-Newton

En grandes dimensions il est souvent difficile de calculer la matrice hessienne. On introduit la fonction $\Delta()$ en réécrivant l'équation A.1 sous la forme :

$$f(X) = \frac{1}{2} \cdot \Delta(X)^\top \cdot \Delta(X) \tag{A.6}$$

On suppose que $\Delta()$ est au moins une fois différentiable, on a donc au voisinage de X :

$$\begin{array}{rl} f(X+\delta) & \simeq \frac{1}{2} \cdot (\Delta(X) + J_{\Delta X})^\top \cdot (\Delta(X) + J_{\Delta X}) \\ & \simeq \frac{1}{2} \cdot \Delta(X)^\top \cdot \Delta(X) + J_{\Delta X} \cdot \Delta(X) \cdot \delta^\top + \frac{1}{2} \cdot J_{\Delta X}^\top \cdot J_{\Delta X} \cdot \Delta(X) \cdot \delta \end{array} \tag{A.7}$$

Comme précédemment on cherche δ annulant $f(X)$:

$$\frac{\partial f(X+\delta)}{\partial \delta} \simeq J_{\Delta X} \cdot \Delta(X) + J_{\Delta X}^\top \cdot J_{\Delta X} \cdot \delta = 0 \tag{A.8}$$

La solution δ est ainsi la solution de l'*équation normale* :

$$J_{\Delta X}^\top \cdot J_{\Delta X} \cdot \delta = -J_{\Delta X} \cdot \Delta(X) \tag{A.9}$$

δ peut être obtenu comme solution de l'équation au sens des moindres carrés linéaires. La méthode *Gauss-Newton* est intéressante car près du minimum elle converge quadratiquement vers la solution optimale. De plus, contrairement à la méthode de descente de gradient il n'est pas nécessaire d'ajouter une étape de détermination de l'amplitude du pas.

A.4 Levenberg-Marquardt

La méthode d'itération de Levenberg-Marquardt est une hybridation des méthodes de descente de gradient et de Gauss-Newton. La première ayant un rayon d'efficacité plus important que la seconde mais convergeant lentement

près de l'optimum, où justement la méthode de Gauss-Newton converge rapidement. Une itération de Levenberg-Marquardt se calcule en recherchant la solution de l'*équation normale augmentée* suivante :

$$\left(J_{\Delta X}^{\top} \cdot J_{\Delta X} + \lambda \cdot Id\right) \cdot \delta = -J_{\Delta X} \cdot \Delta(X) \tag{A.10}$$

Usuellement la méthode de Gauss-Newton est privilégiée en fixant λ petit par rapport aux éléments de $J_{\Delta X}^{\top} \cdot J_{\Delta X}$. A chaque étape l'équation est résolue et si l'erreur est diminuée, la valeur de l'incrément est acceptée, la valeur de λ diminuée en prévision de la prochaine itération. Si par contre l'erreur est augmentée λ est augmenté afin de favoriser la descente de gradient et l'équation résolue jusqu'à ce que l'incrément réduise effectivement les erreurs. L'effet de ce processus est de renforcer l'apport de la méthode de Gauss-Newton près de l'optimum tout en permettant de rebasculer vers une approche de descente de gradient lorsque l'estimation courante des paramètres est trop éloignée.

A.5 Ajustement de faisceaux éparse

La complexité de la résolution de l'équation normale augmentée est en $O(N^3)$, une implémentation directe de l'itération de Levenberg-Marquardt n'est viable que pour un faible nombre de paramètres. Lorsque le nombre de paramètres augmente plusieurs sophitiscations nécessite d'être apportées.

La première consiste à distinguer les paramètres correspondant aux caméras des paramètres correspondant aux points. Cela permet, par l'utilisation du complément de Schur de calculer en deux temps l'incrément à apporter aux paramètres des caméras puis celui à apporter aux paramètres des points. La seconde amélioration consiste à observer que la matrice $J_{\Delta X}^{\top} \cdot J_{\Delta X}$ est souvent très creuse, particulièrement pour des trajectoires exploratoires de la caméra. En effet, chaque point de la carte n'est observé que par une faible proportion des caméras. Cette observation est utilisée pour simplifier le calcul du complément de Schur, [Hartley and Zisserman, 2004] [C. Engels, 2006].

A.6 Mise en pratique

A.6.1 Paramétrisations locales des rotations

Que l'on choisisse de représenter les rotations sous forme matricielle ou de quaternion, la transformation possèdant 3 degrés de libertés est sur-paramétrée. Cela signifie que si l'on cherche à faire varier les paramètres dans l'espace de représentation, $\mathbb{R}^9$ ou $\mathbb{R}^4$, il est probable que les paramètres optimaux calculés brisent les conditions requises pour que la matrice ou le quaternion employé représente une rotation.

Pour éviter ce problème, à chaque itération un plan tangent de dimension 3 est évalué au point défini dans l'espace de représentation. L'incrément à apporter à la rotation est alors estimé dans l'espace du plan.

A.6.2 Bibliothèques logicielles

La mise en pratique de méthodes d'ajustement de faisceaux, et plus généralement de moindres carrés non linéaires, éparses n'étant pas triviale, plusieurs bibliothèques logicielles ont été développées et rendues publiques par la communauté. Nous en avons utilisé deux dans le cadre de la thèse.

Pour nos implémentations, en *matlab* puis en C++, de notre *SLAM* monoculaire nous avons utilisé la bibliothèque *SSBA* [Zach,] de Zach. Cette bibliothèque utilise les quaternions pour représenter les rotations et pour réduire l'influence des *outliers* la fonction de coût robuste de *Huber*.

Pour notre implémentation de *SLAM* monoculaire et profondeur, nous avons utilisé la bibliothèque logicielle *ceres* [Agarwal et al.,]. Cette bibliothèque nous a particulièrement intéressés car elle permet de définir des fonctions de coût génériques et présente en plus l'avantage d'offrir un mécanisme de calcul automatique numérique ou symbolique des matrices jacobiennes. Les auteurs préconisent de privilégier la méthode symbolique, la numérique pouvant être sujette à des instabilités. Nous avons représenté les rotations sous forme de quaternions et avons utilisé la fonction de coût de *Huber* pour robustifier le processus.

Bien que nous ne l'ayons pas utilisée une dernière bibliothèque nous semble intéressante. Présentée dans [Kuemmerle et al., 2011] cette bibliothèque est basée sur un processus d'optimisation de graphe. Comme la bibliothèque *ceres* elle est générique en ce qu'il permet de spécifier des fonctions de coûts non prédéfinies. Cependant les matrices jacobiennes impliquées dans la mini-

misation de la fonction coût peuvent être estimées numériquement ou doivent être fournies par l'utilisateur.

