Thèse de doctorat

Université de Limoges

ED 610 - Sciences et Ingénierie des Systèmes, Mathématiques, Informatique (SISMI)

Xlim

Thèse pour obtenir le grade de
Docteur de l'Université de Limoges
Robotique

Présentée et soutenue par
Bilel CHENCHANA

Le 22 mars 2019

Localisation collaborative visuelle-inertielle de robots hétérogènes communicants

Thèse dirigée par Ouiddad LABBANI-IGBIDA, Stéphane RENAULT, Sébastien BORIA

JURY :

Président du jury
Olivier SIMONIN, Professeur des universités à INSA Lyon

Rapporteurs
Roland CHAPUIS, Professeur des universités à l'Institut Pascal
Pascal MORIN, Professeur des universités à Sorbonne Université

Examinateurs
Isabelle FANTONI, Directrice de recherche CNRS au LS2N

Invités
Jérôme Golenzer, Ingénieur Airbus Group

Dédicace, à mes parents et mes sœurs

Opinions are something in between knowledge and ignorance

Bilel

Remerciements

La réalisation de ce travail a été possible grâce au concours de plusieurs personnes à qui je voudrais témoigner toute ma reconnaissance.

Je voudrais tout d'abord adresser toute ma gratitude à ma directrice de thèse Ouiddad LABBANI-IGBIDA, Merci pour ton optimisme, ton encouragement et pour les moyens dont tu nous fais bénéficier. Merci, enfin, pour l'aide que tu m'as apportée : quand il le fallait, et sans compter. Merci à Stéphane RENAULT pour son aide et sa patience et à Sébastien Boria pour le sujet de thèse extrêmement enrichissant.

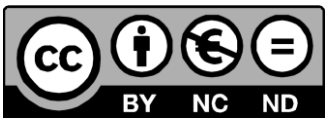
Je remercie Messieurs Roland CHAPUIS et Pascal MORIN d'avoir accepté d'être les rapporteurs de ce travail de thèse. Je remercie également Monsieur Olivier SIMONIN et madame Isabelle FANTONI d'avoir été présents en tant qu'examinateurs de ce travail.

Je remercie mes collègues les doctorants Hela, Fatima, Brice, Guillaume, Karima avec qui j'ai passé toutes ses années.

Je tiens enfin à remercier les personnes proches qui m'ont toujours soutenu, supporté et encouragé. Merci à ma famille, mes parents, mes sœurs et mon oncle.

Bilel Chenchana

Droits d’auteurs



Localisation collaborative visuelle-inertielle de robots hétérogènes communicants

La capacité à se localiser est d'une importance cruciale pour la navigation des robots. Cette importance a permis le développement de plusieurs techniques de localisation de grande précision. Notre contribution consiste à proposer un passage de la technique de localisation visuelle inertielle du cas individuel, au cas multi collaboratif. Ce travail a pour objectif d'aboutir à une localisation collaborative aussi rapide, robuste et précise que la technique individuelle de départ. Notre approche se base sur le filtrage en couplage serré Multi State Constraint Kalman Filter (MSCKF) pour la fusion de données. Les caractéristiques de ce filtrage sont d'abord étudiées dans le cas individuel pour tester la robustesse et la précision dans différentes conditions et avec différents modèles d'observation. Les résultats de cette étude nous ont orienté vers la structure la mieux adaptée à une augmentation au cas de localisation collaborative. L'algorithme collaboratif proposé, est basé sur un processus hiérarchique en trois étapes. Une localisation collaborative est initialisée sur la base des mesures relatives de distances Ultra Large Bande (ULB). Une localisation collaborative améliorée se base ensuite sur le chevauchement des images en utilisant un modèle de mesure adapté, et une structure de fusion de données qui absorbe l'excédent en temps de calcul provoqué par le traitement collaboratif. Enfin, pour augmenter la précision, une extraction des contraintes de structure environnement, suivie d'une intégration à l'aide d'une troncature dans le filtre sont proposées.

Mots-clés : Localisation, VINS, MSCKF, ULB, Collaboration

Visual-inertial collaborative localization of communicating heterogeneous robots

Localization is of crucial importance for robots navigation. This importance has allowed the emergence of several precise localization techniques. Our contribution consists of proposing a transition from an individual inertial visual localization technique to the multi-robots collaborative localization case. This work aims to achieve a collaborative localization as fast, robust and accurate as the individual starting technique. We adopt a tightly coupled MSCKF (Multi State Constraint Kalman Filter) approach to achieve the data fusion. The characteristics of this data fusion are first studied in the individual case to test the robustness and the precision under different conditions and with different observation models. The results of this study directed us towards the best structure adapted to an augmentation to the collaborative localization case. The proposed collaborative algorithm is a hierarchical process of three stages. A collaborative localization is initialized based on the relative distance measurements using Ultra-Wide Band (ULB) sensors. Then, a collaborative localization based on images overlapping using a suitable measurement model, and a data fusion structure that absorbs the computation time excess caused by the collaboration is achieved. Finally, to increase precision, an extraction of the environment constraints, followed by an integration using a truncation in the filter are proposed.

Keywords : Localization, VINS, MSCKF, ULB, Collaboration

Table des matières

Notations

Localisation Individuelle

$\{G\}\{C\}\{I\}$	:	repère global, repère caméra et repère IMU
X_I	:	vecteur d'état IMU
X_k	:	vecteur d'état individuel à l'instant k
$\tilde{X}_k,\ \hat{X}_k$	:	vecteur d'erreur d'état et vecteur d'état nominal à l'instant k
$\tilde{X}_{k\|k-1}$	:	vecteur d'état après propagation
$P_{k\|k-1}$	:	matrice de covariance après propagation
$\tilde{X}_{k\|k}$	:	vecteur d'état après mise à jour
$P_{k\|k}$	:	matrice de covariance après mise à jour
${}^{\beta}p_{\alpha}$	:	vecteur position de α dans le repère β
${}^{\beta}R_{\alpha}$	:	matrice de rotation de α exprimée dans le repère β
${}^{\beta}\overline{q}_{\alpha}$	:	quaternion de la rotation ${}^{\beta}R_{\alpha}$
${}^{G}v_I$	:	vecteur vitesse inertielle dans le repère global
$b_a,\ b_g$	:	vecteurs des biais accéléromètre et gyroscope
m	:	fenêtre de poses MSCKF
${}^{G}g$	:	vecteur gravité
a_m	:	vecteur accélération mesurée
ω_m	:	vecteur vitesse angulaire mesurée
$O_3,\ I_3$	:	respectivement matrice nulle et identité (3 x 3)
n_{ba}, n_{bg}	:	bruits blancs gaussiens des biais
$n_a,\ n_g$	:	vecteurs bruits blancs gaussiens accéléromètre et gyroscope
n	:	vecteur de bruits contenant $[n_a^T\ n_{ba}^T\ n_g^T\ n_{bg}^T]$
$F_c,\ G_c,\ F_d,\ G_d$	:	matrices du système linéarisé : continu, discret
$Q_d,\ Q_c$	:	matrices de covariance
A	:	matrice de projection sur l'espace nul
(u, v)	:	coordonnées d'un pixel dans le plan image
$z_j^i,\ \hat{z}_j^i$	:	vecteurs (2×1) représentant la mesure et l'estimation de la mesure
${}^{Ci}X_j, {}^{Ci}Y_j, {}^{Ci}Z_j$	:	position d'un point j dans le repère de la caméra C_i
n_j^i	:	vecteur (2×1) de bruit d'image
R_j^i	:	matrice de covariance du bruit de mesure (image)
$f_j,\ {}^{G}p_{fj}$	:	un point d'intérêt et sa position dans le repère global
K	:	matrice de calibration
C_1, C_2, C_3	:	centres des caméras
R_{12}, t_{12}	:	rotation et translation relative entre deux images indexées 1 et 2
T_i	:	tenseur trifocal
L	:	dimension du vecteur d'état
λ	:	un paramètre d'échelle dans l'UKF
W_c^l , W_s^l	:	matrices de pondération utilisées en UKF
r_j^i	:	résidu EKF

$H_{Xi}^{(j)}$, $H_f^{(j)}$	:	Jacobiennes de l'équation de mesure relativement à l'état et aux points d'intérêt
K_k	:	gain de Kalman

Localisation Collaborative

X_k^c	:	vecteur d'état collaboratif à l'instant k
${}^G p_{rn}$, ${}^G \overline{q}_{rn}$	:	transformation entre les repères origines du robot n et du robot local dans le repère G
d, $\hat{d}$	:	distance et son estimée entre deux robots
w	:	nombre de mesures de distance par mise à jour
g_i, g_j	:	repères initiaux de deux robots i et j
t_{UWB}	:	intervalle de temps de mise à jour par capteur ULB
M	:	nombre de mesures utilisées pour le calcul de la running variance
h, h_1, h_2, h_{ULB}	:	les modèles de mesure utilisés

Abréviations

ULB	:	Ultra Large Bande
LOS, NLOS	:	Line Of Sight, Non Line Of Sight
Img_k	:	image numéro k
me1, me2	:	équations de mesure 1 et 2
GT	:	Ground Truth
RMSE	:	Root Mean Square Error
UKF	:	Unscented Kalman Filter
EKF	:	Extended Kalman Filter
MSCKF	:	Multi-State Constraint Kalman
IMU	:	Inertial Measurement Unit

Introduction Générale

Contexte

La dernière décennie a connu un grand rapprochement entre la vision des roboticiens de la place des robots dans la vie et les progrès réalisés dans cette direction. Les robots sont plus que jamais présents dans le monde de l'industrie et du service. Avec des visions de plus en plus ambitieuses comme l'usine du futur, les cobots et la conception des robots collaboratifs, cette intégration continuera probablement à croître dans le futur. L'une des caractéristiques clés dans cette vision future est la collaboration entre robots. Ces derniers permettront une plus grande productivité et une meilleure flexibilité pour répondre aux variations et aux exigences d'une application. La collaboration de robots nécessite une connaissance de l'environnement, et plus précisément une connaissance de l'état des robots en collaboration. L'état d'un robot peut être constitué de plusieurs grandeurs physiques, mais la plus importante pour des robots mobiles, volants ou humanoïdes, reste leurs positions dans un repère connu par le groupe collaboratif. Les progrès dans la localisation collaborative permettront une collaboration plus efficace qui impactera significativement la rentabilité des robots.

Les travaux de cette thèse CIFRE (Convention Industrielle de Formation par la Recherche) présentés dans ce manuscrit ont été réalisés dans le laboratoire XLIM de l'Université de Limoges en collaboration avec le Airbus Group. Ce travail s'inscrit dans le cadre du projet Usine du Futur, qui vise une vaste intégration des robots dans l'environnement industriel.

Problématique

Pour localiser un robot, une fusion des données ou d'informations capteurs permet d'estimer sa pose, avec l'objectif de réduire l'incertitude sur l'information résultante. Ces données fusionnées peuvent provenir de capteurs embarqués sur le robot lui-même, de capteurs délocalisés, ou de mesures relatives entre plusieurs robots. Dans ce dernier cas, il s'agit alors d'un problème de localisation collaborative.

La collaboration ne doit pas empêcher ou restreindre l'autonomie. Il est alors indispensable de construire un estimateur de pose capable à la fois de donner des estimations avec et sans besoin de collaboration. La raison est qu'une collaboration n'est généralement possible que sur un petit intervalle de temps dans plusieurs applications. Ces intervalles de collaboration sont une opportunité pour atteindre une meilleure précision ou pour résoudre les difficultés de localisation que peut avoir un seul robot.

Dans cette thèse, nous visons un système de localisation qui peut à la fois localiser un robot mobile ou un drone. Cette hétérogénéité et le fait d'inclure les robots aériens comme les quadrirotors, imposent de chercher un certain degré de robustesse dans l'algorithme

de localisation. En effet, contrairement à un robot mobile terrestre qui peut s'immobiliser pour réduire les conséquences d'une mauvaise localisation, ces conséquences sur un drone ne peuvent être évitées.
Il existe actuellement plusieurs algorithmes pour localiser un robot, et probablement la majorité permettront d'une manière ou d'une autre de proposer une localisation collaborative. Notre objectif est de trouver une structure de localisation que l'on peut qualifier de naturellement extensible au collaboratif puisqu'elle proposera, un gain en temps de calcul, une simplicité sans diminuer la précision, et surtout une robustesse dans les deux cas de localisation, individuelle et collaborative.

Les problématiques majeures entourant la construction d'une telle localisation collaborative sont : (1) le temps de calcul, où les solutions de localisation individuelle, lorsque implémentées sur un drone, nous ramènent déjà à la limite du temps de calcul disponible. L'extension vers la collaboration sera souvent difficile. Dans cette problématique, les algorithmes basés sur le filtrage de Kalman sont dans le contexte de la localisation précise de pose (moins de 1% d'erreur) les plus prometteurs en temps de calcul, mais ces derniers souffrent d'un problème d'initialisation connu dans le cas du multi. (2) La collaboration nécessite d'avoir des observations relatives. Ceci n'est pas souvent facile à avoir directement, surtout si l'information relative ciblée est une position, une orientation ou une pose complète. A cette problématique se conjugue souvent deux autres problèmes : un problème de communication où on a souvent un réseau de communication à bande passante réduite et à grande latence ; et un problème de synchronisation des données échangées entres les robots.
Toutes ces problématiques imposent une étude approfondie des algorithmes de localisation, afin de pouvoir identifier des éléments permettant d'éliminer ces contraintes qui entourent les algorithmes de localisation collaborative.

Contribution

La contribution principale de cette thèse est la proposition d'une architecture de localisation à couplage serré, basée sur un MSC-TUKF collaboratif.
Les sous-contributions réalisant l'objectif final sont :

— Une étude approfondie de la robustesse des variantes du MSCKF au type de mouvement et aux éléments de perturbation existants dans les algorithmes de localisation visuelle inertielle.
— La proposition d'un algorithme de localisation individuelle, extensible au cas de localisation collaborative temps réel.

— La proposition d'une extension ULB pour la localisation collaborative. Cette dernière peut servir comme une solution de géo-référencement de l'intérieur dans des applications de cartographie et d'exploration après catastrophe, ou comme une initialisation pour des futures mises à jour utilisant d'autres capteurs.
— La proposition d'une technique d'extraction des contraintes d'environnement ne nécessitant pas de puissance de calcul externe (exemple Fig.1).

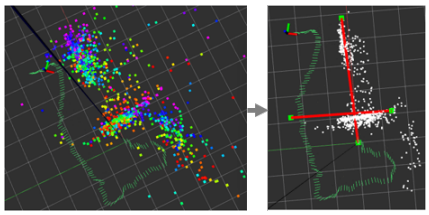


FIGURE 1 – Résultat d'extraction des contraintes d'environnement -les lignes rouges sont les contraintes utiles pour l'amélioration de l'estimation.

— La proposition d'un MSC-TUKF pour l'intégration des contraintes d'environnement.
— La proposition d'une architecture de mise à jour collaborative basée-caméra avec un minimum d'impact sur l'odométrie individuelle du robot (Fig.2).

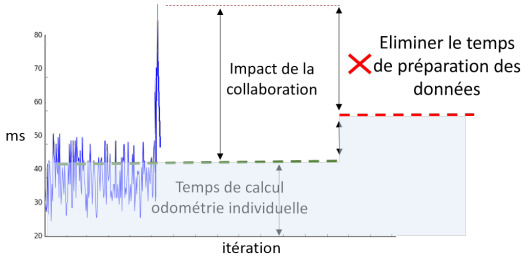


FIGURE 2 – Gain en temps de calcul dans l'architecture proposée.

Structuration du mémoire

L'organisation du manuscrit est illustrée sur la Fig.3 :

Chapitre 1- Localisation — concepts et état de l'art

Dans la première partie de ce chapitre, l'état de l'art des techniques de localisation visuelle est réalisé. Nous présentons les principaux algorithmes existants tout en illustrant les méthodes qui les constituent. La deuxième partie aborde la fusion des données caméra avec les données d'autres capteurs, soit provenant d'un autre capteur embarqué sur le robot lui même, soit de données provenant des robots voisins dans le contexte de la

localisation collaborative. Un état de l'art orienté vers l'intégration des données IMU et ULB sera présenté dans cette partie.

Chapitre 2- Localisation d'un robot par odométrie visuelle inertielle

Dans cette partie, on réalise une étude approfondie comprenant des tests réels et dans plusieurs situations, de plusieurs variantes du filtrage MSCKF. Certaines de ces variantes existent dans l'état de l'art et d'autres sont proposées dans le cadre de notre recherche de la variante idéale qui peut servir d'un algorithme de base pour notre futur algorithme de localisation collaborative. Une architecture en couplage serré des capteurs choisis est adoptée.

Chapitre 3- Localisation collaborative avec un Multi-MSCKF

Après la proposition d'une configuration de localisation individuelle, une architecture de localisation collaborative est proposée dans ce chapitre. Dans cette partie, on adressera des problématiques liées aux capteurs utilisés en collaboration. On proposera des algorithmes de fusion collaborative sans besoin de puissance de calcul externe. Des méthodes pour optimiser les résultats de la collaboration avec des observations extraites de l'environnement seront aussi proposées et testées sur différents types de cibles.

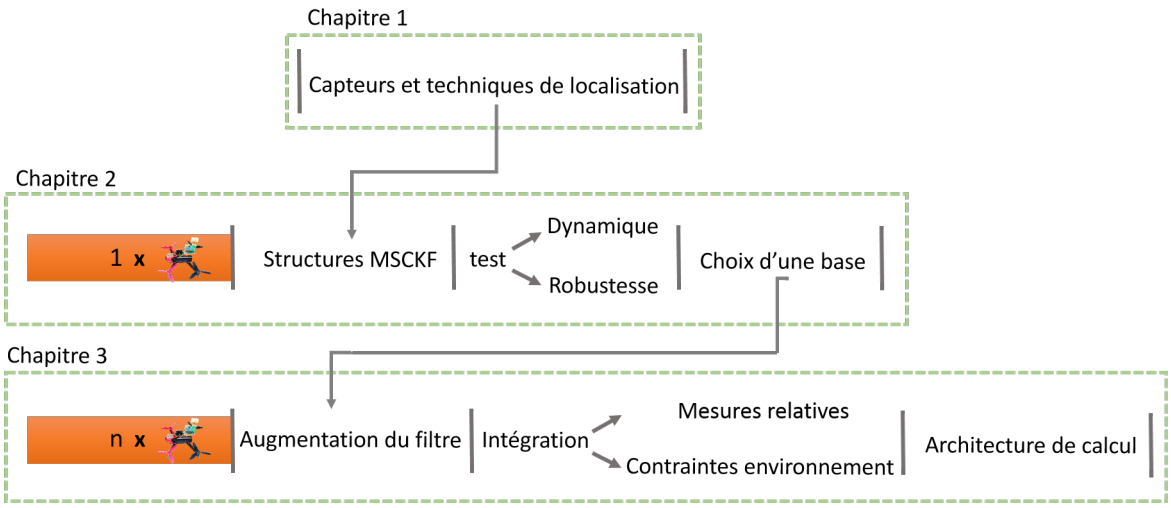


FIGURE 3 – Structuration des propositions de la thèse.

Chapitre 1 :

Localisation — concepts et état de l'art

1.1 Introduction

Localiser un robot fait partie des tâches les plus prioritaires dans un système robotique. Ceci est lié au fait que la localisation d'un robot est souvent couplée à l'objectif de sa mission, mais aussi à la commande de contrôle du robot lui-même.
Pour un robot volant comme les quadrirotors, la localisation devient une question de survie pour le drone. En effet, il est complètement impossible d'envisager un vol autonome sans localisation vu que la stabilisation de vol est entièrement basée sur sa capacité à se localiser.
Pour augmenter l'autonomie des robots, différentes techniques et capteurs ont été adoptés en robotique. Le GPS (Global Positionning System), lorsque la mission est à l'extérieur, est l'une des solutions de localisation [Lee and Chung, 2015][Lee et al., 2009][Lim et al., 2012] nécessitant une infrastructure. Cette technologie fait partie des systèmes les plus utilisés pour sa fiabilité. C'est aussi l'une des technologies les plus anciennes qui a connu un grand développement. D'autres solutions locales fonctionnant avec le même principe, nécessitant une infrastructure, ont vu le jour pour répondre à des besoins de localisation des objets dans des structures telles que les usines, les hôpitaux, etc. Jusqu'à présent, on peut distinguer une dizaine de ces solutions, mais les plus répandues sont des systèmes de balises Wifi [Fan and Chen, 2016], Radio [Fink et al., 2012], Ultra-son [Yi and Chu-na, 2010] et Ultra Large Bande [Wang et al., 2017a]. Toutes ces technologies ont un point commun, qui est la nécessité d'avoir une infrastructure, pouvant être simplement un nombre de balises placées dans des positions fixes dans l'espace où se trouve l'objet à localiser.
Le VICON [Bai et al., 2017] est une autre facette de cette technologie à infrastructure. Ce système de localisation très précis utilise plusieurs caméras placées autour d'un objet portant des marqueurs détectables. C'est l'un des systèmes les plus précis pour l'estimation de mouvement. Il est souvent utilisé dans la capture et l'analyse des mouvements biologiques ou l'enregistrement d'une réalité terrain d'un mouvement de robot. L'avantage de ces solutions à infrastructure est de fournir une localisation absolue de l'objet mobile, indépendamment de sa configuration initiale.
Cependant, le GPS ne fonctionne pas à l'intérieur et le reste des solutions à infrastructures sont des solutions locales. Ce sont les majeurs inconvénients de cette catégorie de solutions. D'autres technologies de capteurs ne nécessitant aucune infrastructure essayent depuis des décennies de remédier à ce problème, avec un succès pour certaines solutions.
Solutions basées télémétrie : certaines solutions de localisation comme le SLAM (Simultaneous Localization and Mapping) montrent qu'une localisation basée sur un capteur laser, permet d'avoir une estimation de pose d'une grande précision [Javanmardi et al., 2017][Droeschel and Behnke, 2018]. Beaucoup d'algorithmes dans

la littérature utilisent des variantes de ce capteur pour une localisation adaptée à l'application robotique ciblée. Les capteurs Hokuyo et Kinect [Hamzeh and Elnagar, 2015] sont des variantes souvent utilisées en laboratoires de recherche, le télémètre Velodyne est la version laser souvent utilisée pour la localisation et la navigation des voitures autonomes.

Solutions basées caméra : l'un des capteurs les plus prometteurs pour faire de l'odométrie est le capteur caméra. Ceci réside dans le fait que ce capteur est l'un des plus compacts et simples d'intégration que l'on peut trouver pour des prix de moins en moins chers. La même stratégie de localisation et cartographie SLAM utilisée pour les lasers est aussi possible à l'aide du capteur caméra [Mur-Artal et al., 2015a]. Contrairement à un laser où on n'a que l'information de profondeur, la caméra peut permettre de remonter à cette information, mais en plus, d'avoir d'autres informations. Ces dernières peuvent servir soit pour améliorer la localisation, soit pour aider à la réalisation de la mission du robots. Plusieurs variantes d'algorithmes d'estimation de pose basés caméra existent dans la littérature, utilisant des caméras monoculaire, binoculaire et dernièrement des capteurs à temps de vol [Fuchs, 2010]. Ces derniers ne transmettent que les pixels en mouvement dans l'image, et permettent ainsi des temps d'acquisition allant jusqu'à 1khz.

Accéléromètre, gyroscope : ces capteurs mesurent respectivement l'accélération et la vitesse angulaire sur les trois axes. Ceci permet une estimation de pose qui n'est pas très mauvaise à court terme. Ils sont très faciles à intégrer, vu que la taille de ces capteurs est inférieure à une pièce de monnaie pour les versions low cost (Fig.1.1). Ceci les a rendu des capteurs incontournables et nécessaires sur tous les drones, vu que le système de stabilisation de vol est souvent entièrement basé sur le filtrage des données de ces capteurs. Ils sont généralement utilisés en localisation fusionnés avec des capteurs plus précis comme la caméra ou le laser. Une estimation de pose basée juste sur l'intégration des données de ces capteurs aura une dérive conséquente.

Altimètre, magnétomètre : ces deux capteurs sont eux aussi des capteurs utilisés fusionnés avec un autre capteur principal plus précis. Ce sont deux capteurs que l'on trouvera souvent embarqués sur des drones. Le premier permet, grâce à l'estimation de la pression atmosphérique, de donner une information sur l'axe vertical du mouvement, ce qui est utile pour la stabilisation d'altitude des drones. Le magnétomètre est un capteur capable de mesurer le champ magnétique suivant les trois axes, ce qui lui permet grâce au champ magnétique terrestre de fixer le Nord, donnant une information d'orientation globale aux robots.

Plusieurs autres capteurs et techniques permettant la localisation existent dans la

littérature (odomètre, ultrasons, etc). Les techniques basées-laser ou caméra sont de nos jours les plus précises et les plus performantes pour la localisation en robotique.

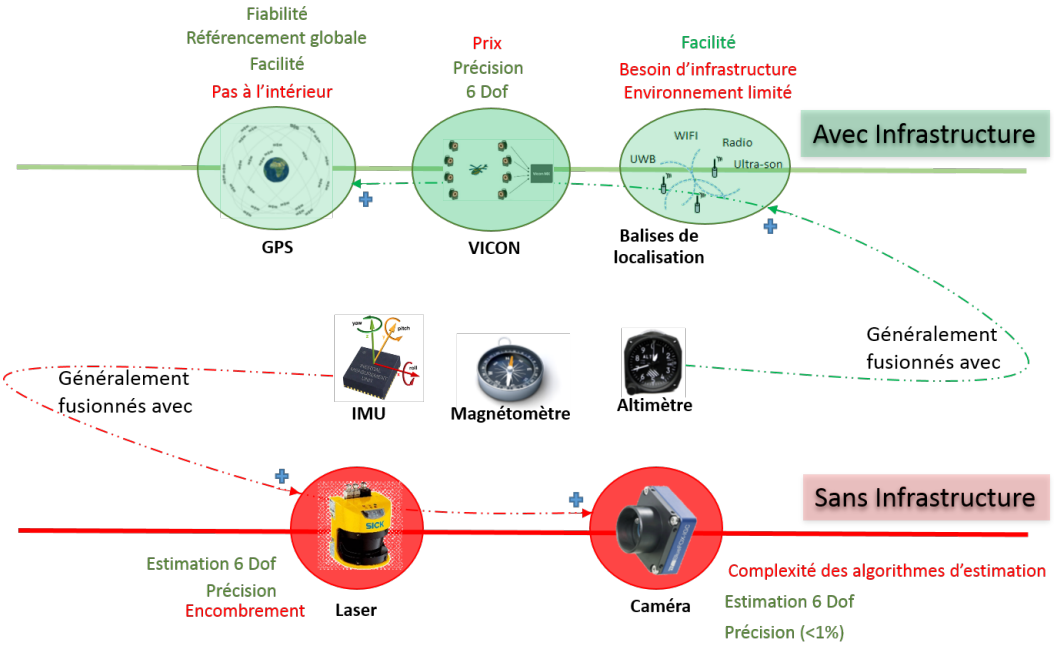


FIGURE 1.1 – Catégories de capteurs utilisés dans des applications de localisation en robotique.