Annexe B

Outils mathématiques

B.1 Quaternions

Découverts par W.R. Hamilton les quaternions permettent, à la manière des nombres complexes dans le plan, de représenter commodément les rotations dans l'espace. Un quaternion Q est composé d'une partie *réelle*, a, et d'une partie *imaginaire* un vecteur $\overrightarrow{V}$ appartenant à $\mathbb{R}^3$. Un quaternion est usuellement noté : $q = a + \overrightarrow{V}$. La somme de deux quaternions est définie comme :

$$\begin{aligned} Q + Q' = & \quad (a + \overrightarrow{V}) + (a' + \overrightarrow{V'}) \\ = & \quad (a + a') + (\overrightarrow{V} + \overrightarrow{V'}) \end{aligned} \tag{B.1}$$

La multiplication par un scalaire comme :

$$\lambda \cdot Q = \lambda \cdot a + \lambda \cdot \overrightarrow{V} \tag{B.2}$$

Ces deux opérations confèrent à l'ensemble des quaternions la structure d'un espace vectoriel. La multiplication de deux quaternions est définie comme :

$$\begin{aligned} Q * *Q' = & \quad (a + \overrightarrow{V}) * *(a' + \overrightarrow{V'}) \\ = & \quad (a \cdot a' - \overrightarrow{V} \cdot \overrightarrow{V'}) + (a \cdot \overrightarrow{V'} + a' \cdot \overrightarrow{V} + \overrightarrow{V} \times \overrightarrow{V'}) \end{aligned} \tag{B.3}$$

On définit également le conjugué d'un quaternion comme :

$$\overline{Q} = a - \overrightarrow{V} \tag{B.4}$$

La propriété qui nous intéresse particulièrement est la suivante : Soit α un vecteur de $\mathbb{R}^3$, il est possible de lui imprimer une rotation R d'angle θ autour d'un vecteur unitaire u grâce au quaternion

$$Q = \cos\frac{\theta}{2} + \sin\frac{\theta}{2} \cdot u \tag{B.5}$$

par la formule :

$$R\alpha = Q \cdot (0 + \alpha) \cdot \overline{Q} \tag{B.6}$$

Ce qui implique qu'un quaternion représentant une rotation doit être de norme unitaire. Finalement, une composition de rotations R_1R_2 peut être représentée par le quaternion $Q_1 * *Q_2$, Q_1 et Q_2 étant les quaternions associés aux matrices R_1 et R_2. Notons également que les quaternions Q et $-Q$ représentent la même rotation.

B.2 La décomposition en valeurs singulieres

La décomposition en valeurs singulières est une décomposition particulièrement utile et régulièrement employée dans ce manuscrit.
Soit une matrice carrée A, la *SVD*, de l'anglais *Singular Value Decomposition*, est la factorisation de A sous la forme :

$$A = U \cdot D \cdot V^\top \tag{B.7}$$

telle que U et V sont des matrices orthogonales et D une matrice diagonale dont les éléments sont positifs. La décomposition en valeurs singulières est également définie pour des matrices non carrées. Dans le cas d'une matrice A de dimension $m \times n$ avec $m > n$, U est rectangulaire de dimension $m \times n$ et ses colonnes sont orthogonales, D est de dimension $n \times n$ et V de dimension $n \times n$ est orthogonale. La décomposition est également définie dans le cas $m < n$ mais ce cas ne nous intéressera pas par la suite.

Selon [Hartley and Zisserman, 2004], le nombre d'opérations à la détermination des matrices U, V et D est $4 \cdot m^2 \cdot n + 8 \cdot m \cdot n^2 + 9 \cdot n^3$. Si l'on ne recherche que V et D alors le nombre d'opérations se réduit à $4 \cdot m^2 \cdot n + 8 \cdot n^3$.
Les coefficients de D sont usuellement ordonnés de manière décroissante. Par la suite nous considèrerons toujours que la décomposition est ainsi réalisée. Dans ce cas, la dernière colonne de la matrice V correspond à la plus petite valeur singulière de A. Cette propriété sera généralement utilisée pour

déterminer une solution aux équations de la forme :

$$A \cdot x = 0, \quad ||x|| = 1 \tag{B.8}$$

B.3 Matrices pseudo-inverses

Nous supposons que les applications s'appliquent sur le corps des réels. L'adjointe d'une matrice est dans ce cas équivalente à sa transposée. Il est parfois nécessaire de chercher à inverser des matrices qui ne sont pas inversibles. Pour cela, il est possible d'utiliser des formes pseudo-inverses. On note f l'application linéaire correspondant à une matrice A. Par définition si f est inversible la relation suivante est vérifiée :

$$\forall x \quad f \circ f^{-1}(x) = f^{-1} \circ f(x) = x \tag{B.9}$$

Cette relation doit être relachée dans le cas des pseudo-inverses et la pseudo-inverse n'est pas unique. Notons g une pseudo-inverse de f, elle doit vérifier les relations suivantes :

$$\begin{aligned} \forall x \quad & f \circ g \circ f(x) = f(x) \\ \forall x \quad & g \circ f \circ g(x) = g(x) \end{aligned} \tag{B.10}$$

La manière la plus générale d'obtenir la pseudo-inverse d'une matrice est d'utiliser une décomposition en valeurs singulières. Si l'on note :

$$A = U \cdot D \cdot V^\top \tag{B.11}$$

alors une pseudo-inverse de A est :

$$A^p = V \cdot D^p \cdot U^\top \tag{B.12}$$

Avec

$$D^p_{ii} = \begin{cases} {D_{ii}}^{-1} & \text{si } D_{ii} \neq 0 \\ 0 & \text{sinon} \end{cases} \tag{B.13}$$

Cependant une décomposition en valeurs singulières peut être coûteuse en ressources de calcul. Dans certains cas particuliers il est possible d'obtenir une pseudo-inverse plus directement. Si le rang de la matrice est égal à son nombre de lignes on a :

$$A^p = A^\top \cdot (A \cdot A^\top)^{-1} \tag{B.14}$$

Si le rang de A est égal à son nombre de colonnes :

$$A^p = (A^\top \cdot A)^{-1} \cdot A^\top \tag{B.15}$$

B.4 Matrice compagnon

Il est souvent nécessaire de déterminer les racines d'un polynôme, sans perte de généralité nous l'assimilons à son équivalent unitaire. Une façon simple de déterminer ses racines est d'avoir recours à la marice compagnon du polynôme. Soit le polynôme $p(X)$:

$$p(X) = X^n + c_{n-1} \cdot X^{n-1} + \cdots + c_1 \cdot X + c_0 \tag{B.16}$$

La matrice compagnon de $p(X)$ est :

$$M_{n\times n} = \begin{pmatrix} -c_{n-1} & -c_{n-2} & \cdots & -c_1 & -c_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix} \tag{B.17}$$

La matrice compagnon ainsi définie, son polynôme caractéristique est égal à $p(X)$. Les valeurs propres de M sont alors les racines du polynôme.