1.2 Odométrie visuelle

La technique d'odométrie visuelle (VO) consiste à estimer la position et l'orientation de la caméra ou de l'objet porteur de celle-ci en étudiant l'évolution des informations contenues dans des images successives. L'odométrie visuelle a connu une évolution spectaculaire depuis 1980 et essentiellement en 2004 par le travail de Nister [Nister et al., 2004] dans la publication "Visual Odometry". Certes, cette technique contient beaucoup de contraintes liées à la nature du capteur lui-même comme la nécessité d'éclairage, des contraintes de géométrie qui font qu'il soit nécessaire d'avoir un déplacement entre les images utilisées, ou même des problèmes de capacité de calcul qui font que certains algorithmes d'odométrie visuelle, malgré leur précision, sont simplement impossibles à embarquer sur des systèmes robotiques à capacité de calcul limitée. Malgré ces contraintes, la recherche scientifique dans le domaine de l'odométrie visuelle a permis le développement de plusieurs types d'algorithmes, donnant des erreurs d'estimation de pose qui peuvent être inférieures à 0.6% de la trajectoire parcourue [Geiger et al., 2013].

1.2.1 Approches de localisation visuelle

Pour estimer la pose d'un mobile à partir d'une succession d'images, plusieurs techniques existent dans la littérature. Selon l'information utilisée dans l'image et selon la technique d'estimation de pose, on peut constater qu'il existe trois grandes familles d'algorithmes : les méthodes basées-points d'intérêt, les méthodes directes et les méthodes hybrides qui combinent les bonnes qualités des deux approches, Fig.1.2.

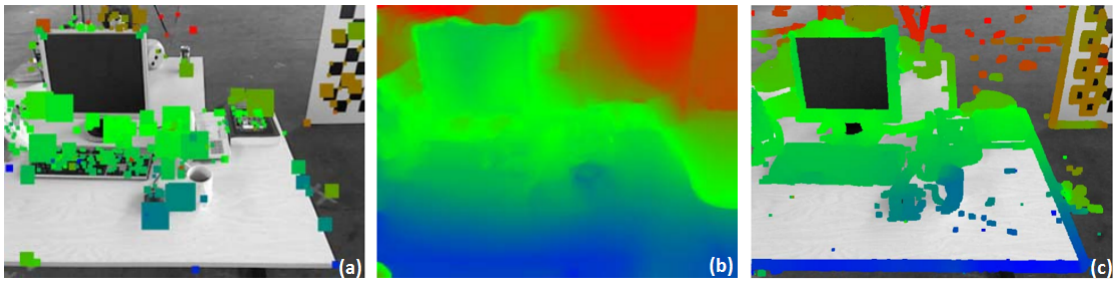


FIGURE 1.2 – Informations utilisées dans l'image [Engel et al., 2013], (a) méthode basée-points d'intérêt, (b) méthode directe, (c) méthode semi-directe.

1.2.1.1 Méthodes directes

Cette technique estime la pose de la caméra en utilisant toutes les informations dans une image [Meng et al., 2016][Wang et al., 2017b][Caruso et al., 2015]. Elle s'appuie sur une minimisation photométrique entre deux images pour trouver une estimation d'un mouvement effectué entre les deux. Si le fait d'utiliser toutes les informations dans une image permet d'avoir plus de précision, cette technique est souvent impossible à implémenter en temps réel sauf en cas d'utilisation d'un GPU ou CPU très performant (avec un capteur de profondeur). Ceci a poussé les développements à proposer des méthodes utilisant juste les régions à fort gradient dans l'image pour alléger les calculs [Schöps et al., 2014][Engel et al., 2013]. On peut noter aussi que les méthodes directes sont :

— Précises (utilisent une abondance d'information sur l'image)
— Simples (permettent d'estimer la pose directement)
— Mais non robustes à l'occultation.

À cause du temps de calcul, elles sont peu utilisées dans l'estimation de pose de robots qui possèdent souvent une capacité de calcul limitée.

1.2.1.2 Méthodes basées points d'intérêt

Dans cette méthode, on commence par extraire certains points ou régions d'intérêt dans l'image, pour ensuite trouver une correspondance de ces points dans la succession

d'images. Une fois cette étape franchie, les correspondances sont utilisées comme contraintes géométriques pour remonter à la transformation entre les deux images [Mur-Artal et al., 2015a].

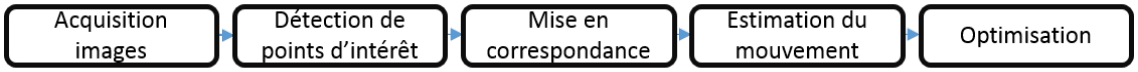


FIGURE 1.3 – Structure des techniques basées points d'intérêt.

Dans l'état de l'art de localisation visuelle basée points d'intérêt, on trouve la librairie Libviso2 [Geiger et al., 2011]. Elle permet, par une approche simple de mise en correspondance d'estimer la localisation d'un robot avec une caméra monoculaire ou une paire stéréo.
Cette librairie permet en plus de l'estimation de pose, la création d'une carte. Ceci n'est pas son objectif principal, car une structure différente est nécessaire pour avoir une carte précise. Mais comme on peut le voir sur la Fig.1.6, sur certains types de mouvement, le rendu n'est pas mauvais.

Dans ce qui suit, une vue globale des techniques utilisées dans les différentes étapes de ce type d'algorithmes est présentée. Il est important de mentionner que ce type d'algorithmes est le plus répandu en robotique, essentiellement pour sa capacité de calcul nécessaire, qui est moindre relativement aux méthodes directes.

1.2.2 Descripteurs et mise en correspondance

1.2.2.1 Détection de points d'intérêt

Dans cette phase, les points d'intérêt qui sont des points saillants pouvant être retrouvés et donc mis en correspondance dans l'image suivante, sont sélectionnés. Ces points saillants sont des régions de l'image qui diffèrent de leur voisinage par la couleur, la texture ou l'intensité. Les points d'intérêt peuvent être de simples points sur l'image, des lignes, des régions, des coins ... etc. Les caractéristiques nécessaires dans un détecteur de points d'intérêt sont : sa capacité à détecter les points avec précision sur plusieurs échelles, la répétabilité (être capable de détecter les mêmes points dans différentes images), l'efficacité en temps de calcul et la robustesse aux changements d'éclairage et aux changements de géométrie comme les distorsions, les rotations et l'échelle. Dans l'état de l'art, plusieurs détecteurs de points d'intérêt (Harris [Harris and Pike, 1987], Shi-Tomasi [C. Tomasi and J. Shi, 1994], FAST [Rosten and Drummond, 2006]) et de Blob (SIFT [Lowe, 2004], SURF [Bay et al., 2006]) sont disponibles Fig.1.4. Cependant, il est vraiment important d'évaluer et de comparer les caractéristiques de ces détecteurs pour pouvoir choisir le mieux adapté en fonction de l'application.

FIGURE 1.4 – Exemple de détection de points d'intérêt, (a) SIFT, (b) ORB, (c) BRISK.

	Détecteur de coins	Détecteur Blob	Invariance à la rotation	Invariance à l'échelle	Transformation affine	Répétabilité	Précision de localisation	Robustesse	Efficacité
Harris	*		*			***	***	**	**
Shi-Tomasi	*		*			***	***	**	**
FAST	*		*	*		**	**	**	****
SIFT		*	*	*	*	***	**	***	*
SURF		*	*	*	*	***	**	**	**

FIGURE 1.5 – Comparaison des caractéristiques des détecteurs de points d'intérêt [Scaramuzza and Fraundorfer, 2011].

Chaque détecteur a ses avantages et ses inconvénients. Les détecteurs de coins sont rapides à calculer, mais sont moins distinctifs que les détecteurs de blobs, qui sont eux plus lents mais plus distinctifs. Un autre point est que les coins sont mieux répartis sur une image comparés aux blobs, mais sont moins bien localisables en échelle. Ceci explique pourquoi les coins ne peuvent pas être re-détectés aussi bien que les blobs en cas d'important changement d'échelle ou de point de vue. Les caractéristiques de certains points d'intérêt sont résumées dans la Fig.1.5.

1.2.2.2 Descripteurs de points d'intérêt

Les points d'intérêt détectés dans l'étape précédente sont associés à des descripteurs. Ces derniers peuvent être comparés avec des descripteurs obtenus sur une autre image, avec le but de trouver une correspondance entre les points d'intérêt des deux images. Un descripteur peut être simplement, l'intensité des pixels autour d'un point. Dans certains cas, la SSD "Sum of Squared Distances" ou la NCC "Normalized Cross Correlation" sont utilisées comme des métriques de similarité pour comparer deux descripteurs. D'autres métriques comme la distance de Hamming sont aussi utilisées si l'application nécessite plus de robustesse. Le problème avec les mesures comme SSD ou NCC, est qu'elles ne sont pas invariantes à l'échelle et au changement de perspective, ce qui limite leur utilisation à des images avec de petits changements de points de vue. Le SIFT (Scale Invariant Feature Transform) est l'un des descripteurs les plus populaires grâce à de son invariance à l'échelle, au changement de point de vue et à l'orientation. Ceci est possible grâce à une représentation multi-échelle du descripteur. Ce dernier est représenté sous forme d'un

vecteur à 128 dimensions correspondant à un histogramme des orientations locales des contours, habilement pondérées et normalisées pour plus de stabilité.
Plus récemment, le descripteur BRIEF [Calonder et al., 2010] a été introduit, il est de plus en plus utilisé. Simple et rapide, il utilise la comparaison d'intensité d'un patch autour du point d'intérêt. Sur le même principe, ORB [Rublee et al., 2011] et BRISK [Leutenegger et al., 2011] ont été proposés avec l'objectif de résoudre le problème d'invariance à la rotation que le BRIEF contient. Un exemple de détection de points d'intérêt est donné dans la Fig.1.4.

1.2.2.3 Mise en correspondance

La mise en correspondance (ou le matching) des points d'intérêt est l'opération de rechercher parmi les points d'une image précédente, celui qui correspond à un point existant sur l'image courante. La méthode la plus simple pour matcher les points est de comparer le descripteur de ce point avec tous les descripteurs des points d'intérêt sur l'autre image. Ce qui nécessite la définition d'une mesure de similarité entre les descripteurs. Si le descripteur représente l'apparence local d'un point d'intérêt, alors une bonne mesure de correspondance serait le SSD ou le NCC. Pour un descripteur SIFT par exemple, la mesure serait la distance Euclidienne. À la fin de cette étape, il y a une forte probabilité d'avoir une liste contenant plusieurs fausses correspondances (outliers). Ces fausses correspondances ont un impact important sur l'odométrie visuelle, ce qui impose leur élimination. Pour le faire, des techniques basées estimateur robuste tel que le RANSAC peuvent être utilisées, une technique sera proposée dans le Chapitre 2.

1.2.3 Estimation de pose à partir des correspondances

Une fois une liste de correspondances de points d'intérêt est disponible, les techniques suivantes sont généralement utilisées pour estimer la pose :

1.2.3.1 Estimation de pose basée RANSAC

Selon le type des points matchés (2D-3D, 3D-3D, 3D-2D) [Scaramuzza and Fraundorfer, 2011], la nature de la scène (coplanaire, 3D, etc), les contraintes sur le mouvement, et selon si la caméra est calibrée ou non, il existe plusieurs solveurs pour estimer la transformation entre deux nuages de points d'intérêt :
Solveur 8 points :

— Pour les caméras calibrées et non calibrées.
— Échoue lorsque les points dans la scène sont coplanaires.

Solveur 5 points :

— Fonctionne pour les scènes coplanaires.

— Juste pour les caméras calibrées.

Solveur 3 points :

— Dans le cas où l'orientation entre les deux images est connue. Ceci peut être le cas, si la caméra est couplée à un capteur inertiel.

Solveur 2 points :

— Utilise la matrice de rotation complète d'une IMU attachée.

— Utilisable si le mouvement est planaire (modèle de mouvement à trois degrés de liberté).

Solveur 1 point :

— Pour un mobile à roues, le mouvement pouvant être décrit comme planaire et circulaire (modèle de mouvement à deux degrés de liberté).

La structure de l'estimateur de pose dans la librairie libviso2 [Geiger et al., 2011], donne un exemple de l'utilisation de l'algorithme de 8 points pour l'estimation de pose. Dans cette librairie l'estimateur RANSAC (RANdom SAmple Consensus) est utilisé pour trouver la matrice fondamentale, qui est l'élément essentiel dans l'estimation de pose par RANSAC. L'ambiguïté de l'échelle que toutes les techniques VO contiennent peut être enlevée par la connaissance a priori de la hauteur de la caméra par rapport au sol. La Fig.1.6 montre les éléments essentiels d'un algorithme basé RANSAC avec l'estimation obtenue lors de nos tests.

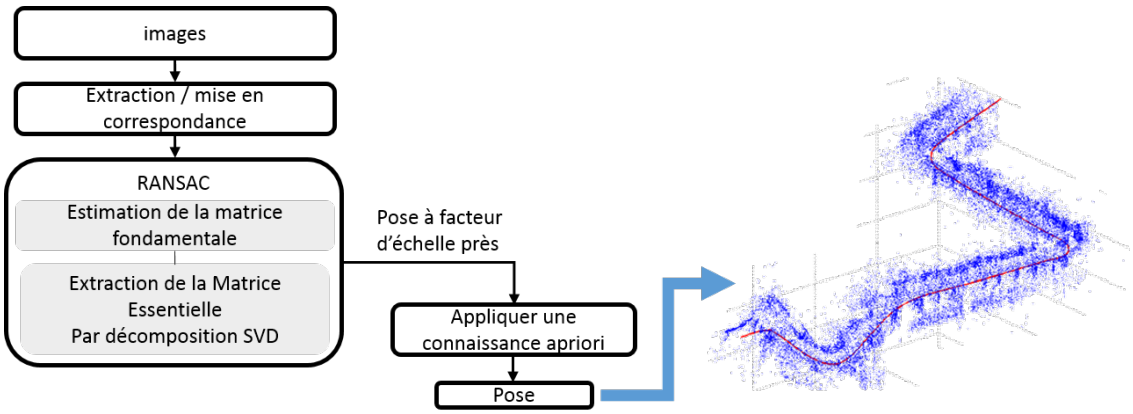


FIGURE 1.6 – Résultat de localisation basée RANSAC.

Lors des expérimentations effectuées en laboratoire, nous avons constaté que beaucoup de ces algorithmes de VO souffrent d'un manque de robustesse apparent et qui provoque souvent d'importantes dérives de l'odométrie. L'une des difficultés majeures dans cette technique, est qu'une fois l'échelle perdue à cause d'une perte de points d'intérêt par exemple, il est très difficile de la retrouver (l'échelle est généralement estimée à l'initialisation et propagée dans les nouvelles images).

1.2.3.2 Estimation par optimisation

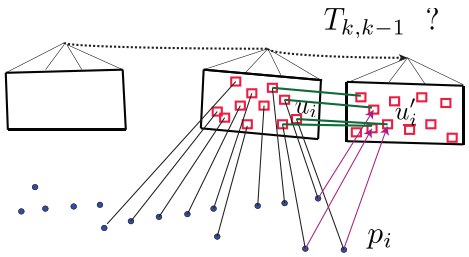


FIGURE 1.7 – Reprojection des points d'intérêt dans un algorithme d'optimisation.

Cette méthode est basée sur la minimisation de l'erreur de re-projection des points d'intérêt [Forster et al., 2014][Badino and Kanade, 2011]. Ce type d'algorithmes donne des résultats précis mais couteux en temps de calcul en comparaison de ces mêmes techniques mais avec un Filtrage de Kalman Étendu. Le critère à minimiser est souvent défini dans la forme suivante :

$$T_{k,k-1} = \arg\max_{T} \sum_{i} \parallel \mathbf{u}'_i - \pi(p_i) \parallel^2 \tag{1.1}$$

Avec $\mathbf{u}'_i$ la position d'un ensemble de points d'intérêt $(u', v')_i$ dans l'image courante k, et $\pi(p_i)$ est la position reprojetée du point correspondant $\mathbf{u}_i$ de l'image précédente $k-1$ dans l'image courante Fig.1.7.

1.3 Localisation et cartographie

Dans l'état de l'art, certaines approches de localisation permettent également de construire une carte de l'environnement. Ceci est réalisé en paralèlle à la localisation. Ce problème est souvent appelé SLAM "Simultaneous Localization and Mapping" [Mur-Artal et al., 2015b][Zhang and Koch, 2011].

1.3.1 Modèles de cartes

Les algorithmes de localisation et surtout ceux basés vision permettent souvent de construire une carte de l'environnement. Ils utilisent différents référentiels de cartes pour représenter des informations, dont les caractéristiques sont généralement dépendantes de la nature de l'application. Dans la majorité des applications de robotique mobile ou aérienne, les cartes de représentation de l'environnement sont d'une importance fondamentale. On peut distinguer deux types d'applications. Le premier type inclut des applications nécessitant une connaissance a priori de la carte pour réaliser des tâches comme la navigation vers un but ou l'interaction avec des objets [Jeon and Lee, 2016].

L'autre type impose aux robots d'être en mesure de construire une carte et donc exige plus d'autonomie et de capacité d'adaptation. Ceci est le cas pour les missions de search and rescue [Cui et al., 2015]. L'objectif principal d'une carte est de donner une référence à des données obtenues à partir d'un capteur. Construire une carte de représentation de l'environnement revient à gérer les incertitudes de la pose du robot et de l'observation, la taille de la carte, sa résolution et les problèmes d'aliasing (des endroits différents mais avec une apparence très proche). On distingue :

Les cartes de points d'intérêt : ce type de cartes est le plus répandu dans les algorithmes de localisation et de cartographie basés vision. Cette carte contient la position 3D des amers utilisés dans l'estimation de pose du robot. Elle contient souvent, en plus de la position des amers, leurs descripteurs qui permettent, au cas où le robot repasse par le même endroit de la carte, de comparer ces derniers avec ceux observés pour déterminer les fermetures de boucles. Cette fermeture permet de réarranger la carte et la pose et ainsi de corriger la dérive odométrique.

Les cartes topologiques : une carte topologique est un graphe de poses dans lequel les nœuds enregistrent des informations correspondant à des endroits spécifiques de l'environnement, et les arêtes sont généralement des transformations qui lient un nœud à un autre. Pratiquement, chaque nœud enregistre les mesures capteurs d'un lieu, avec la position de ce nœud dans la carte. Les arêtes contiennent le coût de passage d'un nœud à un autre, ce qui rend cette représentation particulièrement utile pour les algorithmes de parcours de graphe comme les algorithmes A* et Dijkstra [He and Zhao, 2017], qui permettent de trouver le plus court chemin entre deux nœuds. L'une des problématiques de cette représentation est le choix du moment de création d'un nœud. L'une des solutions peut être de créer des nœuds à intervalles réguliers pour un robot mobile, mais ce choix peut pour certains environnements, créer des nœuds qui contiennent des informations redondantes, du fait que l'environnement n'as pas trop changé. Ce problème nous amène à la création de nœuds plutôt basée sur la quantité de changements dans l'observation pour éviter d'avoir trop de nœuds dans le graphe. Ce choix résout un problème, mais fait surgir un autre, car dans ce cas, un environnement comme un couloir sera perçu par un laser comme un seul endroit représenté par un seul nœud, ce qui n'est pas précis pour la réalisation d'une fermeture de boucle.

Les cartes d'occupation : si l'environnement d'évolution du robot peut être considéré planaire, il peut être représenté par une grille 2D. Sur cette carte, une région est partitionnée en plusieurs cellules dont la taille dépend de la résolution et du degré de précision que l'utilisateur veut donner à la carte. Ces cartes représentent généralement l'espace navigable, ce qui les rend utiles pour la planification de trajectoires et la navigation autonome. Une représentation 3D est aussi possible grâce à des cubes appelés "voxels". La difficulté se situe particulièrement dans l'association des données qui est faite

ici avec une cross-corrélation sur la région de pose du robot. Un autre problème est la résolution de la carte qui est liée directement à la complexité de calcul de l'algorithme.

1.3.2 SLAM - Simultaneous Localization and Mapping

La catégorie des algorithmes SLAM est la plus étudiée lorsqu'il s'agit de localisation visuelle. Cette architecture, contrairement aux algorithmes de localisation par VO, donne une grande importance à la construction de la carte. Ces algorithmes contiennent deux processus liés et dépendants l'un de l'autre : le processus de localisation et le processus de cartographie. La Fig.1.8 ilustre le principe de fonctionnement d'un SLAM [Mur-Artal et al., 2015b] [Engel and Cremers, 2014].

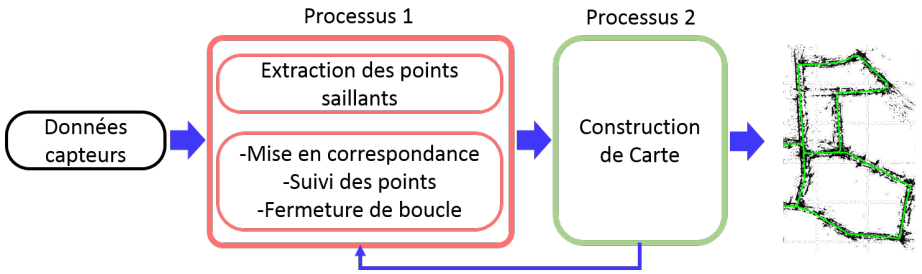


FIGURE 1.8 – Localisation et cartographie simultanées.

Le processus 1 est souvent construit par des techniques similaires à celles déjà citées dans la partie VO. Le processus 2 contient deux étapes qui ne sont généralement pas intégrées dans les algorithmes VO, puisqu'on ne donne pas la même importance à la construction de carte. Ces étapes sont l'ajustement de faisceaux (BA) et l'optimisation des poses de la caméra, qui sont des étapes qui minimisent l'erreur de reprojection de tous les points dans toutes les images de référence. Ce processus est très couteux en temps de calcul, ce qui contraint souvent son utilisation aux N dernières images de référence. Il est important de mentionner ici que cette optimisation de carte permet de garantir la consistance globale de l'estimation, chose qu'on ne trouve pas dans un algorithme de localisation sans cartographie.

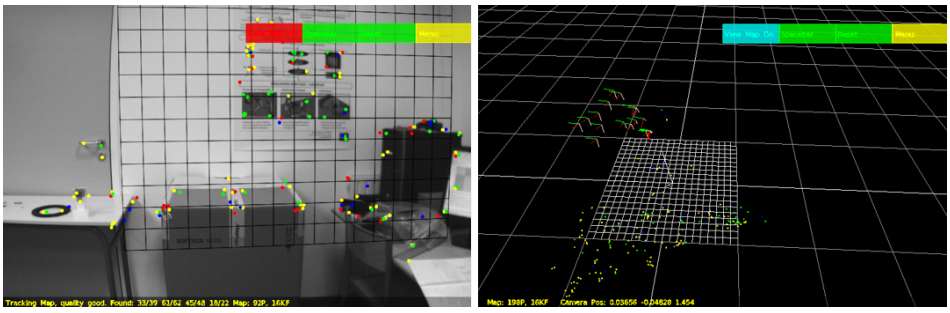


FIGURE 1.9 – Interface de l'algorithme de localisation et cartographie PTAM.

Récemment, le ORB-SLAM [Mur-Artal et al., 2015b] a été largement utilisé pour la localisation et la cartographie, avec des résultats en cartographie impressionnants. Le PTAM (Parallel Tracking and Mapping) est un autre algorithme, plus ancien, très connu pour la localisation dans des petits espaces de travail, Fig.1.9.
Le SLAM est aussi souvent implémenté en stéréovision [Engel et al., 2015], que ce soit en méthode directe, semi directe ou basée points d'intérêt. La stéréovision permet un accès direct à la profondeur grâce à la transformation connue entre les deux caméras. Ceci permet de résoudre le problème d'ambiguïté de l'échelle que les VO monoculaires contiennent.
Notons que le SLAM combiné à une paire stéréo est l'une des rares techniques d'odométrie visuelle à être commercialisée par plusieurs fabricants Fig.1.10. On peut citer, la caméra ZED, DUO MLX, Orsens, etc. L'inconvénient du capteur stéréographique relativement à un capteur monoculaire est son encombrement, qui limite souvent son utilisation aux drones de taille moyenne et grande.

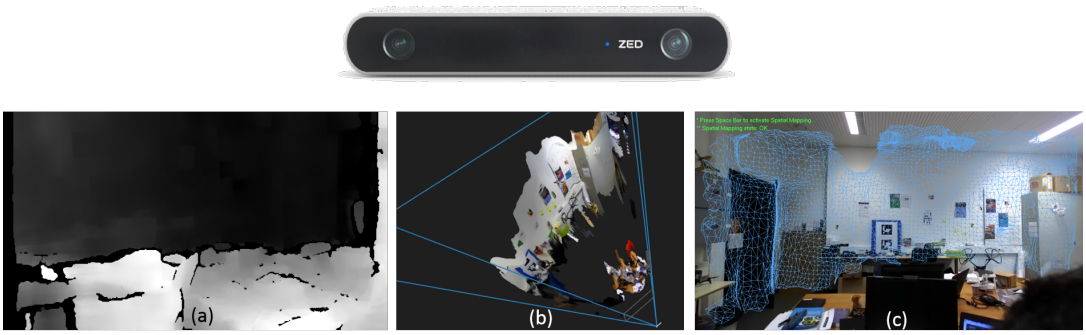


FIGURE 1.10 – Capacité temps réel des solutions stéréo. (a) Carte de disparité, (b) et (c) reconstruction 3D de l'environnement.