Annexe C

Homographie

C.1 La relation d'homographie

Soit un point P de $\mathbb{R}^3$, deux poses de caméras C^1 et C^2, les coordonnées de P dans ces bases P^1 et P^2 appartenant à $\mathbb{R}^3$ et les observations correspondantes en coordonnées normalisées p^1 et p^2 appartenant à $P_2(\mathbb{R})$.

Une homographie est une transformation projective entre deux plans. On suppose que P appartient au plan π de coordonnées $\left(a, b, c, d\right)^\top = \left(\overrightarrow{\mathbf{n}}^\top, d\right)^\top$ dans C^1. Le vecteur $\overrightarrow{\mathbf{n}}$ est la normale au plan et le scalaire d la distance orthogonale du plan à l'origine du repère, ici le centre optique de C^1. On suppose que P appartient au plan π, on vérifie donc :

$$\begin{aligned} & \overrightarrow{\mathbf{n}}^\top \cdot P^1 + d = 0 \\ & \qquad\qquad \Leftrightarrow \\ & \frac{-1}{d} \cdot \overrightarrow{\mathbf{n}}^\top \cdot P^1 = 1 \end{aligned} \tag{C.1}$$

La transformation de P^1 en P^2 étant gouvernée par la matrice de rotation R_1^2 et le vecteur de translation T_1^2, on vérifie :

$$\begin{aligned} & P^2 = R_1^2 \cdot P^1 + T_1^2 \\ \Leftrightarrow \\ & P^2 = R_1^2 \cdot P^1 + T_1^2 \cdot \left(\frac{-1}{d} \cdot \overrightarrow{\mathbf{n}}^\top \cdot P^1 \right) \\ \Leftrightarrow \\ & P^2 = H_1^2 \cdot P^1 \end{aligned} \tag{C.2}$$

Où :

$$H_1^2 = R_1^2 - \frac{1}{d} \cdot \overrightarrow{\mathbf{n}}^\top \cdot T_1^2 \tag{C.3}$$

On a de plus :

$$\begin{array}{l} P^1 = \alpha_1 \cdot p^1 \\ P^2 = \alpha_2 \cdot p^2 \end{array} \quad \Rightarrow \quad p^2 = \alpha \cdot H_1^2 \cdot p^1 \tag{C.4}$$

La matrice H_1^2 représente ainsi la transformation des projections, p^1 et p^2, d'un point appartenant à un plan selon le changement de pose de la caméra. On nomme H_1^2 la matrice d'homographie. Notons bien qu'elle est définie au facteur d'échelle près comme le représente le scalaire α dans la relation C.4.

Avec la méthode de [Faugeras and Lustman, 1988] il est possible de calculer 4 poses relatives de caméra à partir d'une matrice homographie.

C.1.1 Estimation d'une matrice d'homographie

Une matrice d'homographie, H, comporte neuf éléments mais est définie au facteur d'échelle près, ce qui signifie qu'elle possède en réalité huit degrés de libertés. Un appariement de points $p_1 \leftrightarrow p_2$ fournit deux contraintes, ainsi quatre appariements suffisent à déterminer les huits degrés de libertés.

Une manière d'estimer une matrice d'homographie est la méthode *DLT*, *Direct Linear Transformation*, on reprend l'explication de [Hartley and Zisserman, 2004]. Puisque $H \cdot p_1 = p_2$, on sait que $p_1' \times p_2 = 0$. Soient h_1, h_2 et h_3 les lignes de H et $\begin{pmatrix} x_i & y_i & 1 \end{pmatrix}^\top$ les coordonnées de p_i, on a :

$$H \cdot p_1 = \begin{pmatrix} h_1 \cdot p_1 \\ h_2 \cdot p_2 \\ h_3 \cdot p_3 \end{pmatrix}$$

$$\Rightarrow$$

$$p_2 \times p_1 = \begin{pmatrix} y_2 \cdot h_3 \cdot p_1 - h_2 \cdot p_1 \\ h_1 \cdot p_1 - x_2 \cdot h_3 \cdot p_1 \\ x_2 \cdot h_2 \cdot p_1 - y_2 \cdot h_1 \cdot p_1 \end{pmatrix} \tag{C.5}$$

$$\Rightarrow$$

$$\begin{pmatrix} \mathbf{0} & -p_1^\top & y_2 \cdot p_1^\top \\ p_1^\top & \mathbf{0} & -x_2 \cdot p_1^\top \\ -y_2 \cdot p_1^\top & x_2 \cdot p_1^\top & \mathbf{0} \end{pmatrix} \cdot \begin{pmatrix} h_1^\top \\ h_2^\top \\ h_3^\top \end{pmatrix} = \mathbf{0}$$

La dernière matrice étant antisymétrique elle est de rang deux, ainsi un appariement fournit bien deux contraintes. En utilisant quatre appariements on obtient huit contraintes que l'on empile dans une matrice A. Les coefficients de H faisant partie du noyau de A ils sont calculés en effectuant une décomposition en valeurs singulières de A.

Annexe D

Robustification

D.1 M-estimateurs

Soit e_i l'erreur résiduelle associée à une mesure selon un modèle de transformation. Si l'on considère les mesures seulement perturbées par un bruit gaussien centré en 0 alors la meilleure transformation est celle minimisant la somme des erreurs quadratiques sur l'ensemble des paramètres, soit :

$$\sum_i e_i^2 \tag{D.1}$$

Si par contre les paramètres ne sont plus seulement affectés par des erreurs de mesures mais contiennent également des données aberrantes, une solution au sens des moindres carrés n'est pas adaptée à la sélection de la meilleure transformation. En effet, les données aberrantes peuvent avoir un poids très important et ainsi distordre l'évaluation de la transformation. Une solution est alors d'employer un M-estimateur, c'est à dire d'appliquer une fonction de coût robuste aux erreurs résiduelles et de trouver la transformation minimisant la somme de ces valeurs, soit :

$$min \sum_i \rho(e_i) \Leftrightarrow min \sum_i w_i(e_i) \cdot e_i^2 \tag{D.2}$$