1.3.3 Ajustement de faisceaux

Le bundle adjustment (BA) est la solution optimale de ce qui est appelé le problème de localisation et cartographie complet [Özyesil et al., 2017]. Dans ce problème, on recherche une solution maximale de vraisemblance des observations en utilisant toutes les mesures sur toute l'évolution du robot. L'objectif dans le BA est de minimiser l'erreur entre l'observation prédite et les mesures images de n points 3D observés à partir de m capteurs 6 dof. Les mesures et les paramètres à estimer sont considérés des distributions normales et le problème est souvent résolu avec des techniques de moindres carrés non linéaires comme la méthode de Gauss Newton.

La matrice linéarisée qui apparait dans la solution de ce problème correspond à la matrice d'information de Fisher. Elle correspond à celle utilisée pour définir la

FIGURE 1.11 – Reconstruction par ajustement de faisceaux.

bande inférieure de Cramer Rao qui permet de juger la consistance et l'optimalité des estimateurs. Cela explique pourquoi le BA est un algorithme optimal de localisation et cartographie. La technique BA est aussi importante dans des problèmes comme la "structure from motion" Sfm [Özyesil et al., 2017] où on s'intéresse souvent à la reconstruction de scène à partir d'une caméra en mouvement. Dans le contexte de localisation, cette technique donne de meilleurs résultats que les techniques EKF équivalentes, mais elle est couteuse en temps de calcul vu le nombre de paramètres à estimer, surtout dans le contexte d'une fermeture de boucle. Pour réduire cette complexité calculatoire, plusieurs variantes proposent d'appliquer un BA limité à une fenêtre de N poses au lieu de l'intégralité des poses.

Notons aussi que certains algorithmes de VO [Geiger et al., 2011] (Annexe B) utilisent une minimisation Gauss Newton sur les dernières observations pour trianguler les points d'intérêt. La position 3D de ces derniers étant essentielle pour l'estimation de pose dans certains de ces algorithmes.

1.4 Odométrie Visuelle Inertielle

Dans le reste de ce chapitre, nous commencerons par aborder l'état de l'art autour de la fusion de données caméra IMU pour finir ensuite avec la fusion inter-robots que l'on proposera dans les chapitres suivants.

Pour une meilleure robustesse de l'odométrie visuelle, plusieurs travaux utilisent un second capteur, qui a pour objectif de combler le manque d'information que peut avoir une odométrie visuelle seule. L'un des capteurs les plus souvent fusionnés avec un capteur principal d'odométrie visuelle est l'IMU. Notre objectif est de proposer une architecture de fusion caméra IMU, idéale pour un futur passage au cadre multi où notre objectif sera de faire collaborer plusieurs robots en fusionnant leurs données.

Les systèmes hybrides comme les VINS (Visual Inertiel Navigation Systems) sont devenus des techniques de plus en plus utilisées pour résoudre le problème de la

localisation. La combinaison des capteurs vision et inertiel résulte en une odométrie beaucoup plus robuste et précise que ce que l'on peut constater sur les systèmes VO seuls. Ces capteurs permettent de réduire significativement la dérive de l'odométrie visuelle.

1.4.1 Capteur inertiel

Une centrale inertielle (IMU) est un composant qui contient un accéléromètre, un gyroscope et parfois un magnétomètre. Ceci permet d'avoir une mesure des accélérations linéaires, des vitesses angulaires, et si un magnétomètre est intégré, une mesure d'intensité du champ magnétique sur 3 axes. L'énorme progrès dans les technologies MEMS (Micro-Electro-Mechanical-Systems) en terme de miniaturisation a fait que les IMU sont aujourd'hui présents dans un large panel d'applications smartphone, drone, réalité virtuelle, etc.

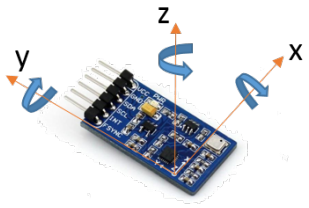


FIGURE 1.12 – Exemple de centrale inertielle.

Le capteur inertiel dans une application d'odométrie permet, grâce à sa fréquence d'acquisition qui peut atteindre 1 khz, de combler le vide d'observation que les mesures d'autres capteurs à basse fréquence vont avoir. De plus, du fait de sa capacité à mesurer la gravité, il permet l'initialisation de certaines grandeurs à estimer (example, l'orientation). L'utilité de la fusion caméra IMU dans la localisation se fait sentir surtout dans le cadre de la localisation de drones en vol intérieur, où la fréquence de mise à jour de la pose doit être élevée pour répondre aux exigences de ce système de haute dynamique.

1.4.1.1 Accéléromètre

L'accéléromètre dans une centrale inertielle, mesure l'accélération d'une masse au repos dans le repère de référence du capteur. Ce capteur au repos sur la surface de la Terre mesure une accélération de $9.81 m/s^2$ qui correspond à l'accélération de la gravité. Le principe physique utilisé dans ce capteur est simple (voir Fig 1.13). Une masse m généralement appelée Seismic mass ou Proof mass est supportée par un ressort C. Un amortisseur visqueux b, permet un amortissement relatif à la vitesse du corps du capteur et à la masse de test. Lorsque le capteur est soumis à une accélération, la masse est

déplacée et ce déplacement est mesuré pour obtenir l'accélération. L'équation (1.2) montre la dynamique du système :

$$\ddot{x}(t) = 2\xi\omega_n\dot{x}(t) + \omega_n^2 x(t) = -\ddot{y}(t) \tag{1.2}$$

où l'accélération du corps du capteur $\ddot{y}(t)$ est convertie en déplacement $x(t)$ avec une fréquence naturelle ω_n et un facteur d'amortissement ξ.

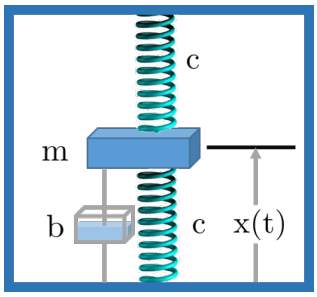


FIGURE 1.13 – Principe de fonctionnement d'un accéléromètre.

Il faut prendre en compte le fait que ce capteur a besoin de plus d'information ou de supposition pour pouvoir faire la différence entre une mesure provoquée par un déplacement du capteur et une mesure due à la présence de l'accélération de la gravité.

1.4.1.2 Gyroscope

Il existe deux grandes catégories de gyroscopes, mécanique et optique. Dans les mécaniques, on retrouve généralement un système constitué d'une roue tournante ou d'un disque sur un axe. Une fois le disque en rotation, il crée une résistance au changement de rotation Fig(1.14). Les gyroscopes optiques utilisent un procédé où deux faisceaux laser sont envoyés dans des fibres optiques performant deux rotations opposées. Selon l'effet Sagnac, le faisceau navigant dans la direction du sens de rotation aura une fréquence plus élevée. La différence entre les deux faisceaux est proportionnelle à la vitesse angulaire. Ce type de gyroscope donne une très grande précision, il est insensible aux chocs et aux vibrations.

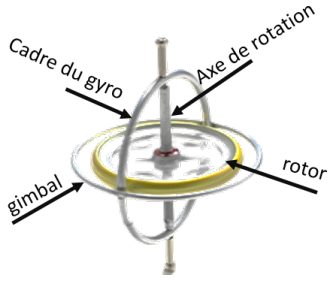


FIGURE 1.14 – Gyroscope.

1.4.2 Couplage caméra / IMU

Utiliser un système VINS offre deux possibilités pour réaliser une odométrie : une fusion en couplage serré ou une fusion en couplage lâche Fig.1.15.

Dans un couplage lâche [Lynen et al., 2013] [Weiss et al., 2013] [Weiss et al., 2012], les données IMU sont fusionnées directement avec une estimation de pose déterminée par une information visuelle. Généralement, cette estimation est connue à un facteur d'échelle près. Elle est obtenue par des algorithmes connus comme PTAM [Klein and Murray, 2007], l'ORB SLAM [Mur-Artal et al., 2015b] ou n'importe quel autre estimateur basé vision.

Par contre, dans un couplage serré [Mourikis and Roumeliotis, 2007] [Guo et al., 2014] [Hu and Chen, 2014] [Leutenegger et al., 2013], les données IMU sont directement fusionnées avec les données brutes de la caméra (les points d'intérêt). Dans cette approche, la majorité des algorithmes utilisent une fusion de données basée sur un EKF ou une optimisation non linéaire. Les versions EKF [Mourikis and Roumeliotis, 2007] [Guo et al., 2014] [Hu and Chen, 2014] sont les plus présentes dans la littérature du fait de leur complexité calculatoire, beaucoup moindre relativement à celles basées optimisation [Leutenegger et al., 2013].

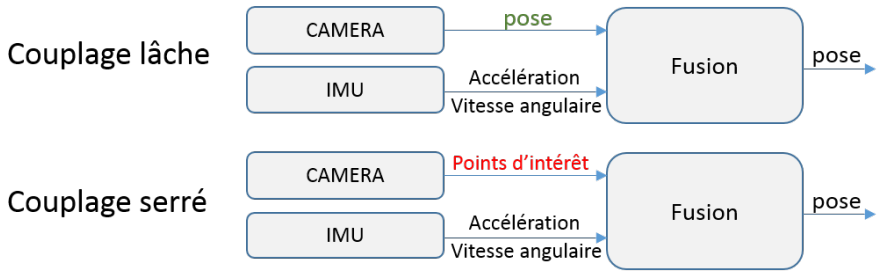


FIGURE 1.15 – Les types de couplage caméra IMU.

La fusion par couplage lâche est beaucoup plus facile et pratique à implémenter que le couplage serré. L'inconvénient cependant est que le bloc basé vision (boite noire) garde exactement les mêmes propriétés qu'avant la fusion lâche avec l'IMU, et donc exactement les mêmes faiblesses. Cette observation est la raison qui a motivé notre choix de l'approche couplage serré basé filtrage de Kalman. On peut distinguer deux familles d'algorithmes pour la fusion en couplage serré avec un Kalman, les algorithmes MSCKF [Mourikis and Roumeliotis, 2007] et les algorithmes EKF-SLAM [Davison et al., 2007]. La différence entre ces deux familles ainsi que les raisons qui nous ont orienté vers l'adoption de la structure MSCKF dans notre travail sont présentées dans l'introduction du chapitre suivant.

1.5 Localisation collaborative

De nombreuses techniques existent pour permettre à un groupe d'agents d'estimer leur poses. L'objectif est souvent d'utiliser les données de plusieurs agents pour permettre soit une meilleure localisation de ces agents, soit l'exploitation des données et des observations relatives pour les besoins d'une mission. Dans la localisation collaborative, presque chaque étape constituant cette problématique est sujette à la recherche scientifique. Le domaine des « réseaux de capteurs » est la thématique la plus évoluée en nombre d'algorithmes et de techniques dans cette sphère de collaboration. A la différence des réseaux de capteurs, où généralement les algorithmes se concentrent sur des mesures de distance entre nœuds, les algorithmes en robotique exploitent majoritairement la vision dans la collaboration, même si jusqu'à présent, l'utilisation de ce capteur reste très couteuse en temps de calcul pour les agents à capacité de calcul limitée. Les thématiques relatives à la localisation collaborative sont très nombreuses et couvrent des domaines très vastes. Il est néanmoins possible de les regrouper dans certaines catégories en fonction de la nature centralisée ou décentralisée de traitement des données relatives, de l'hétérogénéité des données (et des robots), du type de données relatives exploitées et des approches de fusion de données utilisées. Ces variations font que le sujet de la localisation collaborative est très complexe et contient de nombreuses bifurcations.

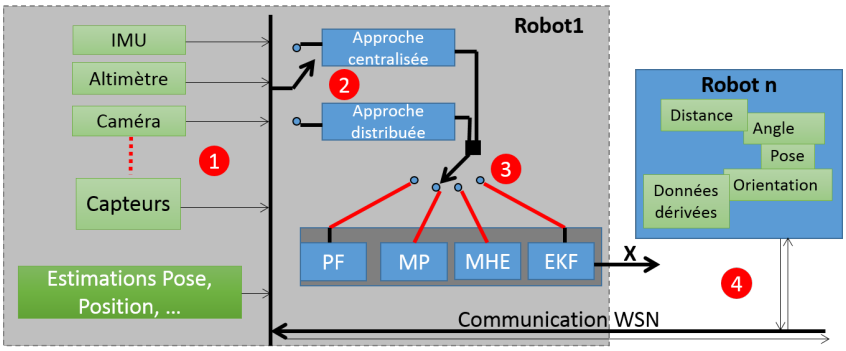


FIGURE 1.16 – Structure globale d'une localisation collaborative.

La Fig.1.16 montre par des cercles rouges les différents choix nécessaires dans le processus de construction d'une localisation collaborative.

1.5.1 Algorithmes de localisation collaborative

La localisation collaborative est souvent une extension de la localisation individuelle. Ceci implique que les algorithmes les plus sujets à des tentatives d'extension au collaboratif sont les techniques les plus utilisées dans la communauté pour la localisation individuelle des robots. Dans le cas de la localisation basée vision, les approches SLAM prédominent,

et on peut trouver plusieurs propositions d'extention vers le multi [Karrer et al., 2018] [Zou and Tan, 2013].

La majorité des algorithmes utilisent essentiellement trois filtres pour fusionner les données en localisation collaborative : le filtre de Kalman étendu (EKF) [Huang and al, 2011] [Achtelik et al., 2011] [Piasco et al., 2016], le moving horizon estimation (MHE) [Wang et al., 2014][Wang et al., 2016] et le maximum apostériori (MAP) [Nerurkar et al., 2009a]. Chacun de ces filtres présente des avantages et des inconvénients. Pour le filtre EKF, l'avantage est la rapidité et la quantité importante d'algorithmes déjà existants qui traitent ce sujet. Les inconvénients sont la précision moindre et la complexité qui peut surgir si l'on souhaite utiliser des modèles d'observation complexes, rendant les Jacobiennes pas toujours faciles à obtenir. L'autre problème de ce type de fusion de données est la nécessité d'une bonne initialisation, chose qui n'est pas toujours possible à avoir.

Pour le MHE, on a une meilleure estimation que l'EKF car il est basé sur une optimisation, avec en plus l'avantage d'être facile à augmenter avec des contraintes sur l'état. Il est cependant plus lent qu'une fusion EKF.

Quant au maximum apostériori, certains travaux comme [Nerurkar et al., 2009b] le proposent comme une solution pour réduire la complexité calculatoire dans des algorithmes de collaboration décentralisés.

1.5.2 Choix des capteurs

Si on prend le problème de la localisation collaborative dans sa globalité, on trouve que le choix des capteurs à utiliser commence d'abord par le choix des capteurs responsables de l'estimation de la pose du robot lui-même (localisation individuelle). En effet, l'estimation d'un robot de sa propre pose impactera directement la collaboration de ce dernier avec les autres robots. Il s'agit donc de choisir une stratégie de localisation individuelle, qui soit cohérente avec l'objectif final qui est la localisation collaborative. Une fois ce choix réalisé (pour nous, on a opté pour une fusion caméra IMU), vient donc le tour de choisir les données à échanger pour collaborer. Ce choix est l'un des grands sujets de publications dans cette thématique. En effet, deux robots peuvent, s'ils sont équipés par les capteurs nécessaires, échanger : leur distances relatives [Cornejo and Nagpal, 2014], leur angles d'observation relative (bearing), la position relative, l'orientation ou la pose relative [Achtelik et al., 2011]. D'autres types de données dérivées d'un capteur peuvent être échangées. Comme par exemple, dans le cas du capteur caméra, où au lieu d'échanger des images, des descripteurs de points d'intérêt ou autres données dérivées peuvent être échangés, pour des raisons de bande passante dans la communication. Dans certains travaux [Knuth and Barooah, 2015], sont évalués

l'efficacité et l'impact du type de données échangées dans la localisation collaborative. Il est stipulé [Knuth and Barooah, 2015] que la meilleure information à échanger est la pose relative. Cette information est aussi la donnée la plus difficile à extraire puisqu'il n'y a pas de capteurs permettant de directement l'obtenir. Elle est généralement estimée par traitement des données caméra. En deuxième position sur la grille d'efficacité des données échangées en localisation collaborative, viennent les données d'orientation et de position relatives. Là encore, en particulier pour l'information d'orientation, il n'est pas facile de l'avoir facilement (bien sûr, certains scénarios le permettent comme un drone survolant un robot mobile [Falanga et al., 2017]). Viennent après les données d'angle et enfin de distance. Les données de distance sont jugées les moins efficaces dans une localisation collaborative, mais elles sont aussi les plus faciles à obtenir grâce à des capteurs faciles à intégrer sur n'importe quel type de cible et à un coût peu élevé (moins de 30 euros par robot).

1.5.3 Positionnement par ULB

Dans la section précédente, on a classé le type de données de collaboration par efficacité. Dans ce travail, on s'orientera vers une proposition d'une collaboration basée sur une complémentarité entre les données de la caméra et du capteur ULB (Ultra Large Bande).

L'ULB est l'une des technologies les plus récentes, précises et prometteuses [Ghavami et al., 2007] dans le domaine de la localisation par signaux radio. Le département de la Défense des États-Unis a été le premier à utiliser le terme Ultra Large Bande. Il est devenu disponible dans le commerce à la fin des années 1990 [Ghavami et al., 2007] mais avec une régulation de son utilisation pour éviter l'interférence avec d'autres réseaux déjà présents sur l'espace réseau personnel.

L'ULB est basé sur la transmission d'impulsions extrêmement courtes et utilise des techniques qui provoquent une propagation de l'énergie radio sur une large bande de fréquences et avec une densité spectrale de puissance très faible [Ghavami et al., 2007]. Cette technologie peut fournir une communication à haut débit qui peut atteindre 100 Méga bits par seconde. C'est une solution idéale pour la transmission de données à courte distance.

La basse fréquence des ULB permet au signal de traverser les obstacles comme les murs ou les objets, ce qui leur confère un intérêt et une attractivité dans les applications de localisation robotique.

Il est possible de classer l'utilisation de ces capteurs dans quatre groupes fonctionnels, la communication, le positionnement, le tracking et enfin l'utilisation dans le contexte de radar. Cette dernière utilisation a connu une grande montée avec les nouvelles puces

Decawave qui ont permis de construire des radars de taille très compacte [dec,].
Lorsque cette technologie est utilisée en application de localisation, le fait que la bande passante soit élevée et que la forme d'ondes soit extrêmement courte, permet de minimiser l'effet des multi-trajets dans le calcul du TOA (time of arrival). A cet avantage, s'ajoute le fait que contrairement aux ultrasons et aux lasers, l'ULB n'as pas forcément besoin d'une ligne de vue directe entre le transmetteur et le récepteur pour fonctionner. Ceci rend cette technologie plus efficace et beaucoup appréciée dans le domaine de la localisation.

1.5.3.1 Algorithmes de localisation ULB

Dans les cinq dernières années, plusieurs solutions commerciales de localisation basées ULB ont vu le jour. Ceci s'explique par le fait que cette technologie offre une solution intéressante pour la localisation en intérieur. Il existe de nombreux algorithmes de positionnement par ULB, qui peuvent être classés en cinq catégories principales : (1) Time Of Arrival (TOA) ; (2) Angle Of Arrival (AOA) ; (3) Received Signal Strength (RSS) ; (4) Time Difference Of Arrival (TDOA) ; et (5) les algorithmes hybrides.

Ces algorithmes sont souvent basés sur une infrastructure fixe, ce qui n'est pas le but de notre travail. Les techniques et les faiblesses que l'on peut rencontrer dans ces algorithmes sont par contre, les mêmes que l'on rencontre en utilisant des capteurs ULB portés par les robots.

1.5.3.1.1 Algorithmes AOA Dans cette technique, pour réaliser une localisation, deux mesures de l'angle de réception du signal et la distance entre les deux récepteurs sont utilisées (voir figure 1.17). Cette technique est très sensible à plusieurs facteurs, ce qui peut causer des erreurs dans l'estimation de la position du nœud à localiser. La géométrie des antennes de réception du signal a un impact majeur sur l'estimation [Al-Jazzar et al., 2011]. L'augmentation de la distance entre l'émetteur et le récepteur peut affecter la précision [Reddy et al., 2011]. À cela s'ajoute le fait que cette technique a une complexité supérieure aux autres.

1.5.3.1.2 Algorithmes TOA Cette technique est basée sur l'intersection des cercles de plusieurs transmetteurs dont le rayon représente la distance entre l'émetteur et le récepteur. Pour estimer cette distance, on calcule le temps de propagation entre les deux. Dans cette estimation, il est souvent inclus (modélisé) tout ce qui peut affecter la propagation pour avoir la meilleure estimation.

Les TOA sont très étudiés. On peut citer [Dardari et al., 2009] qui donne une vue globale sur les techniques de localisation par ULB et les sources d'erreurs qui peuvent influencer les estimations. Les auteurs dans [Segura et al., 2012] [Kok et al., 2015] utilisent cette approche pour une localisation dans un environnement intérieur. Cette technique

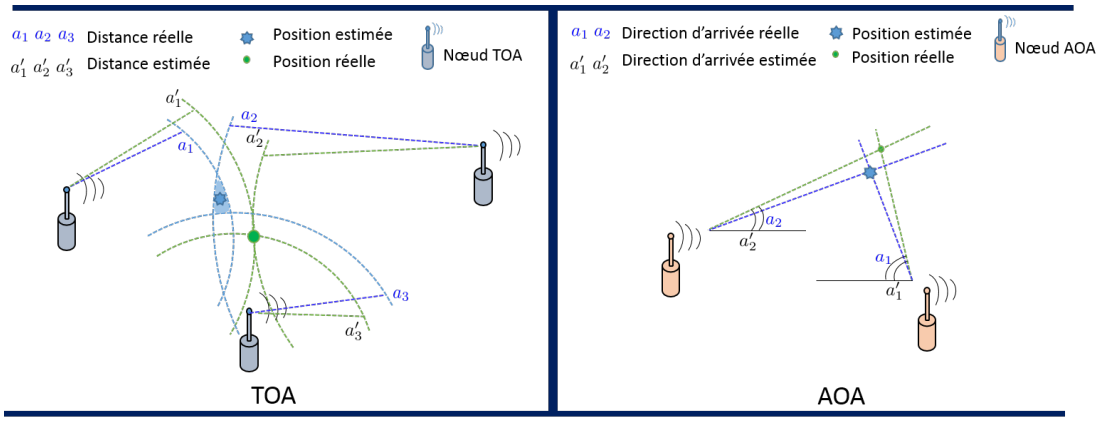


FIGURE 1.17 – Principe des algorithmes AOA et TOA.

est proche de ce qui va être proposé dans notre travail. Le capteur Trek1000 adopté dans notre travail, utilise notamment cette technique pour localiser un mobile en estimant les temps d'arrivée des signaux.

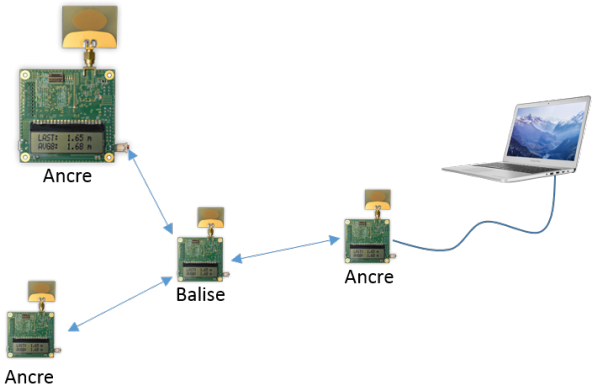


FIGURE 1.18 – Kit de balises ULB Trek1000.

1.5.3.1.3 Algorithmes TDOA Cette technique est basée sur le calcul de différence de temps d'arrivée d'un signal envoyé par un transmetteur (qui est dans ce cas le nœud à localiser) aux récepteurs. Contrairement aux autres techniques, elle nécessite une collaboration importante des récepteurs, ce qui signifie qu'elle nécessite plus de bande passante. Tout comme pour les autres techniques, plusieurs travaux se sont concentrés sur la localisation en intérieur [Krishnan et al., 2007], l'utilisation des plans des bâtiments pour améliorer la localisation [Leitinger et al., 2015] et l'atténuation des effets des multi-trajets.

1.5.3.1.4 Algorithmes RSS Dans cet algorithme, l'objet à localiser estime la puissance du signal reçu provenant de plusieurs transmetteurs, pour déterminer sa distance par rapport à chacun d'entre eux. En se référant à [Pittet et al., 2008], la qualité des

estimations réalisées avec cet algorithme dans des conditions de NLOS (Non Line Of Sight) et dans des environnements de fort multi-trajets, est très mauvaise. Cette méthode est un mauvais choix pour la localisation à l'intérieur de bâtiments. Elle est souvent utilisée combinée avec les autres techniques.

1.6 Conclusion

Dans la première partie de ce chapitre, un état de l'art des techniques d'odométrie visuelle a été présenté. Nous y avons introduit plusieurs éléments incontournables dans la construction d'un algorithme de localisation basé vision, en mettant l'accent sur les techniques VO monoculaires basées points d'intérêt qui sont adoptées dans notre travail.

Cette étude nous aiguille loin des techniques basées optimisation à cause du coût élevé en temps de calcul, qui n'est pas compatible avec notre objectif d'extension vers les systèmes multi hétérogènes. Pour la même raison, on se trouve aussi orienté plutôt vers les solutions VO que SLAM.

Dans la deuxième partie de ce chapitre, nous avons présenté un état de l'art de la fusion de données caméra IMU et la fusion de données entre robots dans le cas d'une localisation collaborative. Notre analyse et tests expérimentaux de quelques travaux de l'état de l'art sur la fusion caméra IMU nous orientent plutôt vers des techniques de fusion basées filtrage de Kalman à cause de leur efficacités en temps de calcul ; ce qui est d'une grande importance pour l'extension vers le multi. L'autre conclusion de notre étude nous conduit à favoriser les approches de fusion à couplage serré (MSCKF), ce qui permettra à la localisation individuelle de gagner en robustesse.