Où ρ est une fonction de coût robuste définie positive, s'annulant en une valeur unique et dont la croissance est inférieure à une croissance quadratique. Il est équivalent de pondérer l'erreur selon sa valeur observée, comme présenté dans D.2. Des fonctions de poids usuelles sont celles de *Tukey* ou *Huber*,

présentées ci dessous :

$$\text{Tukey} : \begin{cases} |e_i| \leq \sigma \\ \text{sinon} \end{cases} \quad w_i = \begin{cases} (1 - (\frac{e_i}{\sigma})^2)^2 \\ 0 \end{cases} \quad \rho_i = \begin{cases} \frac{\sigma^2}{6} \cdot (1 - (1 - (\frac{e_i}{\sigma})^2)^3) \\ \frac{\sigma^2}{6} \end{cases}$$

$$\text{Huber} : \begin{cases} |e_i| \leq \sigma \\ \text{sinon} \end{cases} \quad w_i = \begin{cases} 1 \\ \sigma/|e_i| \end{cases} \quad \rho_i = \begin{cases} \frac{e_i^2}{2} \\ \sigma \cdot (|e_i| - \frac{\sigma}{2}) \end{cases} \tag{D.3}$$

Ces fonction de poids reposent sur l'estimation simultannée de l'écart type σ des erreurs résiduelles. Selon [Rousseeuw and Leroy, 1987] et [Zhang et al., 1995], une estimation robuste de l'écart type est liée à la médiane des erreurs et donnée par la formule suivante :

$$\sigma = 1.4826 \cdot (1 + 5/(n - p)) \cdot \text{median}_i |e_i| \tag{D.4}$$

Où n représente le cardinal de l'ensemble des paramètres et p le nombre de paramètres constituant la transformation estimée.

Ces deux fonctions remplissant un même office sont néanmoins de natures légèrement différentes et ainsi en est il des avantages qu'elles offrent. La fonction de coût de *Tukey* est intéressante en ce qu'elle permet de considérer qu'au delà d'un seuil d'erreur prédéterminé l'erreur correspond à la mesure d'un *outlier*. La distribution de ceux-ci étant a priori uniforme il n'y a pas de raison de leur donner un poids croissant au delà d'un certain point. La fonction de *Huber* n'offre pas cette possibilité mais possède l'avantage d'être convexe. Dans [Hartley and Zisserman, 2004] Hartley et Zisserman démontrent que les fonctions de coût non convexes sont susceptibles de présenter des minima locaux là ou des fonctions convexes n'en présentent pas. Ces propriétés rendent ces deux types de fonctions plus spécifiquement adaptées à des cadres différents. Si l'on cherche à optimiser les paramètres d'une transformation dans le cadre d'une minimisation d'erreur aux sens des moindres carrés non linéaires, la présence de minima locaux est à éviter au maximum, il est préférable d'opter pour la fonction de *Huber*. Si par contre on cherche à estimer la qualité ou la vraissemblance, dans un filtre particulaire par exemple, d'une transformation, la présence de minima locaux a très peu de chances d'influer et il est plus intéressant de pondérer au mieux les observations, ainsi la fonction de *Tukey* est plus adaptée.

D.2 LMedS

Présentée dans [Rousseeuw and Leroy, 1987] et accessible dans [Zhang et al., 1995], la méthode *LMedS*, *Least Median of Squares*, résoud le problème de l'estimation de la transformation en sélectionnant non pas l'hypothèse avec le plus grand support mais celle dont la médiane des erreurs quadratiques est la plus faible. L'avantage de cette méthode est de ne reposer ni sur une estimation *a priori* de l'écart type des erreurs de mesure ni sur un seuil d'erreur acceptable *a priori*. L'inconvénient est que la méthode nécessite une proportion minimale de 50% d'*inliers* pour donner un résultat correct. Le schéma est similaire à celui du *RANSAC*, les différences se trouvent :

- sur le critère de qualité d'une transformation : $\min(\text{médiane}_i(e_i^2))$
- la mise à jour pour chaque transformation de l'écart type selon 2.31
- la classification des *inliers* et *outliers* que l'on peut représenter à l'aide de la fonction de poids suivante :

$$w_i = \begin{cases} 1 & \text{si } e_i^2 < (2.5 \cdot \sigma)^2 \\ 0 & \text{sinon} \end{cases} \tag{D.5}$$

Une fois le nombre d'essais prédit et mis à jour selon 2.26 la solution peut être raffinée en minimisant la fonction de coût suivante :

$$min \sum_i w_i \cdot e_i^2 \tag{D.6}$$

D.3 MSAC et MLESAC

Dans [Torr and Zisserman, 2000], Torr et Zisserman observent que la transformation solution obtenue par *RANSAC* correspond à celle minimisant la fonction de coût :

$$C = \sum_i \rho(e_i)^2 \tag{D.7}$$

où :

$$\rho(e^2) = \begin{cases} 0 & e^2 < t^2 \\ \text{constante} & e^2 \geq t^2 \end{cases} \tag{D.8}$$

Cette fonction de coût implique que le coût des *inliers* est nul et celui des *outliers* constant. Ainsi plus t est élevé plus il existe de solutions de coût identique ce qui induit des estimations de mauvaise qualité. Dans

[Torr and Zisserman, 2000], Torr et Zisserman montrent que cet effet peut être corrigé sans complexité algorithmique additionelle en remplaçant cette fonction de coût par la suivante :

$$C_2 = \sum_i \rho_2(e_i)^2 \tag{D.9}$$

où :

$$\rho_2(e^2) = \begin{cases} e^2 & e^2 < t^2 \\ T^2 & e^2 \geq t^2 \end{cases} \tag{D.10}$$

Ce qui correspond à un M-estimateur paramétré avec une fonction de coût robuste de type *Blake-Zisserman*, voir [Hartley and Zisserman, 2004]. Cette fonction de coût permet de prendre en compte l'adéquation des *inliers* aux données, contrairement à la fonction de coût de *RANSAC*. Les auteurs constatent selon les situations des qualités d'estimations légèrement ou grandement améliorées. Cette méthode porte le nom de *MSAC*.

Dans le même article les auteurs proposent également une méthode encore plus robuste, la méthode *MLESAC*. Afin de prendre en compte les erreurs de mesures et les données aberrantes, les auteurs modélisent la probabilité d'une erreur comme une mixture de distribution gaussienne et uniforme :

$$P(e) = \left(\gamma \cdot \frac{1}{(2 \cdot \pi \cdot \sigma^2)^2} \cdot e^{-\frac{e^2}{2 \cdot \sigma^2}} + (1 - \gamma) \cdot \frac{1}{v} \right) \tag{D.11}$$

La constante v représentant la taille de l'espace possible d'erreur, pour des points appariés cela correspond à la taille de recherche de l'appariement. Cette probabilité étant définie, les auteurs proposent de prendre pour solution la transformation réalisant le maximum de vraisemblance des observations selon D.11. Cependant le paramètre de mélange γ étant inconnu il doit être estimé pour chaque transformation évaluée. Cela est fait par l'usage d'un algorithme *espérance-maximisation* qui est un processus itératif, potentiellement significativement plus consommateur de ressources que les précédents. L'estimation de γ permet simultanément de classifier les *inliers* et *outliers*. Le processus d'estimation de γ et de classification est consultable dans [Torr and Zisserman, 2000].

D.4 Ransac préemptif

Dans [Nistér, 2005], Nister propose un schéma dit pré-emptif pour la validation d'hypothèses de transformation. Ce schéma s'inscrit au sein du

RANSAC et sert à accélérer la sélection de la meilleure transformation parmi un ensemble d'hypothèses. Il procède itérativement et repose sur l'utilisation d'une fonction $f : \mathbf{N} \rightarrow \mathbf{N}$ décroissante, mais pas strictement, servant à déterminer la quantité des meilleures hypothèses à garder depuis l'étape précédente. L'évaluation d'une hypothèse est réalisée de manière incrémentale selon l'examen des données. A l'initialisation du schéma les données sont permutées aléatoirement afin d'éviter des configurations dégénérées ou faiblement conditionnées. A chaque étape i les hypothèses restantes sont mises à jour avec la i-ème donnée. La procédure s'arrête lorsque toutes les données ont été traitées ou bien lorsque à l'étape i $f(i) = 1$.

Les expérimentations montrent qu'avec une fonction f suffisamment bien paramétrée la qualité des résultats est comparable à celle d'un algorithme *RANSAC* classique avec une vitesse de traitement sensiblement accrue.

D.5 Distribution spatiale

Dans [Hartley and Zisserman, 2004], Hartley et Zisserman analysent l'effet de la distribution spatiale des données pour le calcul d'une matrice d'homographie selon un algorithme *DLT*, voir la section C.1.1. Ils constatent que différentes distributions peuvent conduire à des estimations de transformation différentes. Par ailleurs, une sélection localement concentrée de données est plus susceptible d'être une instance de configuration dégénérée pour l'estimation de la transformation et l'influence du bruit de mesure est également plus importante sur l'estimation de la transformation.

D.5.1 Normalisation des transformations

Présentée dans [Hartley and Zisserman, 2004] une solution est d'appliquer un changement de base sur les données, d'estimer la transformation sur ces données replacées et enfin de changer la base de la transformation afin d'obtenir la véritable transformation M.
Les auteurs préconisent de translater les données de sorte qu'elles soient centrées en $\mathbf{0}$, puis de choisir un facteur d'échelle isotropique à leur appliquer de sorte que les points se trouvent en moyenne à une distance de $\sqrt{n}$ du centre, n étant la dimension de l'espace dans lequel les données sont exprimées. Le changement de base peut donc s'exprimer sous la forme d'une matrice T composée de la translation et du changement d'échelle et nécessite

d'être appliqué sur les données exprimées en coordonnées homogènes. La transformation $\tilde{M}$ peut être alors estimée, puis nécessite d'être replacée dans les bases d'origine des données. Si les données sont des singletons, M s'exprime comme $M = T^{-1} \cdot \tilde{M} \cdot T$, si les données sont des paires, alors chaque partie subit un changement de base, T et T', alors M s'exprime comme $M = T'^{-1} \cdot \tilde{M} \cdot T$.
Cette méthode permet d'obtenir des résultats plus fiables et, c'est très important, invariants à la distribution spatiale des points.

D.5.2 Sélection uniforme des données

Une autre solution est de définir une grille uniforme dans l'espace des données et d'affecter chacune de ces dernières à la case à laquelle elle appartient. Alors la sélection des n données pour former l'échantillon minimal pour générer une hypothèse de transformation se fait non plus en sélectionnant les données aléatoirement, mais en sélectionnant aléatoirement n cases de la grilles puis un échantillon par case. Cette méthode est connue sous le nom de méthode de *bucket* et a été introduite dans [Zhang et al., 1995]. Cette méthode permet de limiter l'influence du bruit de mesure et de limiter le risque de sélectionner des échantillons présentant une configuration dégénérée.

Annexe E

Triangulation

E.1 Triangulation par moindres carres

La projection du point dans les images des caméras se note formellement de la manière suivante :

$$\begin{cases} x_1 = \frac{P_{11} \cdot X + P_{12} \cdot X_2 + P_{13} \cdot X_3 + P_{14}}{P_{31} \cdot X + P_{32} \cdot X_2 + P_{33} \cdot X_3 + P_{34}} \\ x_2 = \frac{P_{21} \cdot X + P_{22} \cdot X_2 + P_{23} \cdot X_3 + P_{24}}{P_{31} \cdot X + P_{32} \cdot X_2 + P_{33} \cdot X_3 + P_{34}} \\ x'_1 = \frac{P'_{11} \cdot X + P'_{12} \cdot X_2 + P'_{13} \cdot X_3 + P'_{14}}{P'_{31} \cdot X + P'_{32} \cdot X_2 + P'_{33} \cdot X_3 + P'_{34}} \\ x'_2 = \frac{P'_{21} \cdot X + P'_{22} \cdot X_2 + P'_{23} \cdot X_3 + P'_{24}}{P'_{31} \cdot X + P'_{32} \cdot X_2 + P'_{33} \cdot X_3 + P'_{34}} \end{cases} \tag{E.1}$$

Ce système peut être réécrit sous une forme $A \cdot X = b$:

$$\begin{pmatrix} P_{11} - x_1 \cdot P_{31} & P_{12} - x_1 \cdot P_{32} & P_{13} - x_1 \cdot P_{33} \\ P_{21} - x_2 \cdot P_{31} & P_{22} - x_2 \cdot P_{32} & P_{23} - x_2 \cdot P_{33} \\ P'_{11} - x'_1 \cdot P'_{31} & P'_{12} - x'_1 \cdot P'_{32} & P'_{13} - x'_1 \cdot P'_{33} \\ P'_{21} - x'_2 \cdot P'_{31} & P'_{22} - x'_2 \cdot P'_{32} & P'_{23} - x'_2 \cdot P'_{33} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} x_1 \cdot P_{34} - P_{14} \\ x_2 \cdot P_{34} - P_{24} \\ x'_1 \cdot P'_{34} - P'_{14} \\ x'_2 \cdot P'_{34} - P'_{24} \end{pmatrix} \tag{E.2}$$

Cette forme met en évidence le caractère pseudo inversible du problème. Il est résolu par l'usage de la pseudo inverse de la matrice A et cela revient à calculer la solution au sens des moindres carrés.