Dans l'état de l'art sur la localisation collaborative, nous avons introduit différents choix nécessaires dans la réalisation d'une collaboration. Nous avons discuté de la nature des données échangées, de leur efficacité et des algorithmes de fusion utilisés dans l'état de l'art. A la fin de cette partie, nous avons orienté l'étude vers les ULB qui font partie de notre solution pour la localisation collaborative.

Dans les chapitres qui suivent, nous commencerons par proposer une première contribution qui est une architecture de localisation individuelle adaptée à notre objectif d'extension vers le collaboratif. Une deuxième contribution représentant une solution de localisation collaborative sera ensuite détaillée.

Chapitre 2 :

Localisation d'un robot par odométrie visuelle inertielle

2.1 Introduction

Les systèmes hybrides comme les systèmes d'odométrie visuelle inertielle (VINS) sont devenus très utilisés pour résoudre le problème de la localisation. La combinaison de capteurs de vision (caméra monoculaire) et de capteurs inertiels (IMU) résulte en une odométrie avec moins de problèmes de robustesse que dans la navigation par caméra uniquement, et une dérive significativement réduite par rapport à l'odométrie de l'IMU seule. Cela fait des systèmes d'odométrie visuelle inertielle l'une des solutions émergentes dans l'estimation de pose pour des applications nécessitant à la fois la précision et la rapidité des estimations.

Des études et des implémentations récentes ont montré que l'odométrie visuelle inertielle est capable de diminuer les erreurs d'estimation jusqu'à 1% de la distance parcourue, ce qui rend cette combinaison de capteurs très attrayante en robotique mobile. Cet aspect s'est traduit pas un nombre important de publications sur ce sujet. Cela ouvre également des questions sur la possibilité d'amélioration des solutions de l'état de l'art, avec le maintien de temps de calcul faible. En particulier, lorsqu'on remarque que dans ces implémentations, la technique de fusion de données généralement employée est basée sur le filtre de Kalman Étendu (EKF), que l'on sait très efficace en temps de calcul mais pas optimal.

Ce chapitre détaille notre approche de fusion de données entre la caméra monoculaire et l'IMU avec l'objectif d'étudier les performances d'un système VINS formulé sous la forme de fusion MSCKF (Multi-State Constrained Kalman Filter). Les critères considérés sont la précision, la robustesse et le temps de calcul des estimations de localisation dans différents scénarios. Cette étude permettra de définir la meilleure architecture du filtre pour l'estimation précise de pose adaptée au cas collaboratif.

2.2 Fusion vision - centrale inertielle

Dans ce travail, la fusion de données sera exprimée en couplage serré [Mourikis and Roumeliotis, 2007] [Hu and Chen, 2014] [Guo et al., 2014], où les données IMU sont directement fusionnées avec les points d'intérêt extraits des images, au lieu d'être fusionnées avec une estimation de pose à un facteur d'échelle comme dans la fusion en couplage lâche [Lynen et al., 2013] [Weiss et al., 2012]. Comme indiqué dans [Santoso et al., 2017], le principal inconvénient de la fusion en couplage lâche, comparée au

couplage serré, est l'effet de découplage, qui entraîne une perte d'information et engendre des problèmes de robustesse.

Dans la fusion des données visuelles-inertielles en couplage serré utilisant un filtre de Kalman Étendue, deux approches existent dans la littérature. L'approche EKF-SLAM, dans laquelle le vecteur d'état contient l'état actuel ainsi que les positions des points d'intérêt [Piniés and Tardós, 2007] [Davison et al., 2007]. La seconde approche est le filtre Multi-State Constraint Kalman Filter (MSCKF) [Mourikis and Roumeliotis, 2007] [Hu and Chen, 2014] [Guo et al., 2014] qui maintient un vecteur d'état composé de l'état actuel et d'une fenêtre de m poses précédentes du système (au lieu des positions des points d'intérêt). Dans ce cas, comme les positions des points d'intérêt ne sont pas incluses dans le vecteur d'état, aucune erreur de linéarisation supplémentaire n'est introduite. Le MSCKF est le cadre général de fusion que nous avons adopté dans notre approche où on se posera notamment des questions de définition des modules pour ce filtre et de leurs performances.

2.3 Principe du Multi-State Constraint Kalman Filter

Le MSCKF effectue une fusion en couplage serré sur une fenêtre glissante de m poses de la caméra en utilisant toutes les observations disponibles dans cette fenêtre comme contraintes sur les poses. Dans cette section, nous détaillons la formulation de cette fusion des données visuelles inertielles.

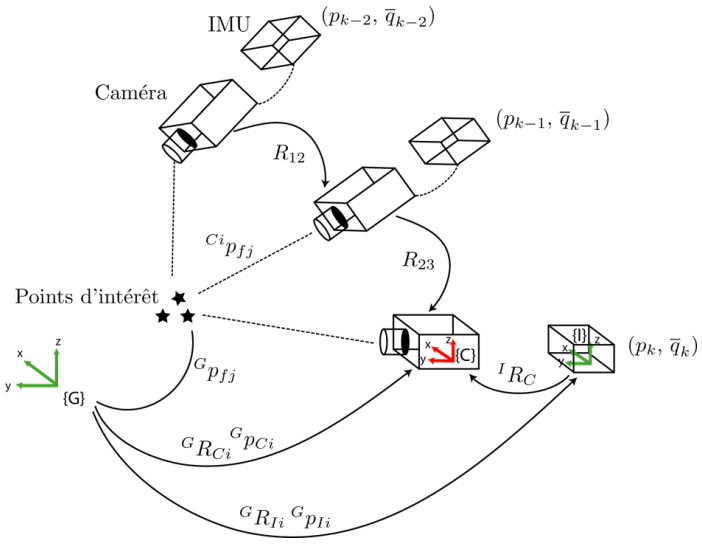


FIGURE 2.1 – Contraintes sur la fenêtre d'anciennes poses. Les points d'intérêt communs entre les anciennes poses sont utilisés comme contraintes pour la correction des poses.

Notations :

{G} {C} {I} représentent respectivement le repère global, le repère caméra et le repère IMU. Soit α et β deux repères, on note ${}^{\beta}p_{\alpha}$ le vecteur position du repère α exprimé dans

β et ${}^{\beta}R_{\alpha}$ la matrice de rotation, ${}^{\beta}\overline{q}_{\alpha}$ est le quaternion équivalant à la rotation.

Vecteur d'état :

Le filtre MSCKF fusionne les données caméra/IMU en maintenant un vecteur d'état composé de l'état de l'IMU X_I, représentant la pose actuelle du robot et d'une fenêtre de m poses précédentes de l'IMU (pour réduire la taille des matrices et les notations la valeur $m = 2$ est utilisée dans toutes les équations et notations). Le vecteur d'état aura ainsi la forme suivante :

$$X_k = [\boldsymbol{X_I} \quad {}^{G}p_{k-1}^{T} \quad {}^{G}\overline{q}_{k-1}^{T} \quad {}^{G}p_{k-2}^{T} \quad {}^{G}\overline{q}_{k-2}^{T}] \tag{2.1}$$

Avec l'état de l'IMU donné par :

$$\boldsymbol{X_I} = [{}^{G}p_{I}^{T} \quad {}^{G}\overline{q}_{I}^{T} \quad {}^{G}v_{I}^{T} \quad b_{a}^{T} \quad b_{g}^{T}] \tag{2.2}$$

Les biais de l'accéléromètre et du gyroscope b_a et b_g sont modélisés comme des processus aléatoires conduits par les bruits blancs Gaussiens n_{ba} et n_{bg} respectivement. ${}^{G}p_I$, ${}^{G}\overline{q}_I$ et ${}^{G}v_I$ indiquent respectivement la position de l'IMU, un quaternion représentant la rotation de l'IMU et la vitesse de l'IMU dans le repère global.

Mesures IMU : Ces mesures sont utilisées pour obtenir une estimation initiale de l'état (propagation d'état). L'accélération mesurée a_m et la vitesse angulaire ω_m sont modélisées par les équations suivantes :

$$\begin{aligned} a_m &= R^T({}^{G}\overline{q}_I)({}^{G}a_I - {}^{G}g) + b_a + n_a, \\ \omega_m &= \omega + b_g + n_g \end{aligned}$$

avec n_a, n_g des bruits blancs Gaussiens de l'accéléromètre et du gyroscope respectivement.

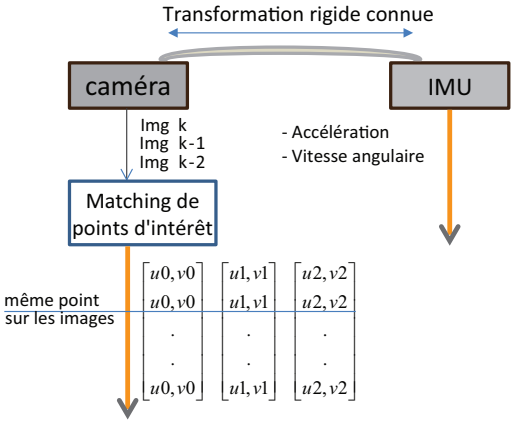


FIGURE 2.2 – Les entrées du filtre.

Mesures Caméra : Les points d'intérêt détectés sont suivis dans une fenêtre de m images précédentes et ensuite utilisés comme contraintes géométriques pour corriger l'estimation d'état obtenue par intégration de l'évolution des données IMU. Les entrées des algorithmes étudiés sont résumées dans la Fig. 2.2. Notons que nous supposons la caméra calibrée ainsi que la transformation rigide entre caméra et IMU déterminée par calibration extrinsèque.

Calibration Caméra IMU : Une des tâches les plus fastidieuse est la calibration IMU-caméra. Cette calibration consiste à trouver la transformation entre les repères propres des deux capteurs. Dans notre application, il est même judicieux d'estimer en plus les paramètres temporelles de la caméra par rapport à l'IMU. Pour le faire, on a opté pour une calibration basée sur un package ROS nommé Kalibr [Furgale et al., 2013] qui permet, à partir d'une séquence de données caméra/IMU face à une mire de calibration, de remonter aux paramètres de calibration (Fig.2.3).

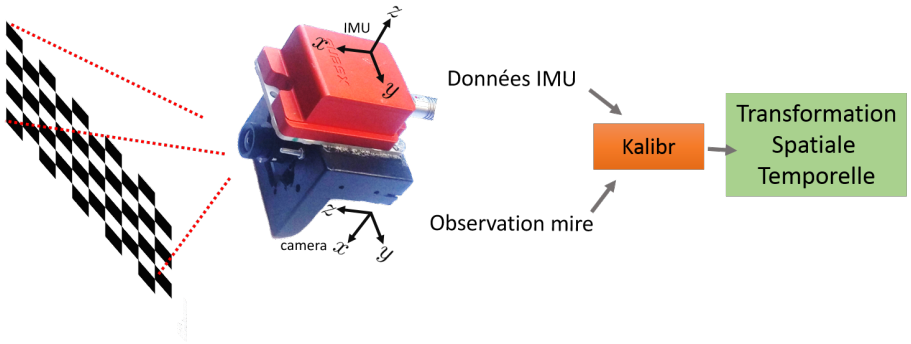
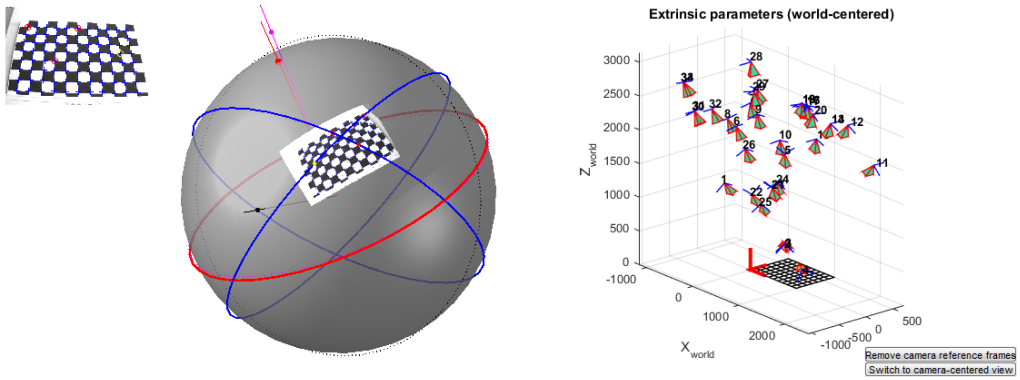


FIGURE 2.3 – Calibration caméra-IMU.



(a) (b)

FIGURE 2.4 – Résultat de calibration caméra-IMU.

La Fig.2.4.b montre les paramètres extrinsèques de la caméra utilisée dans l'estimation de la transformation entre caméra et IMU. La Fig.2.4.a montre le quaternion résultant de la calibration.

La précision d'estimation de la transformation entre la caméra et l'IMU est requise. Une erreur de 3° en rotation est souvent fatale pour cette structure de fusion.

Paramètres des bruits IMU

L'estimation des paramètres des bruits qui entachent les données du capteur IMU est d'une grande importance dans cet algorithme. Dans la Fig.2.5, on peut observer la présence des bruits lors d'un enregistrement de 10 secondes des données IMU en statique. On peut observer Fig.(2.6),(2.7) que ces bruits peuvent être considérés comme Gaussiens.

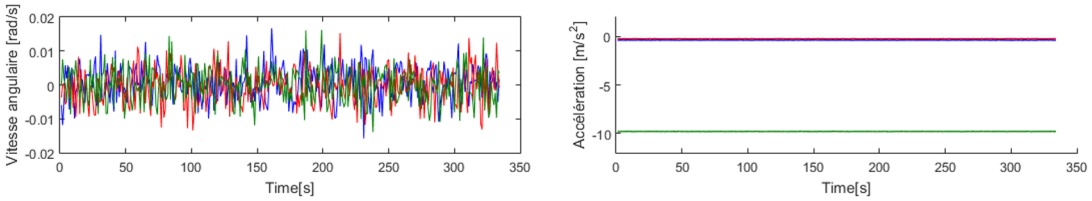


FIGURE 2.5 – Mesures statiques du gyroscope (droite) et accéléromètre (gauche) sur les trois axes.

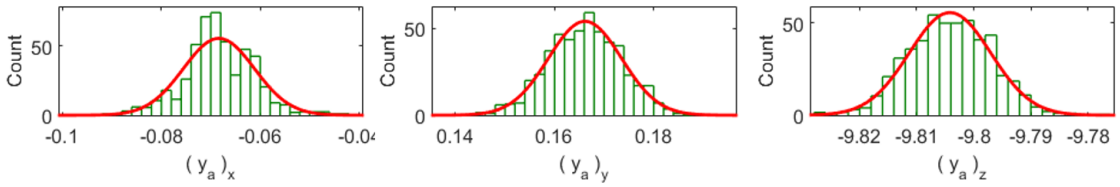


FIGURE 2.6 – Histogrammes (vert) des mesures de l'accéléromètre pendant 10 secondes, les données capteur sont prises en position statique. En (rouge) l'ajustement d'une Gaussienne.

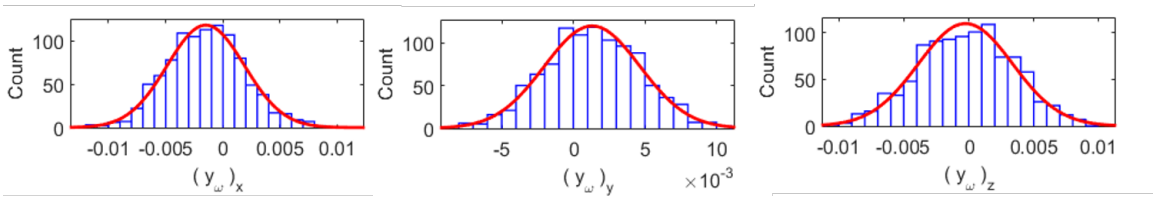


FIGURE 2.7 – Histogrammes (bleu) des mesures du gyroscope pendant 10 secondes, les données capteur sont prises en position statique. En (rouge) l'ajustement d'une Gaussienne.

Il est nécessaire d'estimer les caractéristiques des bruits de mesure (σ_{n_a}, σ_{n_g}) et des bruits blancs responsables de la dérive des biais gyromètre et accéléromètre (σ_{ba}, σ_{bg}).
Il est toujours possible de faire ce réglage manuellement ; une tâche fastidieuse et qui ne garantit pas d'obtenir les meilleures performances. Dans le cas de l'IMU, on dispose souvent de données de spécification fournies par le fabriquant. Il est alors plus facile de calculer les paramètres recherchés ($\sigma_{n_a}, \sigma_{n_g}, \sigma_{ba}, \sigma_{bg}$) à partir de ces spécifications.
Bruits de mesure : Pour estimer l'écart-type des bruits de mesure (σ_{n_a}, σ_{n_g}), on s'appuie sur les données fournies par le constructeur. Pour échantillonner le signal continu, on place

	Accéléromètre	Gyroscope
Axis number	3 axes	3 axes
Full scale	78 m/s2	625°/s
Noise density nd	111.7 micro g/$\sqrt{Hz}$	0.029°/s/$\sqrt{Hz}$
Band width Bf	375 Hz	415 Hz
A/D resolution	16 bits	16 bits

FIGURE 2.8 – Spécifications constructeur du capteur IMU Xsens MTi-100.

un filtre passe-bas d'une fréquence de coupure qui respecte le théorème de Shannon, donc une fréquence de coupure au plus égale à la moitié de la fréquence d'échantillonnage. Le système aura alors une bande passante B_f et une puissance totale du bruit dans la mesure égale à l'intégrale du carré de la densité de bruit n_d sur la plage de la bande passante. La valeur RMS (root mean square) de ce bruit est donnée par $\sigma_n = \sqrt{\int_{B_f} n_d^2}$.
En prenant n_d comme constante et en appliquant un coefficient multiplicateur de 1.6 (filtre premier ordre) pour prendre en compte le fait que le filtre n'est pas parfait, on obtient $\sigma_n = n_d\sqrt{1.6\ B_f}$.
Ainsi, en utilisant les données constructeur de la Fig.2.8, σ_{n_a} et σ_{n_g} peuvent être calculés. Il est aussi possible d'estimer ces valeurs en calculant l'écart type d'un signal enregistré lorsque l'IMU est en position statique.
Bruits Random walk : Pour avoir σ_{ba} et σ_{bg}, la technique d'analyse Allan Variance (AVAR) peut être utilisée. Cette technique, conçue pour estimer la stabilité de la fréquence dans les oscillateurs, est souvent utilisée pour caractériser les bruits dans les mesures capteurs en fonction de la moyenne de temps.

La variance de Allan est calculée avec différents pas d'échantillonnage τ. Différents bruits aléatoires provoquent des tangentes différentes et souvent dans différentes régions de τ ce qui permet de facilement les identifier. La Fig.2.9 montre les différents bruits aléatoires détectables sur différentes régions de τ. Elle montre aussi les courbes de Allan Variance obtenues pour des signaux gyroscope et accéléromètre acquis en position statique. Sur la Fig.2.9, à $t = 1s$ et avec une tangente de $-1/2$, on peut lire l'écart type du bruit blanc de l'accéléromètre et du gyroscope (σ_{ba}, σ_{bg}). Le bias stability (qui est déjà fourni par le constructeur) peut être lu quand la tangente est à 0. Les valeurs des écarts type (σ_{ba}, σ_{bg}) des bruits responsables des processus aléatoires des biais sont lues à $t = 3s$ avec une tangente de $1/2$.

Les valeurs obtenues sont généralement multipliées par dix pour compenser l'effet d'autres bruits et les erreurs de modélisation.

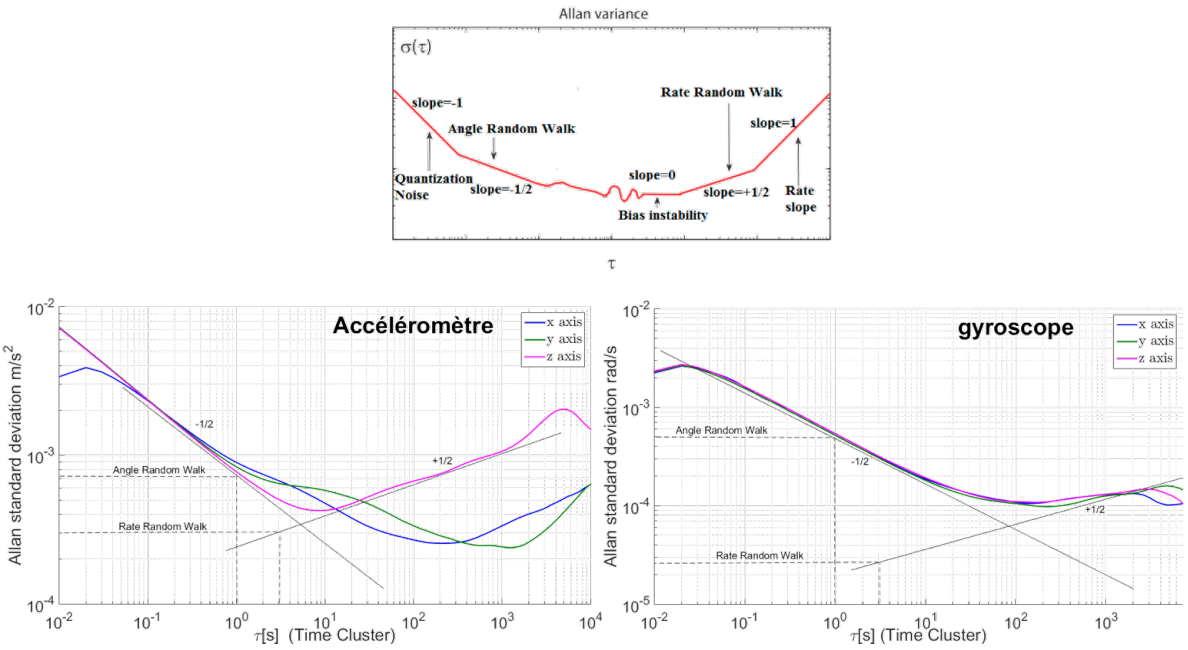


FIGURE 2.9 – Courbes d'Allan Variance.

2.4 Formulation des vecteurs d'état et d'erreur d'état

Suivant la définition du vecteur d'état (Eq.2.1), la dynamique réelle de l'état est exprimée par les équations suivantes :

$$\begin{cases} {}^{G}\dot{p}_I = & {}^{G}v_I \\ {}^{G}\dot{\overline{q}}_I = & \frac{1}{2}\,{}^{G}\overline{q}_I \otimes [0 \quad {}^{I}\omega^T] \\ {}^{G}\dot{v}_I = & {}^{G}a \\ {}^{G}\dot{b}_a = & n_{ba} \\ {}^{G}\dot{b}_g = & n_{bg} \end{cases} \tag{2.3}$$

$$\begin{cases} {}^{G}\dot{p}_{k-1} = & 0_{3\times1} \quad {}^{G}\dot{\overline{q}}_{k-1} = 0_{4\times1} \\ {}^{G}\dot{p}_{k-2} = & 0_{3\times1} \quad {}^{G}\dot{\overline{q}}_{k-2} = 0_{4\times1} \end{cases} \tag{2.4}$$

Les m dernières poses dans le vecteur d'état n'évoluent pas pendant l'étape de propagation du filtre. Ceci est dû au fait que les données IMU disponibles dans l'étape de propagation, ne décrivent que l'évolution de la pose actuelle et ne donnent aucune contrainte permettant la rectification des m poses précédentes. Leur dynamiques sont donc mises à zéro dans l'équation (Eq. 2.4).