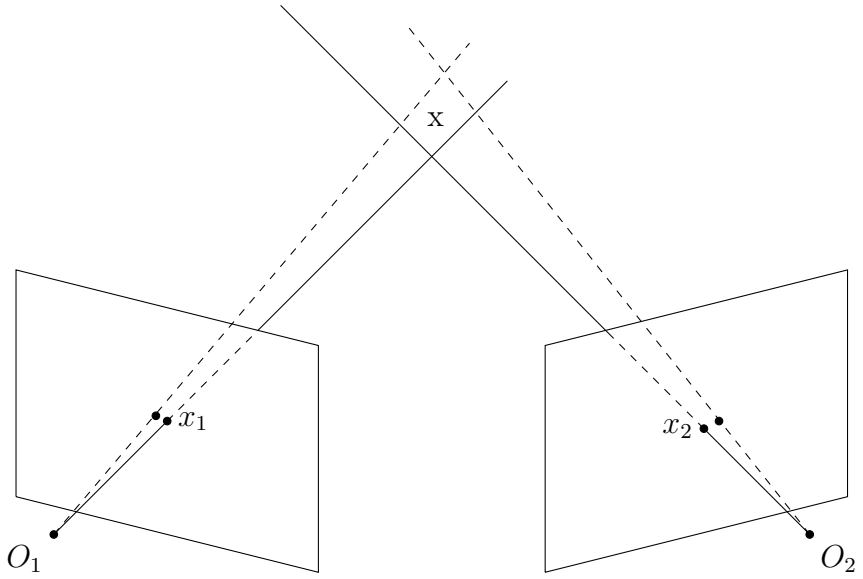


FIGURE E.1 – Illustration d'une triangulation de point à partir de deux vues et de l'influence d'erreurs de mesure sur la reconstruction.

E.2 Triangulation par DLT

Les équations de projection du point dans l'image peuvent également se réécrire sous la forme

$$A \cdot X = 0 \tag{E.3}$$

Pour cela on part des relations suivantes :

$$\begin{cases} x = P \cdot X & \Rightarrow x \times (P \cdot X) = 0 \\ x' = P' \cdot X & \Rightarrow x' \times (P' \cdot X) = 0 \end{cases} \tag{E.4}$$

Notons P^i la i-ème ligne de P, le produit vectoriel pour la première observation se développe comme suit :

$$\begin{array}{rcrl} x_2 \cdot (P^3 \cdot X) & - & P^2 \cdot X & = 0 \\ P^1 \cdot X & - & x_1 \cdot (P^3 \cdot X) & = 0 \\ x_1 \cdot (P^2 \cdot X) & - & x_2 \cdot (P^1 \cdot X) & = 0 \end{array} \tag{E.5}$$

Ainsi on obtient trois équations pour chaque observation. Le point X appartenant à $\mathbf{P}^4$ quatre suffisent, on n'en retient alors que deux pour chaque

caméra. On forme A comme suit :

$$A = \begin{pmatrix} P^2 \cdot X - x_2 \cdot (P^3 \cdot X) \\ P^1 \cdot X - x_1 \cdot (P^3 \cdot X) \\ P'^2 \cdot X - x'_2 \cdot (P'^3 \cdot X) \\ P'^1 \cdot X - x'_1 \cdot (P'^3 \cdot X) \end{pmatrix} \tag{E.6}$$

La position du point est trouvée en résolvant le système E.3 par *DLT*.

E.3 Triangulation optimale

Présentée dans [Hartley and Sturm, 1997] et [Hartley and Zisserman, 2004] cette méthode est en réalité un pré-traitement des observations visant à résoudre le problème de non intersection des rayons projectifs de manière optimale sous l'hypothèse que le bruit des observations est gaussien.

La non-intersection des rayons provient de ce que les observations des points ne respectent pas la contrainte épipolaire. Cette méthode optimale repose alors sur la connaissance de la matrice essentielle E (ou fondamentale) et s'attache à déterminer les points $\hat{x}$ et $\hat{x'}$ tels que :

$$(\hat{x}, \hat{x'}) = argmin \left(d(x, \hat{x})^2 + d(x', \hat{x'})^2 \right) \tag{E.7}$$

Sous la contrainte

$$\hat{x'}^\top \cdot E \cdot \hat{x} = 0$$

On note à présent l_e et l'_e les lignes épipolaires correspondant aux pointx x et x' dans les images de C et C' respectivement. Les points $\hat{x}$ et $\hat{x}'$ doivent donc se trouver sur l et l'. Les points de ces lignes satisfaisant E.7 sont les projetés orthogonaux de x et x'. Puisque les deux observations sont bruitées il n'est pas juste de déterminer les lignes épipolaires en fonction de ceux-ci. Il est donc nécessaire de déterminer les lignes épipolaires optimales et l'erreur à minimiser devient alors :

$$d(x, l)^2 + d(x', l')^2 \tag{E.8}$$

La stratégie repose d'abord sur un changement de base dans C et C' permettant de modifier les coordonnées des observations et des épipoles afin de simplifier la résolution du problème. A partir de ces points modifiés une expression simple de la première ligne épipolaire est paramétrée par un paramètre t. Connaissant E on déduit la seconde ligne épipolaire. Alors E.8 est

exprimée comme polynôme de degré 6 en t et la valeur optimale du paramètre est une des racines du polynômes. Il suffit de sélectionner celle minimisant E.8.

Bibliographie

[Agarwal et al.,] Agarwal, S., Mierle, K., and Others. Ceres solver. `https://code.google.com/p/ceres-solver/`.

[Agarwal et al., 2009] Agarwal, S., Snavely, N., Simon, I., Seitz, S. M., and Szeliski, R. (2009). Building rome in a day. In *ICCV*, pages 72–79. IEEE.

[Agrawal et al., 2008] Agrawal, M., Konolige, K., and Blas, M. R. (2008). Censure : Center surround extremas for realtime feature detection and matching. In *Computer Vision–ECCV 2008*, pages 102–115. Springer.

[A.J. Davison and Stasse, 2007] A.J. Davison, I. D. Reid, N. D. M. and Stasse, O. (2007). Monoslam : Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Alahi et al., 2012] Alahi, A., Ortiz, R., and Vandergheynst, P. (2012). Freak : Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517. IEEE.