Dans le MSCKF, le filtre fonctionne sur l'erreur de l'état au lieu de l'état réel. Cela nous permettra d'utiliser des signaux de faibles amplitudes qui sont linéairement intégrables et adaptés à la linéarisation. Cela réduira également la taille du vecteur d'état. Ainsi, le

vecteur d'état utilisé est $\tilde{X}_k$ au lieu de X_k, de sorte que :

$$X_k = \tilde{X}_k \oplus \hat{X}_k \tag{2.5}$$

où $\hat{X}_k(\hat{p}_I, \hat{\overline{q}}_I, \hat{v}_I, \hat{b}_a, \hat{b}_g, ...)$ est l'état nominal qui est estimé en parallèle du filtre, en utilisant la méthode de Runge-Kutta d'ordre 4 ($\oplus$ est la fonction d'addition d'état). Sa dynamique peut être dérivée des Eqs.(2.3) (2.4) sans les bruits et les perturbations et en utilisant $\hat{a} = a_m - \hat{b}_a, \hat{\omega} = \omega_m - \hat{b}_g$. La dynamique nominale résultante est décrite dans les équations suivantes :

$$\begin{gathered} {}^G\dot{\hat{p}}_I = {}^G\hat{v}_I, \;\; {}^G\dot{\hat{\overline{q}}}_I = \frac{1}{2}{}^G\hat{\overline{q}}_I \otimes \hat{\overline{\omega}}, \;\; {}^G\dot{\hat{v}}_I = R({}^G\hat{\overline{q}}_I)\hat{a} - {}^Gg \\ \dot{\hat{b}}_a = 0_{3\times1}, \;\; \dot{\hat{b}}_g = 0_{3\times1} \\ {}^G\dot{\hat{p}}_{I-1} = 0_{3\times1}, \;\; {}^G\dot{\hat{\overline{q}}}_{I-1} = 0_{4\times1}, \;\; {}^G\dot{\hat{p}}_{I-2} = 0_{3\times1}, \;\; {}^G\dot{\hat{\overline{q}}}_{I-2} = 0_{4\times1} \end{gathered} \tag{2.6}$$

Pour avoir la **dynamique de l'erreur d'état** $\tilde{X}_k(\tilde{p}_I, \delta\theta_I, \tilde{v}_I, \tilde{b}_a, ...)$, un modèle d'erreur additif est utilisé pour la position, la vitesse et les biais (exemple,${}^G\tilde{v}_I = {}^Gv_I - {}^G\hat{v}_I$). Pour l'erreur du quaternion, la paramétrisation locale d'erreur suivante est utilisée :
${}^G\overline{q}_I = {}^G\hat{\overline{q}}_I \otimes [1 \;\; \frac{1}{2}{}^G\delta\theta_I^T]$ avec $\delta\theta$ l'erreur d'orientation exprimée dans le repère global ($\otimes$ est l'opération de multiplication des quaternions). Soulignons qu'une mauvaise paramétrisation de l'erreur du quaternion peut affecter les propriétés d'observabilité du système [Li and Mourikis, 2011].
Le vecteur d'état utilisé dans le filtre est alors :

$$\tilde{X}_k = [{}^G\tilde{p}_I^T \; {}^G\delta\theta_I^T \; {}^G\tilde{v}_I^T \;\; \tilde{b}_a^T \;\; \tilde{b}_g^T \;\; {}^G\tilde{p}_{k-1}^T \; {}^G\delta\theta_{k-1}^T \; {}^G\tilde{p}_{k-2}^T \; {}^G\delta\theta_{k-2}^T]^T \tag{2.7}$$

2.5 Propagation

Dans ce travail, plusieurs variantes du filtre MSCKF seront implémentées (MSC**E**KF, MSC**U**KF). L'étape de propagation est identique dans toutes les variantes. Il n'y a pas de différence entre l'implémentation des filtres dans cette étape, mais plutôt dans l'étape suivante des filtres (la mise à jour).
Le système continu linéarisé de la dynamique d'erreur d'état est le suivant :

$$\begin{aligned} \dot{\tilde{X}} &= F_c\tilde{X} + G_c n \\ &= \begin{bmatrix} F_m & 0_{15\times12} \\ 0_{12\times15} & 0_{12\times12} \end{bmatrix} \tilde{X} + \begin{bmatrix} G_m \\ 0_{12\times12} \end{bmatrix} n \end{aligned} \tag{2.8}$$

Avec $n = [n_a^T \ n_{ba}^T \ n_g^T \ n_{bg}^T]$ est le bruit d'état. Notons 0_3 et I_3 des matrices zéro et identité de dimension (3x3) respectivement, on a alors :

$$F_m = \begin{bmatrix} 0_3 & 0_3 & I_3 & 0_3 & 0_3 \\ 0_3 & -\lfloor\hat{\omega}\rfloor_\times & 0_3 & 0_3 & -I_3 \\ 0_3 & -R(^G\hat{\overline{q}}_I)\lfloor\hat{a}\rfloor_\times & 0_3 & -R(^G\hat{\overline{q}}_I) & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & 0_3 \end{bmatrix} \tag{2.9}$$

$$G_m = \begin{bmatrix} 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & -I_3 & 0_3 \\ -R(^G\hat{\overline{q}}_I) & 0_3 & 0_3 & 0_3 \\ 0_3 & I_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & I_3 \end{bmatrix}$$

En discrétisant le système Eq.(2.8) et en utilisant les séries de Taylor, le système résultant est $\dot{\tilde{X}}_k = F_d\tilde{X}_k + G_d n$. F_d est une matrice "creuse" qui peut être exprimée sous la forme analytique suivante [Weiss and Siegwart, 2011] :

$$F_d = \begin{bmatrix} I_3 & \Phi_{12} & \Phi_{13} & \Phi_{14} & \Phi_{15} & 0_{3\times12} \\ 0_3 & \Phi_{22} & 0_3 & 0_3 & \Phi_{25} & 0_{3\times12} \\ 0_3 & \Phi_{32} & I_3 & \Phi_{34} & \Phi_{35} & 0_{3\times12} \\ 0_3 & 0_3 & 0_3 & I_3 & 0_3 & 0_{3\times12} \\ 0_3 & 0_3 & 0_3 & 0_3 & I_3 & 0_{3\times12} \\ 0_{12\times3} & 0_{12\times3} & 0_{12\times3} & 0_{12\times3} & 0_{12\times3} & I_{12} \end{bmatrix} \tag{2.10}$$

La matrice de covariance discrétisée associée au bruit d'état Q_d est calculée par :

$$Q_d = \int_{\Delta t} F_d(\tau) G_c Q_c G_c^T F_d(\tau)^T d\tau \tag{2.11}$$

où Q_c est la matrice de covariance du bruit d'état :

$$Q_c = diag(\sigma_a^2 I_3,\ \sigma_g^2 I_3,\ \sigma_{ba}^2 I_3,\ \sigma_{bg}^2 I_3) \tag{2.12}$$

La propagation de l'erreur d'état est alors donnée par :

$$\tilde{X}_{k|k-1} = F_d \tilde{X}_{k-1|k-1} \tag{2.13}$$

Et la propagation de la covariance associée à l'état est :

$$P_{k|k-1} = F_d P_{k-1|k-1} F_d^T + Q_d \tag{2.14}$$

En parallèle de la propagation de l'erreur d'état, l'état nominal $\hat{X}$ est obtenu en intégrant Eq.(2.6) par la méthode de Runge-Kutta d'ordre 4.

2.6 Modèles d'observation

Les équations de mesure fournissent des contraintes sur le vecteur d'état et ses estimations précédentes. Elle permettent de corriger les dérives des transformations entre les poses présentes dans le vecteur d'état. Deux types de modèles de mesure seront explorés. Un modèle rapide et pratique pour les mises à jour dans le filtre, et un autre plus lent, mais qui a montré certains avantages dans la partie implémentation, comme nous le verrons dans la section des résultats. Les deux équations de mesure utilisent exactement les mêmes données disponibles dans les dernières $m+1$ images de la fenêtre MSCKF. La différence entre les deux modèles de mesure réside dans la formulation mathématique des contraintes d'observation. On suppose ici que la détection et la mise en correspondance des points d'intérêt sont disponibles à cette étape.

2.6.1 Modèle d'observation basé-triangulation

Cette équation de mesure (que nous noterons me1) représente la position normalisée de chaque point d'intérêt dans chacune des $m+1$ poses de la caméra. Les observations sont décrites par le modèle suivant :

$$z_j^i = \frac{1}{{}^{Ci}Z_j} \begin{bmatrix} {}^{Ci}X_j \\ {}^{Ci}Y_j \end{bmatrix} + n_j^i \tag{2.15}$$

où n_j^i est un vecteur (2×1) de bruit d'image et $R_j^i = \sigma_{img}^2 I_2$ la matrice de covariance correspondante, avec i et j sont les indices représentant respectivement l'image utilisée et le point d'intérêt utilisé dans l'équation de mesure.

Cette équation de mesure est la plus communément utilisée dans la littérature. La mesure correspond dans ce cas à la projection d'un point 3d sur le plan caméra. Pour établir l'équation de mesure, des choix se présentent. Comme le vecteur d'état est constitué de plusieurs positions de l'IMU et non de la caméra, on peut être tenté d'écrire l'équation de mesure qui respecte le schéma (2) de la Fig.2.10. Ce qui revient à projeter le point d'intérêt d'abord dans le repère de l'IMU et ensuite faire un changement de repère vers le repère caméra. Cela permet d'avoir une équation de mesure beaucoup plus simple que si l'on projette le point directement dans le repère caméra. Le problème est que l'étape de normalisation que doit subir un point en le projetant sur le repère caméra ne donne pas la même information si le point est d'abord projeté dans le repère IMU, ce qui nous contraint au schéma (1) de la Fig.2.10.

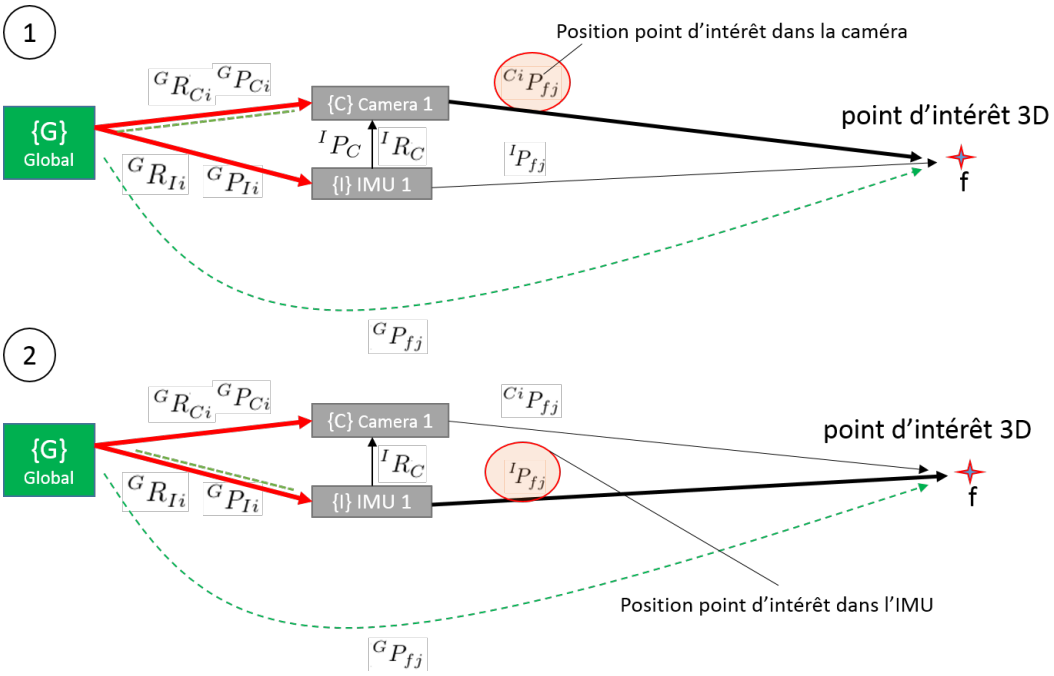


FIGURE 2.10 – Modèle de mesure basé triangulation. En rouge, les grandeurs à estimer, les grandeurs entourées en rouge représentent le choix de la première projection des points d'intérêt. En vert pointillé, les grandeurs utilisées pour écrire l'équation de mesure.

La position d'un point d'intérêt f_j dans chacune des poses de la caméra est alors décrite comme (voir Fig.2.1) :

$$ {}^{Ci}p_{fj} = \begin{bmatrix} {}^{Ci}X_j \\ {}^{Ci}Y_j \\ {}^{Ci}Z_j \end{bmatrix} = {}^{G}R_{Ci}^{T}({}^{G}p_{fj} - {}^{G}p_{Ci}) \tag{2.16} $$

$$ {}^{Ci}p_{fj} = {}^{I}R_{C}^{T}{}^{G}R_{Ii}^{T}({}^{G}p_{fj} - {}^{G}p_{Ii} - {}^{G}R_{Ii}{}^{I}p_{C}) \tag{2.17} $$

$({}^{I}p_C, {}^{I}R_C)$ sont données par la calibration extrinsèque caméra-IMU.
Cette équation de mesure prend en entrée la position triangulée des points d'intérêt ${}^{G}p_{fj}$. Cette triangulation des points d'intérêt est réalisée avec un algorithme d'optimisation Gauss Newton, voir Annexe B.

2.6.2 Modèle d'observation basé-transfert trifocal

Cette forme d'équation de mesure (que nous noterons me2) évite la reconstruction de la position des points d'intérêt ${}^{G}p_{fj}$. Elle est par conséquence plus rapide. Elle définit deux contraintes épipolaires et un transfert trifocal des points d'intérêt de l'image $k-2$ vers l'image courante k (Fig. 2.11). Le même modèle de mesure que [Hu and Chen, 2014] est utilisé. Ce modèle de mesure est appliqué (dans le cas réel où m=6) à chaque trois

images successives dans le vecteur d'état.

$$z_j = h\left(g(\tilde{X}, \hat{X}), \{m_1, m_2, m_3\}_j\right) \tag{2.18}$$

$$= \begin{bmatrix} \tilde{m}_2^T \left({}^G R_{12}\right)^T \lfloor t_{12}\times \rfloor \tilde{m}_1 \\ \tilde{m}_3^T \left({}^G R_{23}\right)^T \lfloor t_{23}\times \rfloor \tilde{m}_2 \\ K\left(\sum_{i=1}^{3} \tilde{m}_{1i} T_i^T\right) l_2 \end{bmatrix} \tag{2.19}$$

m_1, m_2, m_3 sont les coordonnées des points d'intérêt dans les trois dernières images et $\tilde{m}_1, \tilde{m}_2, \tilde{m}_3$ leurs positions normalisées, avec $\tilde{m}_k = K^{-1} m_k$ et K la matrice de calibration intrinsèque de la caméra.

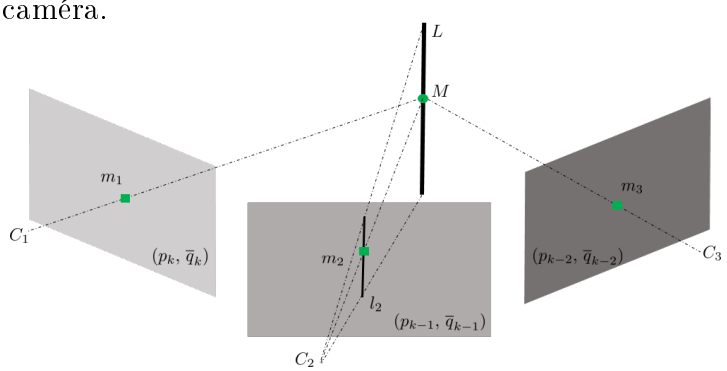


FIGURE 2.11 – Correspondance point-line-point [Hartley and Zisserman, 2003]. Un point 3d M est imagé comme un triplet (m_1, m_2, m_3) dans trois poses de la caméra indiquées par leurs centres C_1, C_2, C_3. Inversement, les lignes correspondantes projetées à partir des trois images s'intersectent toutes en un seul point 3d.

La première et la seconde ligne de l'équation de mesure Eq.2.19 estiment la satisfaction de la contrainte épipolaire pour chaque correspondance des points d'intérêt, où (R_{12}, t_{12}) et (R_{23}, t_{23}) sont les rotations et translations relatives entre les trois dernières poses de la caméra. La dernière ligne de Eq.(2.19) donne une estimation des coordonnées des points sur l'image courante en fonction de leurs coordonnées dans la pose $(p_{k-2}, \overline{q}_{k-2})$ et de la meilleure estimation du vecteur d'état. Le transfert trifocal est obtenu comme suit [Hartley and Zisserman, 2003] :

- Si l'on considère les matrices de projection de la caméra à travers une fenêtre de trois poses comme :

$$\begin{cases} \pi_k &= [I|0] \\ \pi_{k-1} &= [B|b_4] = [R_{12}^T \mid -R_{12}^T t_{12}] \\ \pi_{k-2} &= [C|c_4] = [R_{13}^T \mid -R_{13}^T t_{13}] \end{cases} \tag{2.20}$$

- avec b_i et c_i les ièmes colonnes de π_{k-1} et de π_{k-2} respectivement, et R_{12}, R_{13} sont :

$$R_{12} = {}^G R_{C1}^T {}^G R_{C2} \; ; \quad t_{12} = {}^G R_{C1}^T ({}^G p_{C2} - {}^G p_{C1})$$
$$R_{13} = {}^G R_{C1}^T {}^G R_{C3} \; ; \quad t_{13} = {}^G R_{C1}^T ({}^G p_{C3} - {}^G p_{C1})$$

- Alors, la position d'un point dans le plan image courante est :

$$\tilde{m}_3 = (\sum_{i=1}^{3} \tilde{m}_{1i} T_i^T) l_2 \tag{2.21}$$

T_i est le tenseur trifocal dérivé des matrices de projection de caméra :

$$T_i = b_i c_4^T - b_4 c_i^T \tag{2.22}$$

La ligne l_2 est la ligne perpendiculaire à l'épipole Fig.2.11 :

$$l_2 = (l_{e2}, -l_{e1}, -\tilde{m}_{2u} l_{e2} + \tilde{m}_{2v} l_{e1}) \tag{2.23}$$

Et (l_{e1}, l_{e2}, l_{e3}) est l'épipolaire obtenue à partir du point m_1 et de la matrice de projection P_{k-1} avec $l_e = P_{k-1}\tilde{m}_1$.

2.7 Élimination des fausses correspondances

Les mauvaises correspondances entre les points d'intérêt sont éliminées en utilisant un EKF-RANSAC proche de la technique 1-point décrite dans [Civera et al., 2010] [Troiani et al., 2014]. Dans cette technique (Fig.2.12), nous réalisons une première estimation de la mesure en utilisant seulement un point d'intérêt à chaque fois, puis en utilisant un seuil défini (Eq. 2.24), nous testons si les autres mesures ont le même mouvement que le point utilisé. Cela permet de trouver le plus grand nuage de points d'intérêt qui respectent le même mouvement, les autres sont considérés comme aberrants.

Le critère de test utilisé ici est :

$$\left\| m_3 - \left(K \sum_i \tilde{m}_{1i} T_i^T \right) l_2 \right\| < threshold \tag{2.24}$$

avec un seuil d'arrêt (*threshold*) donné en pixels (typiquement le seuil de 4 pixels est une bonne valeur pour ce type d'algorithme).
Dans notre algorithme, on fait une estimation de mesure basée sur l'équation de mesure (me2). Pour le réaliser, on utilise la pose courante de la caméra, la pose la plus ancienne et une pose au milieu de l'évolution (dans le cas où la fenêtre MSCKF m est supérieure à 3).

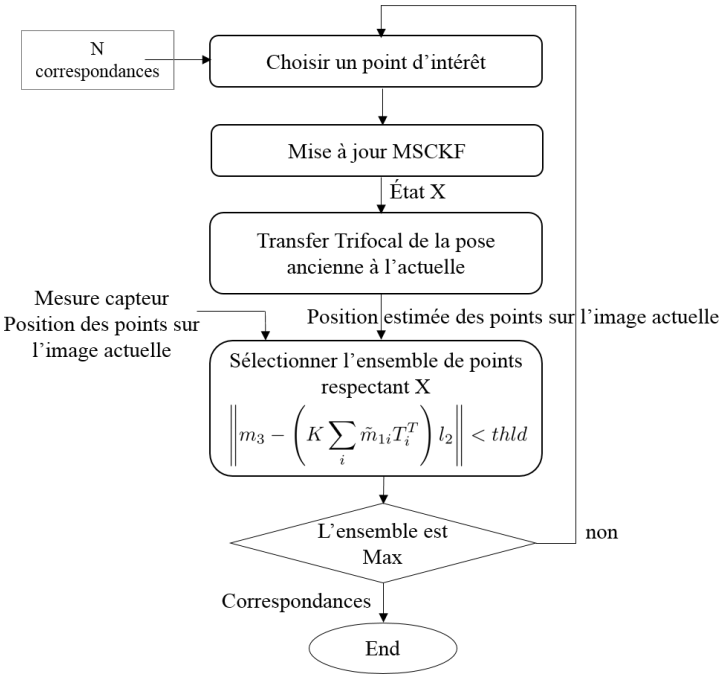


FIGURE 2.12 – Suppression des fausses correspondances.

2.8 Mise à jour du filtre MSCKF

Dans l'étape de mise à jour, nous utilisons l'état propagé Eq.(2.13)(2.14) et les points d'intérêt sur la fenêtre de $m+1$ images (Fig.2.2) dans les modèles d'observation. Nous avons considéré et comparé deux variantes : une mise à jour basée UKF et une autre basée EKF.

2.8.1 Mise à jour basée Sigma-points (UKF)

Le filtrage UKF utilise une transformation "unscented" pour déterminer une représentation appropriée des variables d'état. Ce filtrage est utilisé en particulier lorsque le modèle du système est hautement non linéaire. Une approche par points Sigma est utilisée. Elle consiste à créer un nuage de points appelé points Sigma à partir de l'état actuel et de la matrice de covariance :

$$\tilde{X}^l_{k|k-1} = \tilde{X}_{k|k-1} \pm \left(\sqrt{(L+\lambda) P_{k|k-1}} \right)_l \tag{2.25}$$

Un nuage de $2L+1$ points Sigma est créé dans l'équation précédente (L est la dimension du vecteur d'état et λ un paramètre d'échelle). Dans l'étape suivante, on injecte chacun

des points Sigma dans l'équation de mesure pour avoir une estimation de mesure.

$$\begin{aligned} Z_j^l &= h\left(g\left(\hat{X}_{k|k-1}, \tilde{X}_{k|k-1}^l\right), (m_1, m_2, m_3)_j\right) \\ \hat{z}_j &= \sum_{l=0}^{2L} W_s^l Z_j^l \end{aligned} \tag{2.26}$$

Le gain K_k du filtre est calculé par :

$$\begin{aligned} P_{z_j z_j} &= \sum_{l=0}^{2L} \left(Z_j^l - \hat{z}_j\right)\left(Z_j^l - \hat{z}_j\right)^T + R \\ P_{x z_j} &= \sum_{l=0}^{2L} W_c^l \left(\tilde{X}_{k|k-1}^l - \tilde{X}_{k|k-1}\right)\left(Z_j^l - \hat{z}_j\right)^T \\ K_k &= P_{x z_j} P_{z_j z_j}^{-1} \end{aligned}$$

Avec W_c^l et W_s^l des matrices de poids utilisées dans le filtre UKF. La mise à jour de l'état et la covariance sont données par :

$$\begin{aligned} \tilde{X}_{k|k} &= \tilde{X}_{k|k-1} + K_k\left(z_j - \hat{z}_j\right) \\ P_{k|k} &= P_{k|k-1} - K_k P_{z_j z_j} K_k^T \end{aligned}$$

En additionnant la correction à l'état nominal, on obtient :

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} \oplus \tilde{X}_{k|k}$$

Notons que la fonction h dans Eq.2.26 est générique et peut correspondre à (me1 Eq.2.16) ou (me2 Eq.2.18) au choix. Puisqu'il n'y a pas de linéarisation, il est possible de changer l'équation de mesure facilement. Cela donne à l'UKF une flexibilité qui peut être utile lors du développement et des tests des algorithmes.

2.8.2 Mise à jour basée EKF

Basé sur le concept de linéarisation, l'EKF utilise des matrices jacobiennes pour dériver la transition d'état. Notons que la dérivation des équations de mesure peut être intraitable dans certains cas, comme pour le calcul des jacobiennes de l'équation de mesure du tenseur trifocal (me2). Ceci limite l'utilisation de l'EKF dans cette étude à l'équation de mesure (me1). En utilisant ce modèle d'observation pour réaliser des mises à jour du filtre EKF,

nous avons :

$$\hat{z}_j^i = \frac{1}{{}^{Ci}\hat{Z}_j}\begin{bmatrix} {}^{Ci}\hat{X}_j \\ {}^{Ci}\hat{Y}_j \end{bmatrix}, \begin{bmatrix} {}^{Ci}\hat{X}_j \\ {}^{Ci}\hat{Y}_j \\ {}^{Ci}\hat{Z}_j \end{bmatrix} = {}^{I}R_C^T {}^{G}\hat{R}_{Ii}^T({}^{G}\hat{p}_{fj} - {}^{G}\hat{p}_{Ii}) - {}^{I}R_C^T {}^{I}p_C \tag{2.27}$$

Dans l'Eq.2.27, la position des points d'intérêt dans le repère global ${}^{G}p_{fj}$, est supposée connue (la position des points est estimée à l'extérieur du filtre avec une minimisation moindre carré, Annexe.B).
En utilisant l'équation (2.15) et (2.27), le résidu s'écrit de la forme :

$$r_j^i = z_j^i - \hat{z}_j^i \tag{2.28}$$

En linéarisant par rapport à l'état et la position des points d'intérêt, le résidu peut être écrit sous la forme :

$$r_j^i \simeq H_{Xi}^{(j)}\tilde{X} + H_f^{(j)G}\tilde{p}_{f_j} + n_j^i \tag{2.29}$$

Où $H_{Xi}^{(j)}$ et $H_f^{(j)}$ sont respectivement les jacobiennes par rapport au vecteur d'état et à la position reconstruite des points d'intérêt :

$$H_{Xi}^{(j)} = \begin{bmatrix} .... & 0_{2\times 9} & -J_i^{(j)I}R_C^T{}^{G}\hat{R}_{I1}^T & J_i^{(j)I}R_C^T\lfloor {}^{G}\hat{R}_{I1}^T({}^{G}\hat{p}_{fj} - {}^{G}\hat{p}_{I1})\ \times \rfloor & ... \end{bmatrix} \tag{2.30}$$

$$H_{fi}^{(j)} = J_i^{(j)I}R_C^T{}^{G}\hat{R}_{I_i}^T \tag{2.31}$$

Avec :

$$J_i^{(j)} = \frac{1}{{}^{Ci}\hat{Z}_j}\begin{bmatrix} 1 & 0 & -\frac{{}^{Ci}\hat{X}_j}{{}^{Ci}\hat{Z}_j} \\ 0 & 1 & -\frac{{}^{Ci}\hat{Y}_j}{{}^{Ci}\hat{Z}_j} \end{bmatrix} \tag{2.32}$$

Le résidu Eq.2.29 n'est pas dans la forme habituelle utilisée dans un filtre EKF. Cela est dû à l'utilisation du vecteur d'état dans la reconstruction de la position des points d'intérêt, ce qui signifie que l'erreur ${}^{G}\tilde{p}_{f_j}$ est corrélée avec l'erreur d'état. Suivant [Mourikis and Roumeliotis, 2007], le résidu Eq.2.29 est projeté sur l'espace nul A de la matrice $H_{fi}^{(j)}$ pour obtenir le nouveau résiduel :

$$r_{0j}^i = A^T(z_j^i - \hat{z}_j^i) \simeq A^T H_{Xi}^{(j)}\tilde{X} + 0 + A^T n_j^i \tag{2.33}$$

$$= H_0^{(j)}\tilde{X} + n_{0j}^i \tag{2.34}$$

La matrice de covariance de n_{0j} est $R_0 = A^T R A$. La mise à jour est alors obtenue par :

$$\begin{aligned} K_k &= P_{k|k-1} H_0^T (H_0 P_{k|k-1} H_0^T + R_0)^{-1} \\ \tilde{X}_{k|k} &= \tilde{X}_{k|k-1} + K_k r_0 \\ P_{k|k} &= (I - K_k H_0) P_{k|k-1} (I - K_k H_0)^T + K_k R_0 K_k^T \end{aligned}$$

En ajoutant l'état nominal à l'erreur d'état, l'estimation de l'état global est :

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} \oplus \tilde{X}_{k|k}$$

2.9 Comparaison MSC-UKF/MSC-EKF

Dans l'état de l'art, l'algorithme MSCKF original [Mourikis and Roumeliotis, 2007] utilise une mise à jour EKF avec l'équation de mesure (me1) Eq.2.16. Plus récemment, [Sejong Heo, 2017], [Kong et al., 2015], [Hu and Chen, 2014] ont défini une mise à jour UKF qui utilise l'équation de mesure (me2) Eq.2.18.