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

[Boucher et al., 2013] Boucher, M., Ababsa, F.-E., and Mallem, M. (2013). Reducing the slam drift error propagation using sparse but accurate 3d models for augmented reality applications. In *Proceedings of the Virtual Reality International Conference : Laval Virtual*, page 11. ACM.

[Boucher et al., 2012] Boucher, M., Ababsa, F.-E., Mallem, M., et al. (2012). Vers une solution à la dérive du slam visuel en environnement urbain par une connaissance éparse de l'environnement. In *Proc. du 15ème colloque COmpression et REprésentation des Signaux Audiovisuels (CORESA 2012)*.

[C. Engels, 2006] C. Engels, H. Stewenius, D. N. (2006). Bundle adjustment rules. In *PCV*.

[Calonder et al., 2010] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief : Binary robust independent elementary features. In *Computer Vision–ECCV 2010*, pages 778–792. Springer.

[Castellanos et al., 2004] Castellanos, J. A., Neira, J., and Tardós, J. D. (2004). Limits to the consistency of ekf-based slam 1.

[Chow et al., 2012] Chow, J., Ang, K., Lichti, D., and Teskey, W. (2012). Performance analysis of a low-cost triangulation-based 3d camera : Microsoft kinect system. In *Int. Soc. for Photogrammetry and Remote Sensing Congress (ISPRS)*, volume 39, page B5.

[Davison, 2003] Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *ICCV*, pages 1403–1410. IEEE Computer Society.

[Durrant-Whyte and Bailey, 2006] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping : Part1. *IEEE Robotics and Automation Magazine*.

[E. Eade, 2007] E. Eade, T. D. (2007). Monocular slam as a graph of coalesced observations. In *ICCV*.

[E. Royer, 2005] E. Royer, M. Lhuillier, M. D. T. C. (2005). Localization in urban environments - monocular vision compared to a differential gps sensor. In *CVPR*.

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

[Faugeras and Lustman, 1988] Faugeras, O. and Lustman, F. (1988). Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern and Artificial Intelligence*, 2(3) :485–508.

[Fioraio and Konolige, 2011] Fioraio, N. and Konolige, K. (2011). Realtime visual and point cloud slam. In *Proc. of the RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics : Science and Systems Conf.(RSS)*, volume 27.

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

[G. Klein, 2007] G. Klein, D. M. (2007). Parallel tracking and mapping for small a.r. workspaces. *International Symposium on Mixed and Augmented Reality*.

[G. Klein, 2008] G. Klein, D. M. (2008). Improving the agility of keyframe-based slam. In *ECCV*.

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

[H. Bay and Gool, 2006] H. Bay, A. Ess, T. T. and Gool, L. V. (2006). Speeded-up robust features (surf). In *ECCV*.

[Haralick et al., 1994] Haralick, R. M., Lee, C.-N., Ottenberg, K., and Nölle, M. (1994). Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision*, 13(3) :331–356.

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK.

[Hartley and Schaffalitzky, 2004] Hartley, R. and Schaffalitzky, F. (2004). l_∞ minimization in geometric reconstruction problems. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–504. IEEE.

[Hartley and Zisserman, 2004] Hartley, R. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press.

[Hartley and Sturm, 1997] Hartley, R. I. and Sturm, P. F. (1997). Triangulation. *Computer Vision and Image Understanding*, 68(2) :146–157.

[Henry et al., 2010] Henry, P., Krainin, M., Herbst, E., Ren, X., and Fox, D. (2010). Rgb-d mapping : Using depth cameras for dense 3d modeling of indoor environments. In *the 12th International Symposium on Experimental Robotics (ISER)*, volume 20, pages 22–25.

[Herrera et al., 2012] Herrera, C., Kannala, J., Heikkilä, J., et al. (2012). Joint depth and color camera calibration with distortion correction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(10) :2058–2064.

[Holmes et al., 2008] Holmes, S., Klein, G., and Murray, D. W. (2008). A square root unscented kalman filter for visual monoslam. In *Robotics and*

Automation, 2008. ICRA 2008. IEEE International Conference on, pages 3710–3716. IEEE.

[I. Gordon, 2006] I. Gordon, D. L. (2006). *What and where : 3D object recognition with accurate pose*. Springer-Verlag.

[Irschara et al., 2009] Irschara, A., Zach, C., Frahm, J.-M., and Bischof, H. (2009). From structure-from-motion point clouds to fast location recognition. In *CVPR*, pages 2599–2606.

[Julier and Uhlmann, 2001] Julier, S. J. and Uhlmann, J. K. (2001). A counter example to the theory of simultaneous localization and map building. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 4238–4243. IEEE.

[Jung and Lacroix, 2003] Jung, I.-K. and Lacroix, S. (2003). High resolution terrain mapping using low attitude aerial stereo imagery. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 946–951. IEEE.

[Kneip et al., 2011] Kneip, L., Chli, M., and Siegwart, R. (2011). Robust real-time visual odometry with a single camera and an imu. In Hoey, J., McKenna, S. J., and Trucco, E., editors, *BMVC*, pages 1–11. BMVA Press.

[Kuemmerle et al., 2011] Kuemmerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). g2o : A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*.

[Leonard and Durrant-Whyte, 1991] Leonard, J. J. and Durrant-Whyte, H. F. (1991). Simultaneous map building and localization for an autonomous mobile robot. In *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on*, pages 1442–1447. Ieee.

[Lepetit et al., 2009] Lepetit, V., Moreno-Noguer, F., and Fua, P. (2009). Epnp : An accurate $o(n)$ solution to the pnp problem. *International Journal of Computer Vision*, 81(2) :155–166.

[Leutenegger et al., 2011] Leutenegger, S., Chli, M., and Siegwart, R. Y. (2011). Brisk : Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555. IEEE.

[Leutenegger et al., 2013] Leutenegger, S., Furgale, P., Rabaud, V., Chli, M., Konolige, K., and Siegwart, R. (2013). Keyframe-based visual-inertial slam

using nonlinear optimization. In *Proceedings of Robotics : Science and Systems (RSS)*.

[Longuet-Higgins, 1987] Longuet-Higgins, H. (1987). A computer algorithm for reconstructing a scene from two projections. *Readings in Computer Vision : Issues, Problems, Principles, and Paradigms, MA Fischler and O. Firschein, eds*, pages 61–62.

[Lothe et al., 2009] Lothe, P., Bourgeois, S., Dekeyser, F., Royer, E., and Dhome, M. (2009). Towards geographical referencing of monocular slam reconstruction using 3d city models : Application to real-time accurate vision-based localization. In *CVPR*, pages 2882–2889.

[Lothe et al., 2010] Lothe, P., Bourgeois, S., Royer, E., Dhome, M., and Naudet-Collette, S. (2010). Real-time vehicle global localisation with a single camera in dense urban areas : Exploitation of coarse 3d city models. In *CVPR*, pages 863–870. IEEE.

[Lowe, 1999] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *ICCV*.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *IJCV*.

[M. Aron, 2004] M. Aron, G. Simon, M. B. (2004). Handling uncertain sensor in vision based camera tracking. In *ISMAR*.

[Michot, 2010] Michot, J. (2010). *Recherche lineaire et fusion de donnees par ajustement de faisceaux*. PhD thesis, Universite Blaise Pascal.

[Mikolajczyk et al., 2005] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Van Gool, L. (2005). A comparison of affine region detectors. *International journal of computer vision*, 65(1-2) :43–72.

[Montemerlo et al., 2003] Montemerlo, M., Thrun, S., Koller, D., B, and Wegbreit (2003). Fastslam 2.0 : An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *IJCAI*.

[Montemerlo et al., 2002] Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2002). Fastslam : A factored solution to the simultaneous localization and mapping problem. *AAAI*.

[Mouragnon et al., 2006] Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., and Sayd, P. (2006). Real time localization and 3d reconstruction. In *CVPR (1)*, pages 363–370. IEEE Computer Society.

[Mouragnon et al., 2009] Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., and Sayd, P. (2009). Generic and real-time structure from motion using local bundle adjustement. *IVC*.

[Mulloni et al., 2013] Mulloni, A., Ramachandran, M., Reitmayr, G., Wagner, D., Grasset, R., and Diaz, S. (2013). User friendly slam initialization. *IEEE ISMAR*.

[Newcombe et al., 2011] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. W. (2011). Kinectfusion : Real-time dense surface mapping and tracking. In *ISMAR*, pages 127–136.

[Newman, 1999] Newman, P. (1999). *On the structure and solution of the simultaneous localization and mapping problem*. PhD thesis, University of Sydney, Sydney, Australia.

[Nister, 2004] Nister, D. (2004). An efficient solution to the five-point relative pose problem. *PAMI*.

[Nistér, 2005] Nistér, D. (2005). Preemptive ransac for live structure and motion estimation. *Mach. Vis. Appl.*, 16(5) :321–329.

[Nistér et al., 2004] Nistér, D., Naroditsky, O., and Bergen, J. R. (2004). Visual odometry. In *CVPR (1)*, pages 652–659. IEEE Computer Society.

[Nistér et al., 2006] Nistér, D., Naroditsky, O., and Bergen, J. R. (2006). Visual odometry for ground vehicle applications. *J. Field Robotics*, 23(1) :3–20.

[Nister and Stewenius, 2006] Nister, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2161–2168. IEEE.

[Pirchheim et al., 2013] Pirchheim, C., Schmalstieg, D., and Reitmayr, G. (2013). Handling pure camera rotation in keyframe-based slam. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 229–238. IEEE.

[Pollefeys et al., 2008] Pollefeys, M., Nister, D., Frahm, J.-M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S.-J., Salmi, C., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewenius, H., Yang, R., Welch, G., and Towles, H. (2008). Detailed real-time urban 3d reconstruction from video. In *IJCV*.

[Roberts, 1963] Roberts, L. G. (1963). *MACHINE PERCEPTION OF THREE-DIMENSIONAL soups*. PhD thesis, Massachusetts Institute of Technology.

[Rosten and Drummond, 2006] Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006*, pages 430–443. Springer.

[Rousseeuw and Leroy, 1987] Rousseeuw, P. and Leroy, A. (1987). *Robust Regression and Outlier Detection*. Wiley, New-York.

[Rublee et al., 2011] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb : an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE.

[Scaramuzza et al., 2009] Scaramuzza, D., Fraundorfer, F., and Siegwart, R. (2009). Real-time monocular visual odometry for on-road vehicles with 1-point ransac. In *ICRA*, pages 4293–4299.

[Scherer et al., 2012] Scherer, S. A., Dube, D., and Zell, A. (2012). Using depth in visual simultaneous localisation and mapping. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 5216–5221. IEEE.

[Servant, 2009] Servant, F. (2009). *Localisation et cartographie simultanées en vision monoculaire et en temps réel basé sur les structures planes*. PhD thesis, Université de Rennes 1.

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE.

[Smisek et al., 2013] Smisek, J., Jancosek, M., and Pajdla, T. (2013). 3d with kinect. In *Consumer Depth Cameras for Computer Vision*, pages 3–25. Springer.

[Smith et al., 1988] Smith, R., Self, M., and Cheeseman, P. (1988). A stochastic map for uncertain spatial relationships. In *Proceedings of the 4th international symposium on Robotics Research*, pages 467–474, Cambridge, MA, USA. MIT Press.

[Sourimant et al., 2007] Sourimant, G., Morin, L., and Bouatouch, K. (2007). Gps, gis and video registration for building reconstruction. In *ICIP (6)*, pages 401–404.

[Strasdat, 2012] Strasdat, H. (2012). *Local Accuracy and Global Consistency for Efficient Visual SLAM*. PhD thesis, Imperial College.

[Strasdat et al., 2010] Strasdat, H., Montiel, J. M. M., and Davison, A. J. (2010). Real-time monocular slam : Why filter ? In *ICRA*, pages 2657–2664. IEEE.

[Teichman et al., 2013] Teichman, A., Miller, S., and Thrun, S. (2013). Unsupervised extrinsic calibration of depth sensors in dynamic scenes. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2695–2702. IEEE.

[Thrun et al., 2004] Thrun, S., Liu, Y., Koller, D., Ng, A. Y., Ghahramani, Z., and Durrant-Whyte, H. (2004). Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, 23(7-8) :693–716.

[Torr and Zisserman, 2000] Torr, P. H. S. and Zisserman, A. (2000). Mlesac : A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1) :138–156.

[Triggs et al., 1999] Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W. (1999). Bundle adjustment - a modern synthesis. In Triggs, B., Zisserman, A., and Szeliski, R., editors, *Workshop on Vision Algorithms*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372. Springer.

[Zach,] Zach, C. Simple sparse bundle adjustment. `http://www.inf.ethz.ch/personal/chzach/opensource.html`.

[Zhang, 2000] Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11) :1330–1334.

[Zhang et al., 1995] Zhang, Z., Deriche, R., Faugeras, O. D., and Luong, Q.-T. (1995). A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artif. Intell.*, 78(1-2) :87–119.