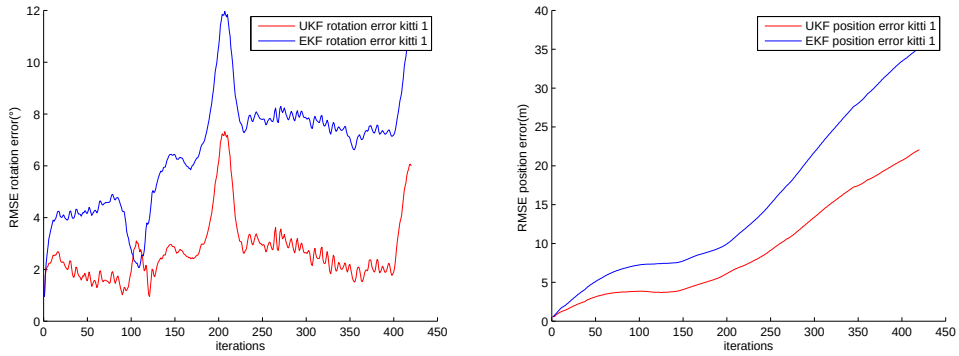
Dans notre travail, nous avons comparé les filtres MSCKF dans leur formulation EKF ou UKF, et en variant les modèles d'observation. Le but est de définir une architecture de filtre qui répond aux spécifications de précision dans une dynamique rapide, et qui peut être étendue au cas multi-robots.

Dans ce cas, le filtre MSCKF peut avoir quatre formes : EKF-me1, EKF-me2, UKF-me1 et UKF-me2.

Les algorithmes résultants des combinaisons UKF-me1 et EKF-me1 souffrent d'un problème de corrélation de l'état avec l'erreur de position des points d'intérêt. Dans le EKF, cela se traduit par un résidu qui n'est pas sous la forme habituelle utilisée en EKF. Cependant, l'étape de linéarisation dans un EKF permet de faire une projection sur l'espace nul et de trouver un nouveau résidu non corrélé, ce qui résout le problème pour la combinaison EKF-me1. Ceci n'est pas le cas dans la combinaison UKF-me1 où le UKF n'effectue aucune linéarisation qui permettrait une projection sur l'espace nul. Pour surmonter ce problème, nous utilisons un seul point Sigma de l'UKF (choisi pour être la moyenne des points Sigma par simplicité) pour imiter l'opération EKF et créer un nouveau résidu. Cela réduira l'effet de corrélation sur le filtre UKF-me1, mais on s'attend néanmoins à une petite détérioration de l'estimation car la corrélation n'est pas entièrement traitée.

La Fig.2.13 montre la dérive élevée de l'estimation de pose constatée en omettant la projection sur l'espace nul. L'effet de la corrélation de l'état à l'erreur de position des points d'intérêt est significatif sur les deux filtres (UKF-me1 et EKF-me1).

L'autre combinaison possible à considérer est la EKF-me2, mais qui a une Jacobienne



(a) (b)

FIGURE 2.13 – La sensibilité des filtres EKF et UKF aux erreurs de position des points d'intérêt en utilisant l'équation de mesure (me1) dans les deux filtres et en omettant la projection sur l'espace nul : (a) représente l'erreur en rotation sur KITTI1, et (b) l'erreur en position. [Clement et al., 2015] a également démontré le même effet sur l'EKF.

intraitable en raison de l'équation de mesure compliquée du transfert trifocal et n'est donc pas implémentée.

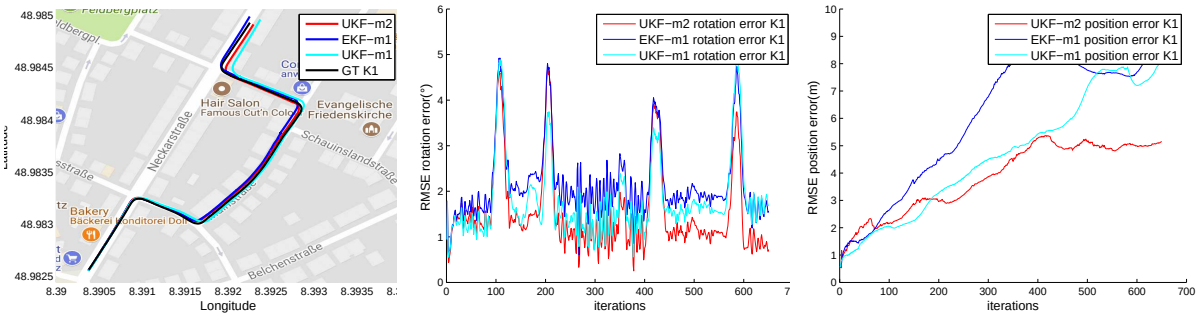
Les combinaisons proposées (EKF-me1, UKF-me1, UKF-me2) ont été implémentées en C++ sous ROS Kinetic (Robot Operating System). Des évaluations ont été menées sur deux scénarios de dynamiques variées :

- Le premier scénario exploite le jeu de données de la dataset KITTI [Geiger et al., 2013], qui contient des données provenant de deux caméras niveaux de gris, deux caméras couleur (Point Grey Flea2, 10 Hz, 1392x512 pixel, 90°x35° d'ouverture), d'un scanner laser et d'un GPS/IMU INS (OXTS RT 3003, 100 Hz). Une synchronisation de la caméra et des données IMU est réalisée dans KITTI de sorte que la différence maximale entre les horodatages est de 4 ms. Dans l'évaluation, nous avons seulement besoin des données d'une caméra niveaux de gris, de l'accélération et de la vitesse angulaire de l'IMU. Les données de calibration telles que le bruit IMU, la transformation caméra/IMU et la vitesse initiale sont fournies pour l'ensemble des données. Dans cette comparaison, nous utilisons cinq trajectoires KITTI, 2011-10-03-drive-0027, 2011-09-30-drive-0028, 2011-09-30-drive-0018, 2011-09-30-drive-0034 et 2011-09-30-drive-0020 (dénommées respectivement dans la suite KITTI 1,3,5,6,7). Ces trajectoires sont les plus utilisées dans la littérature [Hu and Chen, 2014], ce qui permet de confirmer la consistance de nos implémentations et résultats.
- Dans le deuxième scénario, on réalise des expériences en utilisant un drone quadrirotor équipé d'une IMU (100Hz) et d'une caméra Bluefox (752x480, 25Hz) dans un environnement intérieur (Fig.2.17). L'implémentation des filtres nécessite quelques ajustements dans cette étape pour réduire le temps de calcul (pour faire face à la capacité de calcul limitée sur un UAV). Dans ce scénario, le système de localisation Marvelmind basé sur des balises ultrasons est utilisé pour obtenir une réalité terrain. Ce système permet une localisation en position avec une précision de 2cm (Annexe.A).

2.9.1 Comparaison de la précision des filtres

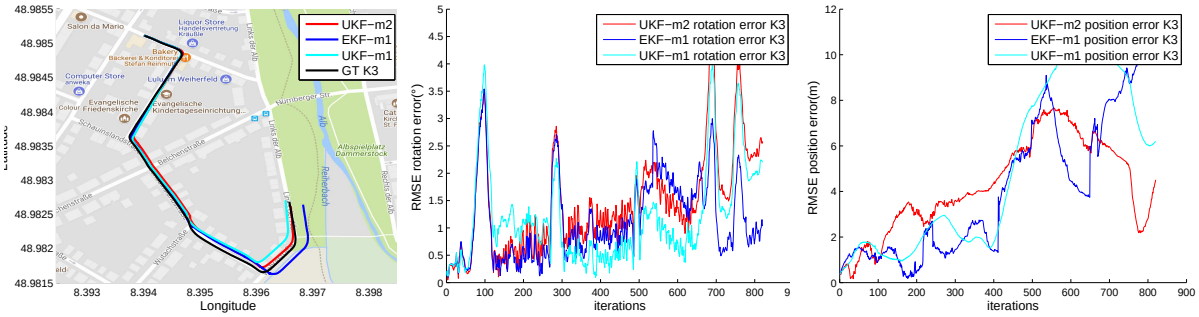
Ici, nous comparons la précision des filtres dans différentes conditions en utilisant les jeux de données KITTI. Les résultats de l'estimation de la pose sont rapportés dans Figs (2.14), (2.15), (2.16). Le tableau (2.1) donne les erreurs quadratiques moyennes (RMSE) globales et les erreurs d'estimation au point final.

On peut voir que dans toutes les trajectoires KITTI, la valeur du RMSE en position



(a) (b) (c)

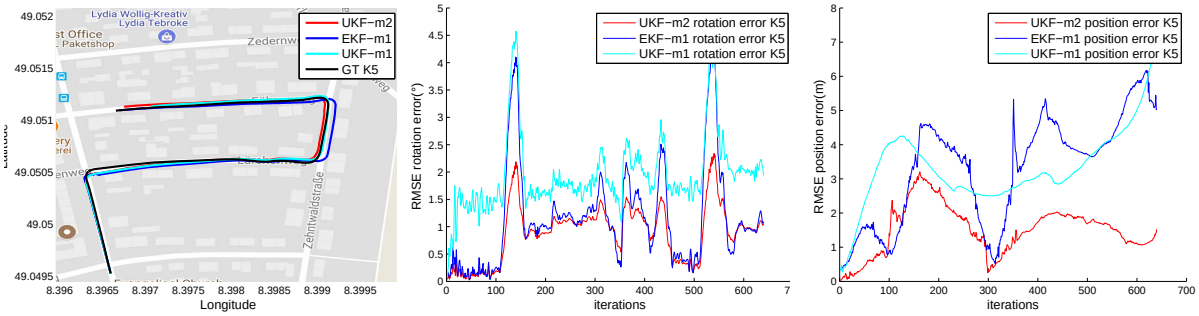
FIGURE 2.14 – Comparaison des performances des filtres sur la trajectoire KITTI 1. (a) Précision sur la KITTI 1, GT représente la vérité terrain. (b)(c) Erreur RMSE en rotation et position.



(a) (b) (c)

FIGURE 2.15 – Comparaison des performances des filtres sur la trajectoire KITTI 3. (a) Précision sur la KITTI 3, GT représente la vérité terrain. (b)(c) Erreur RMSE en rotation et position.

de la combinaison UKF-me2 est meilleure que celle du EKF-me1 et UKF-me1. Ceci est également vrai pour les erreurs en position et rotation du point final. Cela signifie que l'UKF combiné avec le modèle d'observation (me2) a une meilleure précision que les autres combinaisons. Maintenant, si nous comparons UKF-me1 à EKF-me1 où nous avons exactement la même équation de mesure, on trouve que malgré le fait que le UKF souffre encore du problème de corrélation entre l'état et les erreurs de position des points d'intérêt (car nous projetons sur l'espace nul en utilisant un seul point Sigma), les filtres donnent des résultats proches. En joignant les deux observations, nous pouvons



(a) (b) (c)

FIGURE 2.16 – Comparaison des performances des filtres sur la trajectoire KITTI 5. (a) Précision sur la KITTI 5, GT représente la vérité terrain. (b)(c) Erreur RMSE en rotation et position.

affirmer que les propriétés statistiques supérieures de l'UKF par rapport à l'EKF ont été observées dans ce test de mouvement relativement uniforme. L'équation de mesure (me2) semble également être un meilleur choix dans ce cas. Notons que l'amplitude des erreurs d'estimation dans nos résultats est équivalente à celle obtenue dans l'implémentation UKF dans [Hu and Chen, 2014].

	EKF-me1 RMSE totale (m)-(deg)	UKF-me2 RMSE totale (m)-(deg)	UKF-me1 RMSE totale (m)-(deg)	EKF-me1 erreur point final (m)	UKF-me2 erreur point final (m)	UKF-me1 erreur point final (m)
KITTI 1	6.51/2.44	3.98/2.06	4.95/1.98	8.24	4.98	7.38
KITTI 3	5.29/1.13	4.69/1.24	6.23/1.47	11.55	4.81	6.52
KITTI 5	3.63/1.46	1.67/0.73	3.48/2.07	5.03	1.52	7.24
KITTI 6	7.69/1.34	5.91/0.93	6.94/1.42	12.99	8.82	12.08
KITTI 7	8.23/2.31	6.72/1.70	8.79/2.24	17.88	15.25	18.62

Tableau 2.1 – RMSE globale et erreurs de point final sur KITTI.

2.9.2 Comparaison de la robustesse à des dynamiques rapides et non uniformes

Des tests expérimentaux UAV sont réalisés pour permettre la comparaison de la robustesse des filtres à des mouvements à haute dynamique et agilité. En effet le mouvement d'un drone combine souvent des déplacements rapides avec des phases de vol stationnaire. Les filtres fonctionnent à 20Hz dans une implémentation C++ ROS Kinetic. Certaines adaptations supplémentaires ont été nécessaires pour passer des tests KITTI à une implémentation en temps réel "mou" sur le drone. Tout d'abord, la transformation rigide entre la caméra et l'IMU est estimée en utilisant la procédure décrite dans (Sec.2.3) [Furgale et al., 2013]. Le détecteur et le matcheur des points d'intérêt sont changés du SIFT au détecteur BRIEF avec une mise en correspondance utilisant une

corrélation croisée normalisée (NCC). Enfin, les variances des bruits du gyroscope et de l'accéléromètre ont été estimées en utilisant les courbes de "Allan Standard Deviation Plot" (Sec.2.3). Les trajectoires ci-dessous, Fig. (2.18) et (2.19), ont été réalisées pour étudier la précision et la robustesse des filtres dans le cas d'un mouvement UAV, tout en essayant d'identifier la combinaison idéale pour ce type de mouvement.

FIGURE 2.17 – Le drone AscTec Pelican utilisé dans les expérimentations.

Une approche type "inverse depth parametrization" (Annexe.B) est utilisée pour trianguler les points d'intérêt de l'équation de mesure (me1). Cela n'a aucun impact dans le cas de KITTI car le mouvement uniforme et rapide du véhicule rendait les points d'intérêt de la scène possibles à suivre sur trois images en moyenne avec une grande base-line. Ce n'est pas le cas dans le mouvement d'un drone où le vol stationnaire peut se produire sur de longues périodes de temps.
Comme on peut le voir dans Fig.(2.18) et (2.19), l'implémentation EKF-me1 (en bleu) donne la meilleure précision, suivie de UKF-me1 (en rouge), et enfin de UKF-me2 (en vert). Notons que le UKF-me2 était le plus précis dans les tests KITTI mais a échoué dans cette situation. Ceci souligne l'inadéquation et les limites de l'utilisation du modèle de transfert trifocal dans le MSCKF basé UKF en cas de mouvement à dynamique élevée et des scènes en vol stationnaire.
Les résultats montrent également que même le UKF-me1 qui souffre d'une corrélation d'état avec l'erreur en position des points d'intérêt est meilleur que le UKF-me2. Ce qui indique clairement que l'équation de mesure (me2) doit être évitée en cas de navigation par drone. L'explication réside dans le fait que l'équation de mesure (me1) effectue une triangulation des points d'intérêt, ce qui revient à créer localement des amers fixes dans la scène. Ceci permet au filtre de réduire la dérive. Cette référence fixe n'est pas disponible lors de l'utilisation de l'équation de mesure (me2).

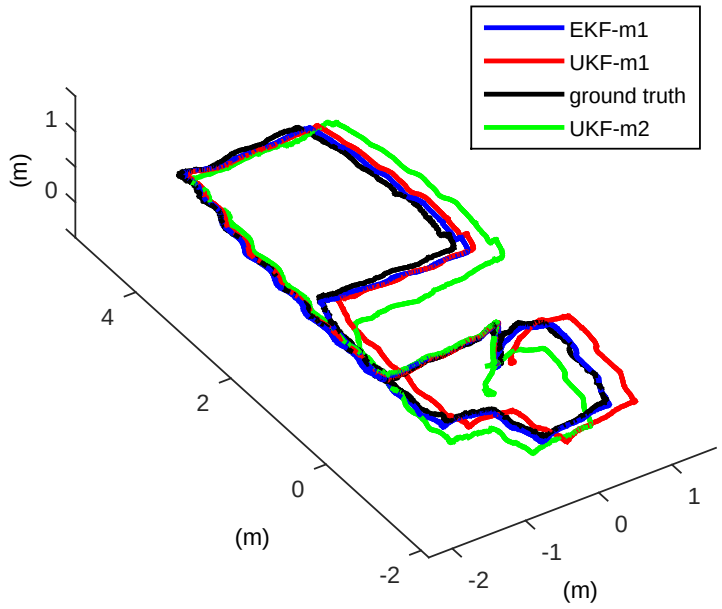


FIGURE 2.18 – Comparaison des implémentations des filtres dans une navigation UAV. La vérité terrain est obtenue par localisation Marvelmind (précision de 2cm).

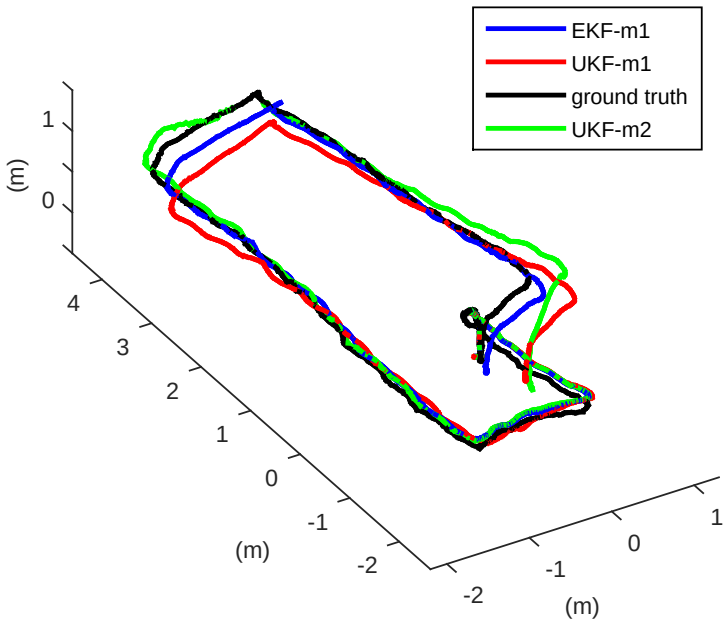


FIGURE 2.19 – Comparaison des implémentations des filtres dans une navigation UAV. La vérité terrain est obtenue par localisation Marvelmind (précision de 2cm).

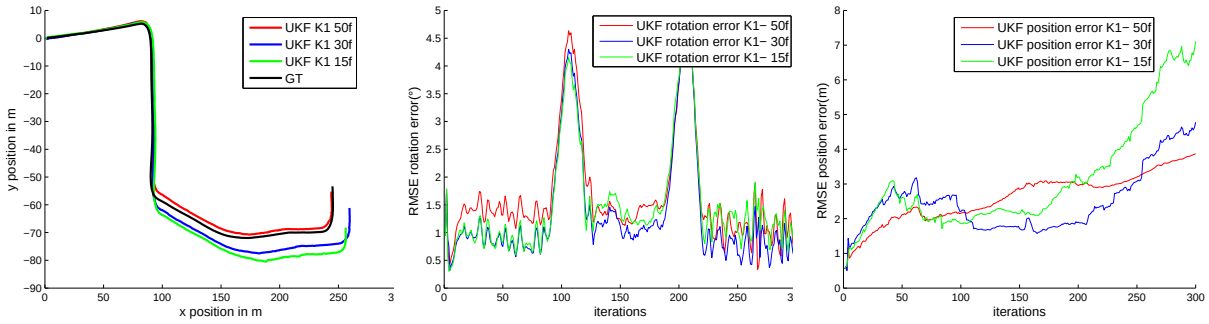
Nous pouvons conclure que la performance de l'estimation MSCKF est liée non seulement aux propriétés statistiques du filtre, mais aussi à l'adéquation du modèle de mesure à la dynamique et à la scène.

2.9.3 Comparaison de la robustesse des filtres au manque de données image

Nous étudions ici la sensibilité du MSCKF relativement au nombre et à la distribution des points d'intérêt dans l'image, et comment cela impacterait sur les performances des filtres formulés. À partir de maintenant, nous concentrons notre étude sur les meilleures combinaisons de filtres des tests précédents : EKF-me1 et UKF-me2 (ce sont également les formulations les plus utilisées dans la littérature).

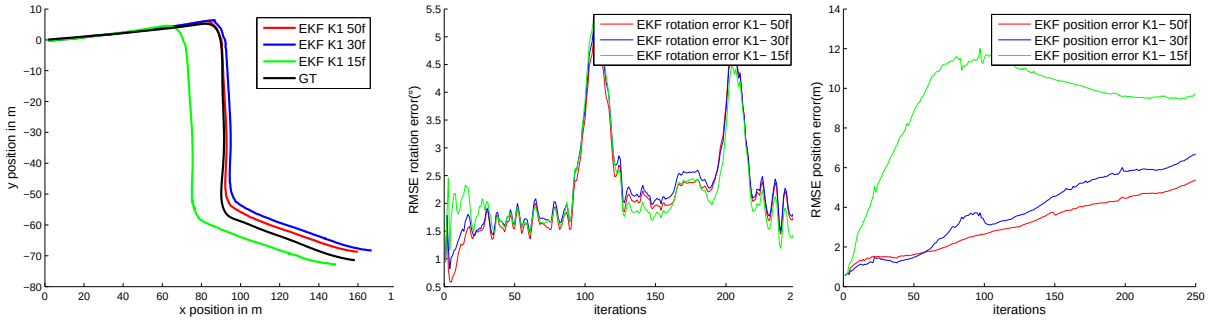
2.9.3.1 Robustesse au manque de points d'intérêt

En utilisant les jeux de données KITTI (KITTI1 et KITTI3), nous limitons à 50, 30 et 15 le nombre maximal de points d'intérêt utilisés pour la mise à jour des filtres.



(a) (b) (c)

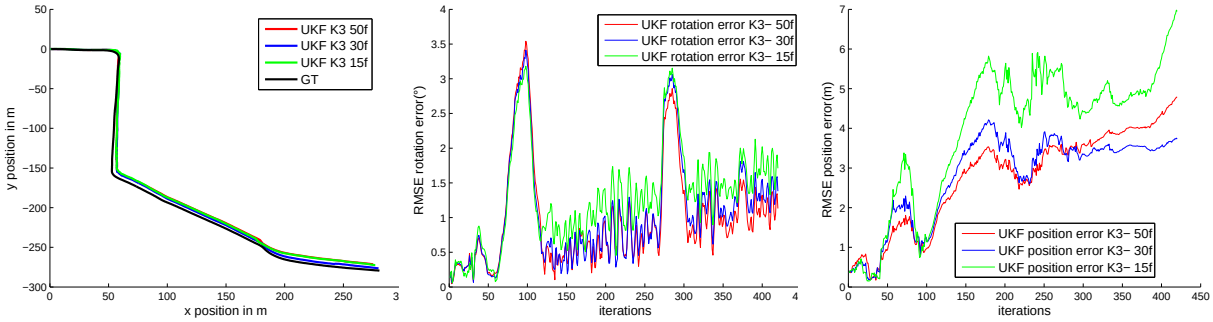
FIGURE 2.20 – Évaluation du filtre UKF-me2 sur KITTI 1 en réduisant le nombre de points. (a) Trajectoire réalisée, GT représente la vérité terrain. (b)(c) Erreur RMSE en rotation et en position de l'UKF-me2.



(a) (b) (c)

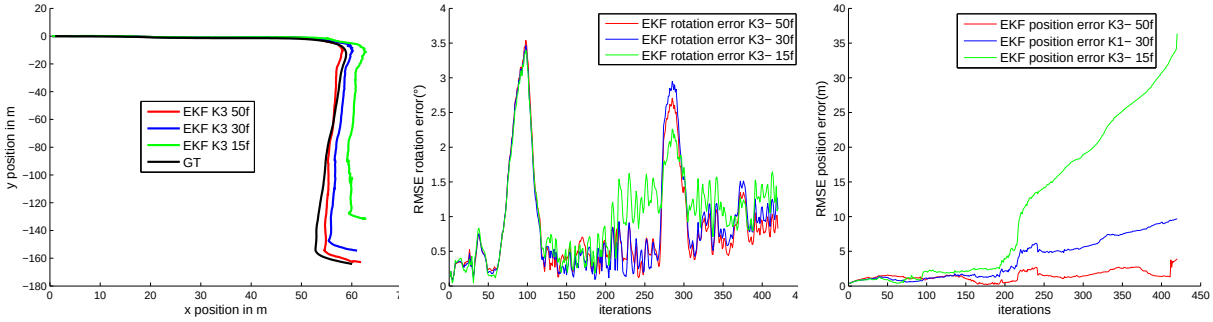
FIGURE 2.21 – Évaluation du filtre EKF-me1 sur KITTI 1 en réduisant le nombre de points. (a) Trajectoire réalisée, GT représente la vérité terrain. (b)(c) Erreur RMSE en rotation et en position de l'EKF-me1.

Nous observons globalement une dérive rapide et élevée de l'estimation des filtres lorsque le nombre de points de l'image est réduit. Les courbes (Fig.(2.20), (2.21), (2.22)(2.23)) décrivent l'évolution de l'estimation de pose. Les résultats de la comparaison



(a) (b) (c)

FIGURE 2.22 – Évaluation du filtre UKF-me2 sur KITTI 3 en réduisant le nombre de points. (a) Trajectoire réalisée, GT représente la vérité terrain. (b)(c) Erreur RMSE en rotation et en position de l'UKF-me2.



(a) (b) (c)

FIGURE 2.23 – Évaluation du filtre EKF-me1 sur KITTI 3 en réduisant le nombre de points. (a) Trajectoire réalisée, GT représente la vérité terrain. (b)(c) Erreur RMSE en rotation et en position de l'EKF-me1.

sont calibrés à la portion réussie de la trajectoire (avant divergence) quand le nombre de points utilisés est de 15 points. Pour éviter la divergence du filtre, nous avons défini 10 degrés comme l'erreur de rotation maximale tolérée, c'est à dire, si une dérive de 10 degrés est atteinte, cela signifie qu'il y a une forte probabilité que le filtre ait besoin d'une étape de ré-initialisation. Au-delà de cette limite, l'estimation du filtre est arrêtée et le filtre est considéré comme échouant dans ce pourcentage de la trajectoire.

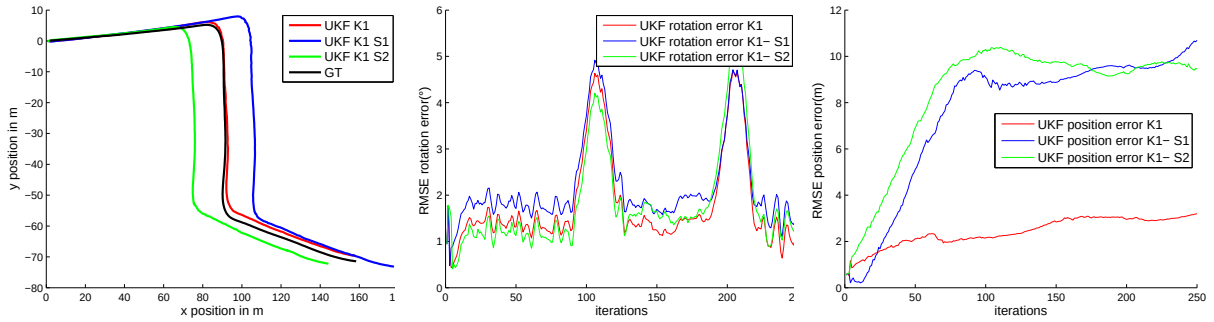
Nous pouvons voir que, pour les deux filtres, l'estimation de la rotation n'est pas sévèrement affectée lors du changement du nombre de points de 50 à 30. Ceci est également vrai pour l'estimation de la position. Cependant, lorsque le nombre est réduit à un maximum de 15 points, alors que UKF-me2 perd seulement en précision (2.20c)(2.22c), l'erreur de position de l'EKF-me1 devient rapidement très significative (2.21c) (2.23c). Un point intéressant à noter serait le pourcentage de la trajectoire atteint par chaque filtre avant de dépasser l'erreur maximale tolérée. UKF-me2, en utilisant un maximum de 15 points, a atteint 75% de KITTI 1 et 90% de KITTI 3, tandis que EKF-me1 n'a atteint que 45% de KITTI 1 et 40% de KITTI 3 (voir Fig.(2.14), (2.15) pour avoir une vue globale de la trajectoire complète).

La précision des filtres s'améliore avec le nombre de points. Nous notons également que la combinaison UKF-me2 surpasse l'EKF-me1. En effet si le premier souffre seulement du manque de points d'intérêt, ce dernier cumule en plus, l'impact d'une mauvaise triangulation des points d'intérêt.

2.9.3.2 Robustesse à la mauvaise distribution des points d'intérêt

Ici, nous comparons les filtres dans leur capacités à faire face à une mauvaise distribution des points d'intérêt dans l'image. Dans de bonnes conditions, où nous avons un nombre suffisant de points, le processus de "bucketing" assure une bonne distribution des points d'intérêt. Ce dernier consiste à diviser l'image en une grille et à choisir les meilleurs points dans chaque cellule.

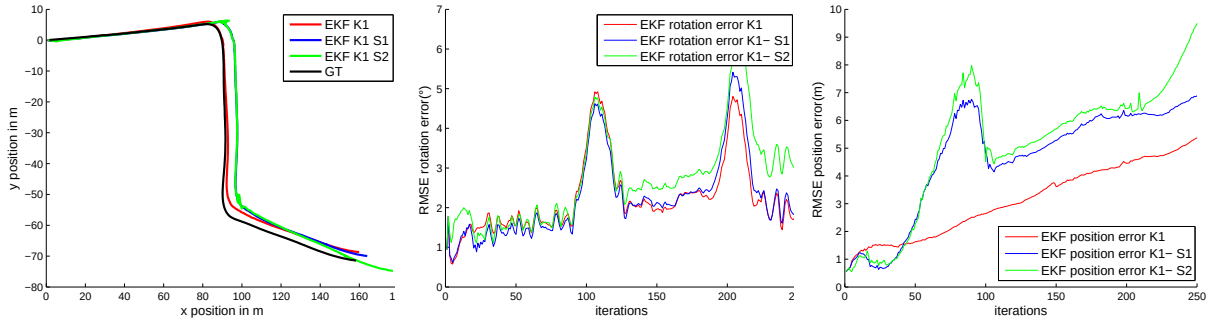
Cependant, la plupart des environnements intérieurs présentent des scènes avec des points présents seulement dans une partie de la scène. Pour évaluer les filtres dans de telles circonstances, nous considérons deux situations S1 et S2, où nous éliminons artificiellement les points sur le côté droit des images (nous prenons des points seulement des buckets de gauche). Dans S1, nous alternons une image d'une mauvaise distribution après chaque image bien répartie. Dans S2, nous considérons deux images mal distribuées après chaque image bien distribuée.



(a) (b) (c)

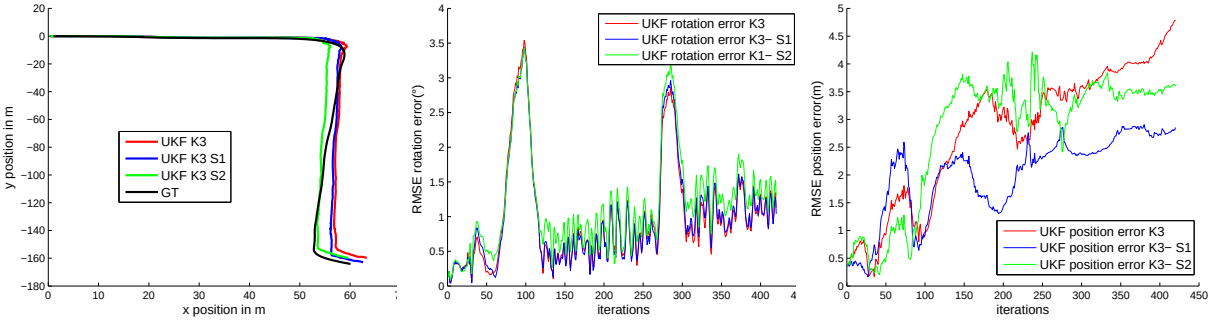
FIGURE 2.24 – Évaluation du filtre UKF-me2 sur KITTI 1 avec une mauvaise distribution des points. (a) Trajectoire réalisée, GT représente la vérité terrain. (b)(c) Erreur RMSE en rotation et en position de l'UKF-me2.

A partir des Fig.(2.24)(2.25)(2.26)(2.27), nous constatons que les deux filtres sont fortement affectés par la mauvaise distribution des points d'intérêt. L'estimation de la rotation ainsi que celle de la position sont affectées. Nous remarquons également que les filtres ont échoué à peu près au même endroit dans les deux datasets réalisant seulement 45% de KITTI 1 et 40% de KITTI 3.



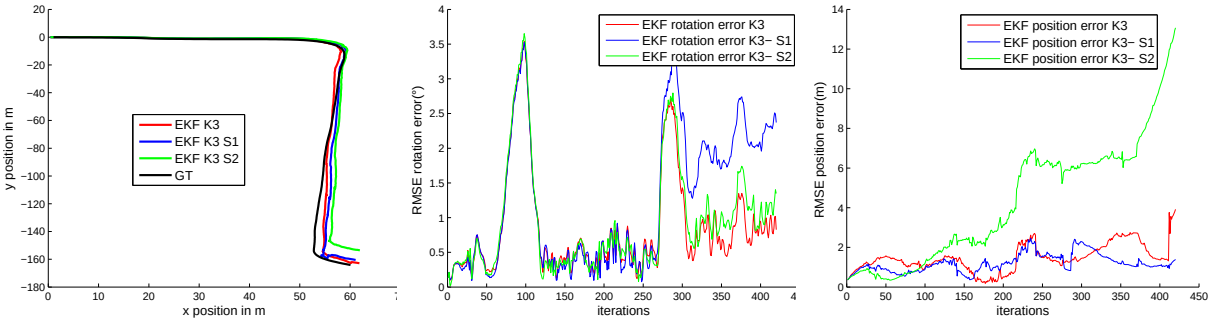
(a) (b) (c)

FIGURE 2.25 – Évaluation du filtre EKF-me1 sur KITTI 1 avec une mauvaise distribution des points. (a) Trajectoire réalisée, GT représente la vérité terrain. (b)(c) Erreur RMSE en rotation et en position de l'EKF-me1.



(a) (b) (c)

FIGURE 2.26 – Évaluation du filtre UKF-me2 sur KITTI 3 avec une mauvaise distribution des points. (a) Trajectoire réalisée, GT représente la vérité terrain. (b)(c) Erreur RMSE en rotation et en position de l'UKF-me2.



(a) (b) (c)

FIGURE 2.27 – Évaluation du filtre EKF-me1 sur KITTI 3 avec une mauvaise distribution des points. (a) Trajectoire réalisée, GT représente la vérité terrain. (b)(c) Erreur RMSE en rotation et en position de l'EKF-me1.

2.9.4 Comparaison du temps de calcul des filtres

Dans plusieurs études comparatives entre les filtres EKF et UKF, le UKF a toujours l'image d'être plus couteux en temps de calcul. Dans notre contexte, nous avons constaté que ce n'est pas toujours le cas et que cela dépend essentiellement du modèle d'observation. La raison est simplement que le UKF permet l'utilisation directe d'équations mathématiquement complexes, tandis que le EKF nous contraint à utiliser

des équations plus simples à dériver, qui peuvent ne pas être équivalentes en terme de qualité ou de temps de calcul. Dans cette implémentation, l'équation de mesure la plus rapide ne pouvait pas être utilisée avec le EKF en raison de la complexité de dérivation de ses jacobiennes. Cette contrainte sur le EKF nous oblige à utiliser une équation de mesure plus lente, ce qui a induit une compensation du faible surcoût en temps de calcul dans le UKF. Les tableaux (2.2) et (2.3) donnent une idée sur le temps de calcul de mise à jour des deux filtres.

	EKF-me1	UKF-me2	UKF-me1
KITTI 1	44.35 ms	29.76 ms	49.21 ms
KITTI 3	39.55 ms	25.78 ms	45.01 ms

Tableau 2.2 – Valeurs moyennes des temps de calcul des mises à jour des deux filtres avec différentes équations de mesures.

	EKF-me1	UKF-me1
UAV	45.01 ms	50.16 ms

Tableau 2.3 – Valeurs moyennes des temps de calcul des mises à jour des deux filtres dans l'implémentation UAV en utilisant la même équation de mesure (avec inverse depth parametrization).

Le tableau (2.2) montre, en utilisant les trajectoires KITTI, que les mises à jour du UKF-me2 sont plus rapides que celles du EKF-me1. Le gain de temps (15ms) est dû à l'utilisation d'une équation de mesure plus rapide dans le UKF-me2. Cela compense les 5ms (observable sur le tableau (2.3) où on a utilisé la même équation de mesure (me1) pour les deux filtres) que le UKF a en plus de EKF. Le tableau (2.2) montre également la différence dans le temps de calcul des mises à jour lors de l'utilisation des deux modèles de mesure dans un UKF. Nous pouvons voir comme prévu que le modèle de mesure (me2) est 20ms plus rapide que le modèle de mesure (me1).

2.10 Conclusion

Dans cette partie du travail, différentes variantes du filtrage VINS MSCKF ont été implémentées et comparées. Cette étude a couvert l'algorithme original de Mourikis basé sur le filtrage EKF avec un modèle de mesure qui triangule les positions des points d'intérêt de l'image, et un nouveau algorithme utilisant le filtrage UKF avec un modèle de mesure basé sur le transfert trifocal des points.

Le premier point à noter est la flexibilité du filtre UKF permettant de changer facilement les équations de mesure, ce qui a été remarquablement utile dans la phase de développement.

Dans l'étude comparative, nous avons considéré différents scénarios : la navigation en intérieur d'un drone et le benchmark KITTI d'un véhicule conduit à l'extérieur. Lorsque comparés en conditions "normales" à l'aide des jeux de données KITTI, nous avons trouvé que la formulation MSC-UKF utilisant le transfert trifocal surpassait dans toutes les trajectoires comparées, l'algorithme MSCKF original basé sur le EKF et le modèle de mesure basé triangulation. Cependant, dans les expériences UAV, la formulation UKF avec transfert trifocal a montré ses limites dans l'estimation de mouvement à haute dynamique et en vol stationnaire. Ces tests sur drone ont clairement montré la nécessité de choisir avec soin le modèle de mesure dans la formulation du filtre en fonction du type de mouvement.

Pour analyser d'avantage les observations, nous avons considéré deux autres scénarios, où dans le premier scénario, le nombre de points d'intérêt est diminué et, dans le second, la distribution des points est modifiée. Lorsque le nombre de points est faible, nous avons observé et expliqué pourquoi le MSCKF basé sur UKF surpassait significativement et de manière répétée le MSCKF original. En cas de mauvaise distribution des points, une situation qui peut être fréquemment rencontrée en navigation intérieure, nous avons constaté que les deux filtres étaient affectés de la même mauvaise manière.

Nous avons également évalué le temps de calcul des filtres MSC-EKF vs. MSC-UKF et trouvé que notre implémentation MSC-UKF, malgré la réputation de lenteur du filtre UKF, est plus efficace dans certaines situations, car elle nous a permis d'utiliser une équation de mesure équivalente plus rapide que celle de EKF.

Nous avons en outre souligné la forte dépendance des performances des filtres au modèle d'observation et montré que dans le cadre MSCKF, en raison de certaines contraintes structurelles, la comparaison entre EKF et UKF doit prendre en compte les contraintes sur le choix du modèle de mesure. Comme résultat de cette étude, nous pouvons conclure que l'implémentation MSC-UKF avec transfert trifocal est sûrement une amélioration par rapport à l'algorithme original dans le cas d'un mouvement rapide et fluide comme une voiture, mais pas dans une navigation de drone. Dans le cadre des VINS MSCKF, nous recommandons également l'utilisation de UKF sur EKF quand cela est possible, car cela assure au moins la même précision que EKF et offre une meilleure robustesse à certains problèmes comme la corrélation de l'état avec les erreurs de position des points d'intérêt.

Pour la suite du travail, nous avons opté pour la combinaison UKF-me1 comme filtre de localisation individuelle. Les résultats de ce chapitre vont permettre une connaissance profonde sur ce type d'algorithmes, ce qui permet des choix justifiés et adaptés pour l'étape de localisation collaborative.

Chapitre 3 :

Localisation collaborative avec un Multi-MSCKF

3.1 Introduction

La localisation d'un robot utilisant différents types de capteurs a fait l'objet d'un large éventail d'études en robotique. Plus précisément, nous avons vu que les algorithmes visuel et visuel-inertiel offrent une grande variété d'estimateurs de pose, allant des approches SLAM, utilisant uniquement des données de caméra, aux estimateurs visuel-inertiel où les données inertielles sont également intégrées pour obtenir une plus grande robustesse.

Étant donnés les progrès récents de la localisation VINS en temps réel, nous cherchons à élargir ces techniques vers le multi-robots, avec l'objectif de tendre vers une localisation collaborative efficace et une meilleure connaissance de l'environnement. Ces dernières considérations sont d'une grande pertinence dans des applications telles que le contrôle de flotte, l'exploration collaborative et la planification de missions multi-robots [Huntsberger et al., 2004][Tews et al., 2004].

3.2 Positionnement

Pour résoudre le problème de localisation collaborative, la plupart des approches utilisent principalement le filtre EKF [Huang and al, 2011] [Achtelik et al., 2011] [Piasco et al., 2016], le Moving Horizon Estimation (MHE) [Wang et al., 2014][Wang et al., 2016] ou le Maximum a Posteriori (MAP) [Nerurkar et al., 2009a].

Des résultats intéressants ont été obtenus dans [Wang et al., 2014] [Wang et al., 2016], où les auteurs ont résolu le problème de localisation collaborative en utilisant une seule balise et la structure du MHE pour inclure des contraintes sur l'état.

Dans les méthodes basées EKF, on peut citer [Achtelik et al., 2011] où les auteurs ont considéré deux robots comme un seul capteur dynamique et ont créé une plate-forme stéréo en utilisant la caméra embarquée et les données inertielles. Ce schéma de filtrage EKF est aussi utilisé pour créer une vision stéréo distribuée avec plusieurs UAV [Piasco et al., 2016]. L'un des travaux les plus pertinents en utilisant un EKF est le [Huang and al, 2011], qui, en ajoutant des contraintes d'observabilité, cherche à optimiser l'estimation en changeant le point de linéarisation dans le filtre. Basés sur des mesures sans fil entre robots, Coppola et al. [Coppola et al., 2016] ont construit un système complet qui, en échangeant les mesures proprioceptives et la puissance du signal entre les robots, permet d'éviter les collisions dans des flottes de drones de petite taille.

Alternativement, plusieurs autres études ont été menées en utilisant la sortie des algorithmes SLAM [Mur-Artal et al., 2015b], comme dans [Schmuck and Chli, 2017] [Morrison et al., 2016] [Deutsch et al., 2016], pour fournir un serveur centralisé, qui gère l'estimation collaborative et crée la carte globale. Cependant, les exigences de traitement et de stockage des méthodes SLAM collaboratives peuvent être une vraie contrainte dans les grands environnements.

Melnyk et al. [Melnyk et al., 2012] ont étendu la structure MSCKF initiale à une localisation collaborative centralisée, basée sur des mesures visuelles indirectes de caractéristiques environnementales communes. Dans [Melnyk et al., 2012], un seul vecteur d'état global est défini, qui inclut toutes les poses de tous les robots. Les résultats se concentrent sur le cas de localisation à deux robots et sur les contraintes minimales sur les points d'intérêt pour récupérer les poses des deux robots.

Dans ce chapitre, nous fournissons un cadre distribué pour étendre l'estimation MSCKF au cas multi-robots avec le minimum d'impact sur le temps de calcul de l'odométrie individuelle proposée.

Pour réaliser cet objectif, les propriétés structurelles du MSCKF sont utilisées pour créer un calcul étendu dans une approche distribuée, avec en plus l'utilisation d'une équation de mesure efficace pour la localisation collaborative. Dans ce travail, nous utilisons également les sorties du MSCKF pour extraire et intégrer des contraintes d'environnement que nous incluons comme contraintes d'inégalités. Enfin, une collaboration basée observations relatives de distance est introduite dans le but de localiser et d'initialiser la partie collaborative basée caméra. La structure proposée sera testée en utilisant des données réelles pour évaluer sa performance sur des systèmes collaboratifs aéro-terrestre dans des environnements sans GPS.

3.3 MSCKF collaboratif

Relativement à des techniques basées SLAM ou à des algorithmes denses, le MSCKF présente divers caractéristiques et avantages qui le rendent pertinent pour une extension à un cadre de localisation collaborative. D'abord, le MSCKF produit une précision relativement élevée de l'estimation de pose avec un temps de calcul significativement faible. Aussi, en maintenant une fenêtre de poses précédentes dans le vecteur d'état, il permet une intégration intuitive des nouvelles données sur une grande fenêtre temporelle. Cela sera utilisé dans la suite dans ce que nous appelons un calcul étendu. Un autre avantage de la fenêtre de poses maintenue est qu'elle permet de contourner de manière simple les problèmes de latence ou de bande passante dans la communication.

Ci-dessous, nous proposons une formulation étendue de l'estimation basée MSCKF. Notons que l'estimation collaborative est réalisée par chaque robot dans une architecture

distribuée.

Formulation et augmentation du vecteur d'état : Le premier point à définir est le choix d'un algorithme parmi ceux étudiés dans le chapitre précédent, comme base que l'on va élargir au collaboratif. D'après les résultats de l'étude précédente et d'après le type de mouvement qui nous intéresse, à savoir essentiellement la navigation de drones, le choix du modèle d'observation se porte sur l'équation de mesure basée sur la triangulation des points. Pour le type de filtre, nous allons partir sur l'implémentation UKF, vu la flexibilité et la qualité d'estimation de ce filtre. Ce choix représente le noyau de notre nouveau système, c'est-à-dire, l'estimateur de pose individuelle pour chaque robot du groupe.

Augmentation du vecteur d'état : Le MSCKF collaboratif que nous proposons est défini par a un vecteur d'état augmenté, contenant le vecteur original du MSCKF individuel, suivi de n transformations entre les repères *initiaux* des robots voisins et celui du robot local :

$$X_k^c = [\boldsymbol{X}_k \quad {}^G p_{r1}^T \quad {}^G \overline{q}_{r1}^T \quad \dots \quad {}^G p_{rn}^T \quad {}^G \overline{q}_{rn}^T]$$

où $({}^G p_{rn}^T, {}^G \overline{q}_{rn}^T)$ représente la transformation entre le repère zéro du robot local r et le repère zéro du robot voisin n, exprimée dans le repère global du robot local.
Le choix d'estimer la transformation entre les repères initiaux des robots voisins et du robot local plutôt que la pose actuelle, est justifié par le fait que l'odométrie fournie par le MSCKF individuel $\boldsymbol{X}_k$ sur chaque robot est précise sur une courte période de temps. Ainsi, l'estimation de la position du repère initial d'un robot voisin et le partage des odométries des robots, sont suffisants pour suivre avec précision la position actuelle d'un robot voisin.

Ce choix permettra au nouveau filtre d'estimer des variables à faible dynamique (qui sont supposées être fixes si l'odométrie des robots est parfaite). En effet, les odométries accumulent des erreurs et donc les repères initiaux des robots voisins vont lentement dériver pour compenser ces erreurs.

$$\text{état des voisins} : \begin{cases} {}^G\dot{p}_{r1} = 0 & {}^G\dot{\overline{q}}_{r1} = 0 \\ \vdots & \\ {}^G\dot{p}_{rn} = 0 & {}^G\dot{\overline{q}}_{rn} = 0 \end{cases} \tag{3.1}$$

Pour les raisons citées précédemment, la dynamique de la transformation entre les repères origines des robots à l'étape de la propagation est mise à zéro (Eq.3.1). Ce qui

signifie que l'évolution de la transformation entre les repères n'est réalisée que lors de la présence de mesures.

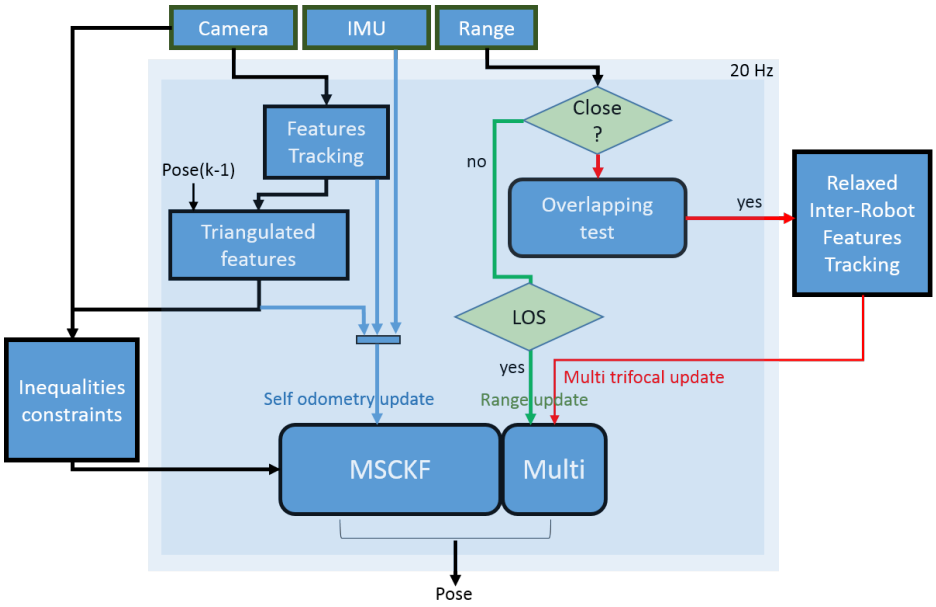


FIGURE 3.1 – Structure globale de l'algorithme proposé.

Dans ce qui suit, nous allons considérer différentes situations où des données de capteurs de distance et d'observations visuelles communes entre robots sont utilisées pour réaliser une localisation collaborative.

La Fig.3.1 montre les différents niveaux de l'algorithme proposé où les flèches bleues représentent le flux de données utilisé pour l'odométrie individuelle, en vert le fonctionnement basé sur les mesures de distance, et en rouge, les opérations impliquées dans la partie collaborative basée sur la caméra.

3.4 Localisation collaborative basée mesures de distance

Différentes études ont porté sur le type de données à utiliser dans un cadre de collaboration [Knuth and Barooah, 2015]. Les mesures de distance sont classées comme les moins efficaces si l'on recherche la précision. Cependant, ces capteurs et notamment avec la montée en précision et la miniaturisation des puces de localisation ultra large bande, sont de loin les capteurs les plus faciles à intégrer sur les drones. Motivé par cet avantage et le fait que dans notre algorithme, l'utilisation principale de ce capteur est d'obtenir une bonne initialisation des robots voisins pour préparer l'étape d'association des données de la caméra, on propose dans ce qui suit, une structure d'estimation basée distances relatives.

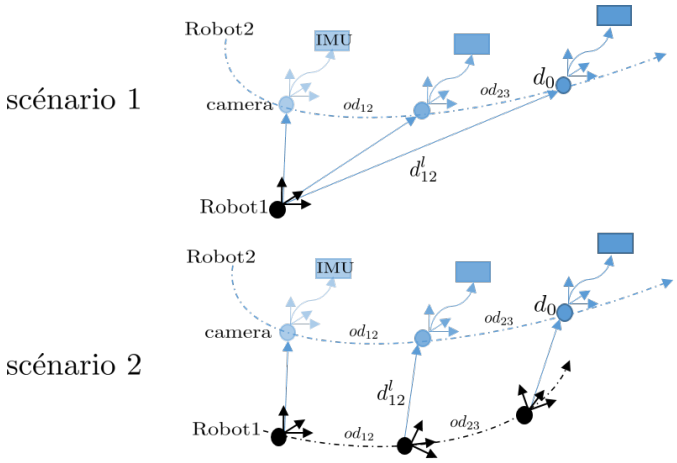


FIGURE 3.2 – Scénarios de localisation collaborative basée distance. Dans le premier scénario *(haut)*, seule la position est observable ; dans le deuxième *(bas)*, la pose complète (position et orientation) est observable.

Nous visons à laisser les drones en mouvement libre et éviter toute coordination de mouvement entre les robots. Dans ce cas, on peut rencontrer deux scénarios de configuration (Fig.3.2). Le premier scénario est lorsqu'un des drones est statique (en vol stationnaire) et observe d'autres mouvements de drones. Dans cette situation, seules les positions des repères initiaux des voisins sont observables. Dans le deuxième scénario, le drone et son voisin sont en mouvement. Dans ce cas, il est possible de récupérer la totalité de la pose du repère initial du voisin.

3.4.1 Modèle de mesure basé ULB

Une fenêtre w de distances relatives entre le robot local et son voisin est utilisée pour construire le modèle de mesure. La taille de la fenêtre w peut varier de une à plusieurs mesures espacées par des intervalles temporelles ou par des contraintes de déplacement minimum.
L'équation de mesure qui traduit l'observation relative entre le robot i et sont voisin j s'écrit :

$$z_l = h_{ULB}\left(\varphi(\tilde{X}^c, \hat{X}^c), \{d^l_{ij}\}\right) \tag{3.2}$$

$$= \| -{}^{g_i}R^T_{r_i}\,{}^{g_i}p_{r_i} + {}^{g_i}R^T_{r_i}t_{ij} + {}^{g_i}R^T_{r_i}R_{ij}\,{}^{g_j}p_{r_j} \| \tag{3.3}$$

Où φ est une fonction d'addition, et d^l_{ij} est la distance entre le robot r_i et son voisin r_j, l'indice l représente la distance utilisée du vecteur de mesures disponibles de taille w. (t_{ij}, R_{ij}) est la transformation recherchée entre les repères initiaux des robots r_i et r_j (voir Fig.3.3).

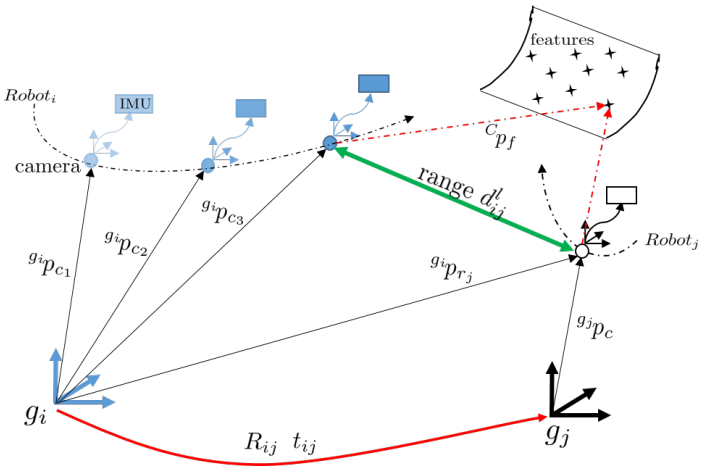


FIGURE 3.3 – Graphe d'évolution de deux robots. La ligne *verte en gras* représente une mesure de distance et donc une des w mesures de l'équation de mesure Eq.3.2 entre deux robots. Les lignes *rouges discontinues* représentent les données caméra communes. Le lien en *rouge gras*, est la transformation entre les repères initiaux à estimer.

3.4.2 Identification des NLOS

Afin d'améliorer la fiabilité des mesures de distance, nous fournissons une procédure simple pour différencier les mesures en ligne de vue LOS (Line Of Sight) de celles non en ligne de vue NLOS (Non Line Of Sight). Ceci est d'une grande importance lorsqu'on utilise des capteurs ULB car l'erreur de mesure peut varier de 20cm en ligne de vue à 60cm dans le cas d'une mesure indirecte à travers un mur porteur de 20cm d'épaisseur.

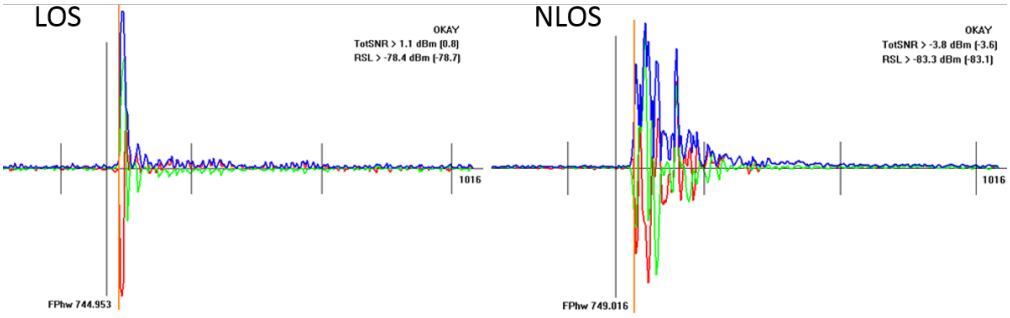


FIGURE 3.4 – CIR (channel impulse response) dans une balise ULB dans les deux conditions, NLOS (droite) et LOS (gauche). Le CIR est un tableau de valeurs complexes, réelles en rouge, imaginaires en vert et le module en bleu.

Dans la Fig.3.4, on on peut constater que la réponse impulsionnelle indique l'amplitude de l'énergie reçue le long du trajet direct et de chacun des trajets multiples qui le suivent. Pour le cas LOS, le premier chemin est clairement visible et constitue le signal le plus puissant reçu (c'est l'estimation de la distance équivalente à cette amplitude qui permet d'avoir une mesure). Dans le cas du NLOS, on voit que le graphe CIR contient une grande quantité de bruit et que la première impulsion entrante a une amplitude similaire

au reste des impulsions entrantes. Ce bruit et cette perturbation dans les amplitudes de l'énergie reçue sont la source de l'imprécision des mesures NLOS.

Pour détecter les NLOS sévères, on définit une *covariance dynamique* δ_{run} [Khodjaev et al., 2010] qui est estimée sur M mesures de distances consécutives $\hat{d}_m$ et on la compare à une variance seuil δ_{thresh} qui est la variance du capteur dans des conditions de LOS.

$$\delta_{thresh} = \sigma^2_{LOS} + \frac{M(M-1)}{12}(v_{max} t_{UWB})^2 \tag{3.4}$$

$$\delta_{run} = \frac{\sum_{m=1}^{M}(\hat{d}_m - \mu_d)^2}{M-1} \begin{cases} < \delta_{thresh} & \hookrightarrow \quad LOS \\ > \delta_{thresh} & \hookrightarrow \quad NLOS \end{cases}$$

Avec $\mu_d = 1/M \sum_{m=1}^{M} \hat{d}_m$ et t_{UWB} l'intervalle de temps de mise à jour et v_{max} la vitesse de mouvement du drone. Si la vitesse est nulle, le seuil est égal à la variance connue du capteur dans les conditions de LOS. Si le capteur est en mouvement, le second terme compense la surestimation de la variance dynamique [Khodjaev et al., 2010].

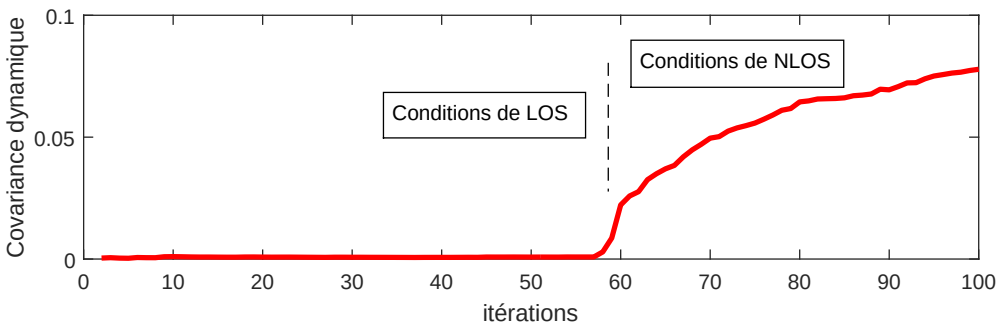


FIGURE 3.5 – Évolution de la covariance dynamique.

La Fig.3.5 montre l'évolution de la covariance dynamique avant et après le passage à une condition NLOS. On peut voir que la covariance dynamique explose au moment du passage au NLOS. Cette augmentation est expliquée par la quantité de bruit affectant les mesures NLOS Fig.3.4.

3.5 Extraction et intégration des contraintes d'environnement

Les contraintes d'environnement peuvent aider à la consistance et à l'accélération de la convergence des filtres si ces dernières sont possibles à extraire. L'étape d'intégration de ces contraintes dans le filtre est dans notre cas simplifiée grâce à la capacité de l'UKF à intégrer

des contraintes d'inégalités sur l'état. La technique d'extraction que nous proposons repose sur 3 étapes : i) l'extraction des lignes de fuite, ii) l'estimation multi-modèles, et iii) le filtrage de données.

3.5.1 Point de fuite

En géométrie perspective, la projection des lignes parallèles a une extension finie (intersection en un point) que l'on appelle "point de fuite".

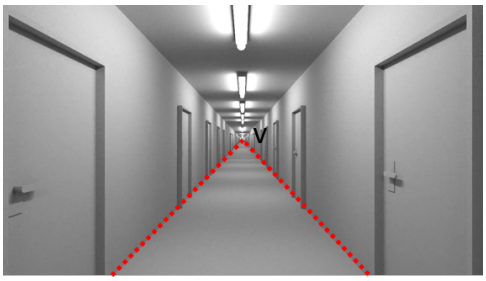


FIGURE 3.6 – Point de fuite dans un environnement intérieur.

Ce point de fuite est obtenu par l'intersection entre le plan image et une droite parallèle à la direction de la ligne et passant par le centre de la caméra. Cela signifie que la position d'un point de fuite ne dépend que de l'orientation de la caméra et non de sa position. Cela signifie également que plusieurs droites parallèles dans la scène vont avoir le même point de fuite car leur orientation est la même. Cette invariance permet de mettre des contraintes qui peuvent corriger les dérives en rotation de l'odométrie visuelle. L'estimation de point de fuite a démontré dans des travaux comme [Lee and Yoon, 2015] sa robustesse pour l'estimation de la rotation d'une caméra dans les environnements de type Manhattan.

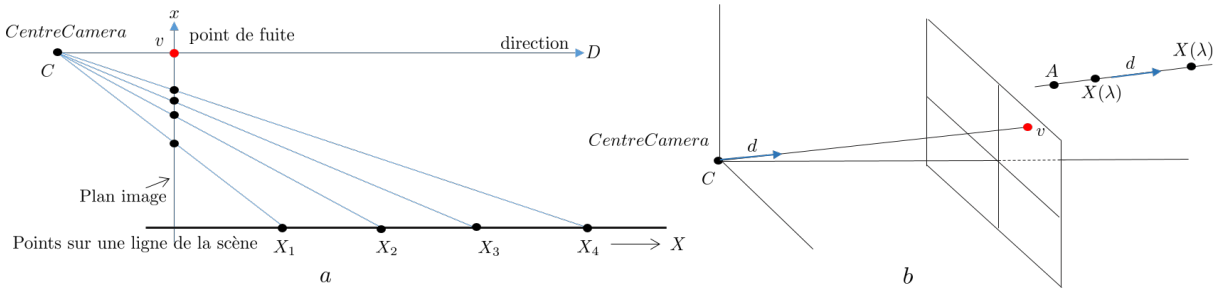


FIGURE 3.7 – Formation du point de fuite [Hartley and Zisserman, 2003].

Suivant le schéma Fig.3.7, les points sur une droite dans la scène passant par $\mathbf{A}$ et d'une direction $\mathbf{D} = (d^t, 0)^t$ peuvent être écrits comme $X(\lambda) = A + \lambda D$, avec λ allant de 0 à ∞. Le point $X(\lambda)$ varie donc du point $\mathbf{A}$ au point $\mathbf{D}$ à ∞. En utilisant le modèle projectif $P = K[I|0]$, le point $X(\lambda)$ est imagé comme $x(\lambda) = PX(\lambda) = PA + \lambda PD = a + \lambda Kd$ où a est l'image du point A. Dans ce cas, le point de fuite est obtenu mathématiquement

comme la limite du point imagé :

$$v = \lim_{\lambda \to \infty} x(\lambda) = \lim_{\lambda \to \infty} (a + \lambda K d) = K d. \tag{3.5}$$

$v = Kd$ indique que le point de fuite se reprojette comme un vecteur de direction **d**.

3.5.2 Ligne de fuite

Tout comme le lien entre les droites parallèles et le point de fuite, il existe une relation appelée ligne de fuite entre les plans parallèles dans une scène. Géométriquement, une ligne de fuite est construite en intersectant le plan image avec un plan parallèle au plan de la scène et passant par le centre de la caméra. Là aussi, il est clair qu'une ligne de fuite ne dépend que de l'orientation du plan de scène et ne dépend pas de sa position (Fig.3.8).

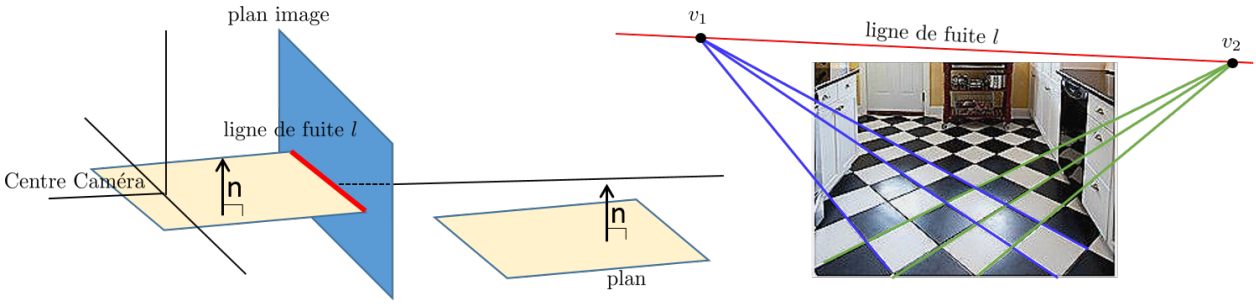


FIGURE 3.8 – Formation du ligne de fuite.

Si la caméra est calibrée, les lignes de fuite ou les points de fuite peuvent être utilisés pour extraire des informations sur l'orientation de la caméra ou pour effectuer des opérations sur l'image, comme :

- Un plan peut être rectifié métriquement avec seulement sa ligne de fuite.
- L'orientation d'un plan par rapport à la caméra peut être déterminée à partir de sa ligne de fuite.
- L'angle entre deux plans de la scène peut être déterminé à partir de leurs lignes de fuite.
- L'angle entre deux directions correspondant à deux points de fuite de la scène peut être déterminé.

3.5.3 Estimation multi-modèles

Un problème connu en vision par ordinateur est d'extraire un modèle à partir de données bruitées. Le RANSAC est une solution largement utilisée pour résoudre ce problème. Cette technique fonctionne plutôt bien lorsque les données sont bruitées ou même fortement corrompues par un bruit. Si l'ensemble des données contient plusieurs

instances du modèle, le problème est beaucoup plus complexe, car on doit faire la différence entre les "vrais" outliers et les outliers à la structure du modèle d'intérêt, mais qui sont des inliers pour une autre instance de ce modèle.

Pour résoudre ce problème, plusieurs propositions existent dans la littérature, allant de la plus simple et efficace mais non optimale [Zuliani et al., 2005] qui applique un RANSAC pour enlever les inliers et itère ensuite pour les autres instances du modèle. Ceci implique que le nombre de modèles recherchés doit être connu au préalable, ce qui peut être une forte contrainte dans certaines applications.

L'autre technique populaire est la RHT "Randomized Hough Transform" [Xu et al., 1990] qui construit un histogramme sur l'espace des paramètres où les pics dans l'histogramme représentent les modèles recherchés. Cette technique, du fait qu'elle s'appuie sur la construction d'histogramme, n'a pas besoin de la connaissance a priori du nombre de modèles à rechercher. Elle souffre cependant de deux problèmes, la précision limitée et l'inefficacité en temps de calcul.

Certaines méthodes comme dans [Toldo and Fusiello,], qui est celle que nous avons adoptée, essayent de tirer profit des avantages de la RHT en exploitant le regroupement des hypothèses, mais en évitant de travailler sur l'espace des paramètres car jugé source de complexité. La Fig.3.9 montre les résultats du regroupement des lignes par points de fuite en utilisant la méthode [Toldo and Fusiello,].

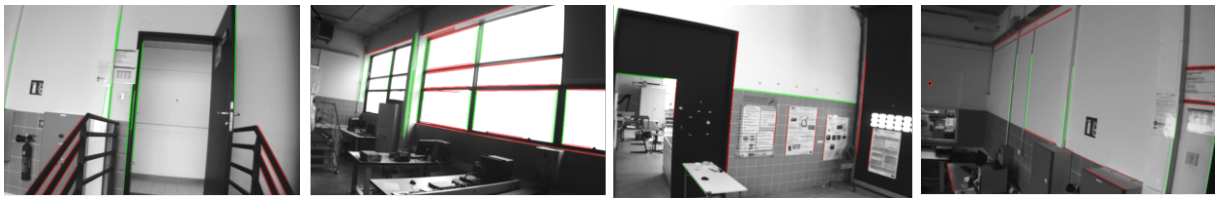


FIGURE 3.9 – Résultats de regroupement des lignes par points de fuite en utilisant un estimateur multi-modèles.

3.5.4 Extraction des contraintes

Notre objectif est d'utiliser les points d'intérêt qui sont déjà disponibles triangulés dans le filtre MSCKF, et de les combiner avec une estimation robuste des directions de fuite, voir Fig.(3.10)(3.11) pour un exemple de résultat. L'idée est de regrouper les segments par leurs points de fuite correspondants en utilisant un algorithme de "j-linkage tailored agglomerative clustering" [Tardif, 2009] [Toldo and Fusiello,] et un détecteur de segments [Gioi et al., 2010]. Une fois les directions de fuite obtenues, nous recherchons le nuage de points respectant un modèle plan de normale égale à l'une des normales aux directions de fuite (NDF) obtenues. Il en résulte une détection d'obstacles correspondants aux structures (murs) dans les environnements intérieurs. Ces obstacles fournissent des limites

de position pour les drones et peuvent donc être inclus comme contraintes d'inégalités de la forme : $x_l \leqslant x \leqslant x_H$.

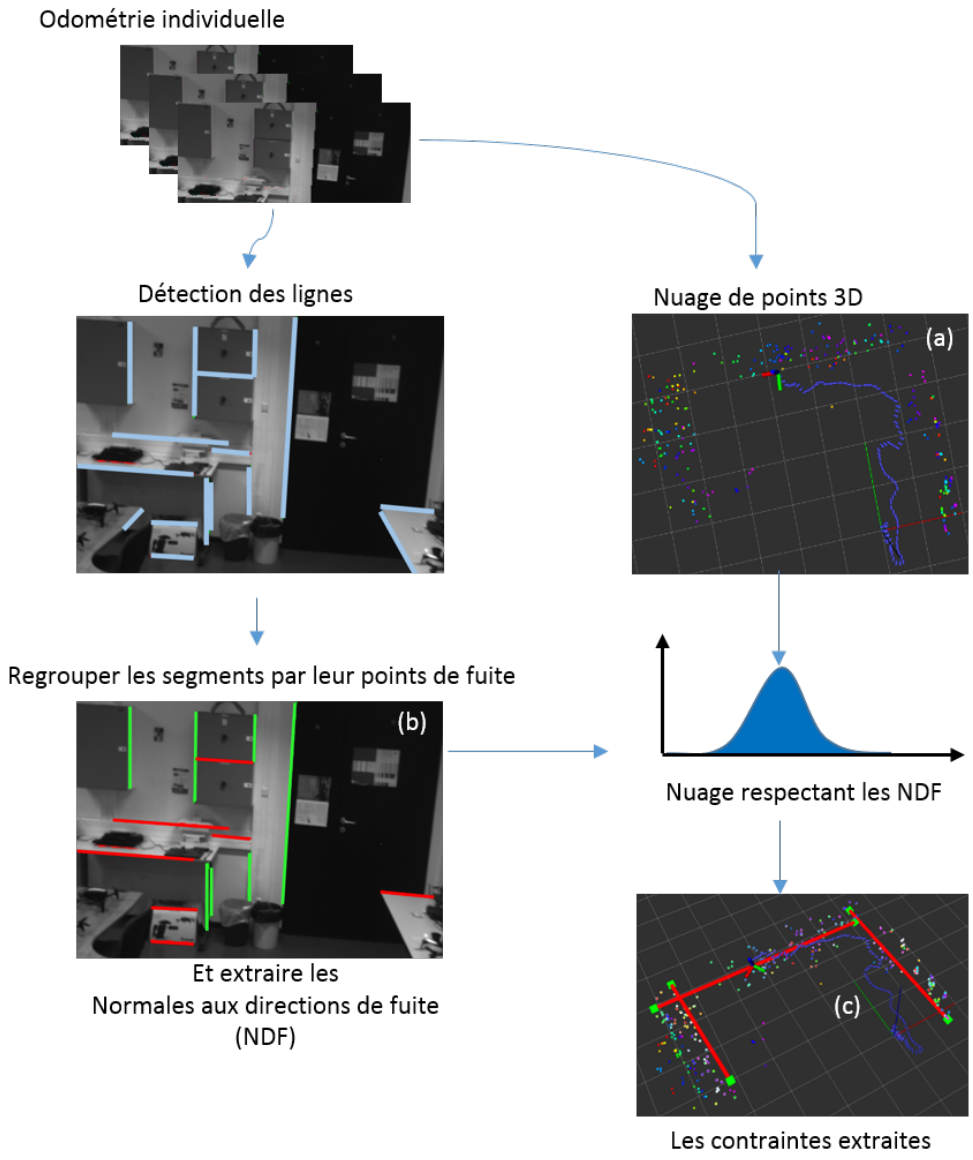


FIGURE 3.10 – Combinaison du nuage de points 3D triangulés dans le MSCKF *(a)* avec les lignes de fuite groupées *(b)*, ce qui définit les limites des contraintes d'inégalités *(c)*.

3.5.5 Troncature des données du filtre

Nous cherchons à inclure les contraintes précédemment extraites (Sec.3.5.4) dans la formulation du MSCKF collaboratif proposé. Ces contraintes peuvent être formulées sous forme de contraintes d'inégalités dans le filtre. Dans la littérature, différentes solutions pour inclure ces contraintes ont été proposées pour le filtrage UKF. Cette multitude de choix vient du fait que dans la partie propagation d'un filtre UKF, on peut soit adopter la méthode "Unscented Transform" (UT) ou "Interval Constrained UT" (ICUT), et que l'étape d'assimilation des données peut, elle aussi, suivre l'une des approches suivantes : une mise à jour basée Kalman classique (KF), une mise à jour basée Constrained Kalman (CUKF), l'approche par points Sigma contraints (SIUKF) ou la mise à jour basée Kalman classique suivie soit par une troncature, soit par une projection.
Ces variations dans la propagation et l'assimilation des données font que le type de contraintes (linéaire/non linéaire, égalité/inégalité) possible à intégrer d'une implémentation à l'autre, n'est pas le même, mais aussi, que l'étape dans laquelle l'intégration est possible change d'une structure à l'autre.

	Propagation		Mise à jour		Troncature/Projection	
	$\hat{x}_{k\|k-1}$	$\hat{P}_{k\|k-1}$	$\hat{x}_{k\|k}$	$\hat{P}_{k\|k}$	$\hat{x}^t_{k\|k}$ ou $\hat{x}^p_{k\|k}$	$\hat{P}^t_{k\|k}$ ou $\hat{P}^p_{k\|k}$
UKF	X	X	X	X	-	-
SIUKF	✔	✔	✔	✔	-	-
CUKF	X	X	✔	X	-	-
CIUKF	✔	✔	✔	X	-	-
IUKF	✔	✔	X	X	-	-
TUKF	X	X	X	X	✔	✔
TIUKF	✔	✔	X	X	✔	✔
PUKF	X	X	X	X	✔	-

Tableau 3.1 – Troncature dans les filtres Kalman basés Unscented Transform [Teixeira et al., 2008].

En se référant au tableau (3.1), on peut constater que les trois derniers filtres (TUKF, TIUKF et PUKF) sont des filtres à trois étapes et qu'ils nécessitent une opération de plus à effectuer après la mise à jour du filtre. Sur ces trois derniers, deux d'entre eux n'intègrent les contraintes que dans cette étape supplémentaire. Ce qui est intéressant dans notre étude, car cela permet d'ajouter à notre filtre MSCKF collaboratif un bloc qui n'intervient qu'à la fin des opérations du filtre. Cette modularité nous permet de garder les mêmes caractéristiques de notre filtre. Entre les deux implémentations permettant cette modularité (TUKF et PUKF), il est préférable de choisir le TUKF car pour le même temps de calcul, la contrainte est intégrée dans d'état et dans la covariance. Un autre avantage de l'algorithme TUKF sur le SIUKF, CIUKF, CUKF, et le PUKF est qu'il évite de calculer explicitement une solution en ligne du problème d'optimisation contraint.

Soit S échantillons de l'état $X^c_{k,i}$ $i = 1, 2, ..., S$ extraits en utilisant la matrice de covariance $P_{k|k}$ résultante de l'étape de mise à jour. Cela va créer deux groupes : l'un respectant les contraintes d'environnement $\mathcal{Q}_k$ que l'on note $X^q_{k,i}$ avec $i = 1, 2, ..., S^q$, et l'autre groupe ne respectant pas ces contraintes. La moyenne et la covariance de la nouvelle distribution peuvent être estimées par :

$$\begin{aligned} \tilde{X}^q_{k|k} &\approx \frac{1}{S^q}\sum_{i=1}^{S^q}\tilde{X}^q_{k,i} \\ P^q_{k|k} &\approx \frac{1}{S^q}\sum_{i=1}^{S^q}(\tilde{X}^q_{k,i} - \tilde{X}^q_{k|k})(\tilde{X}^q_{k,i} - \tilde{X}^q_{k|k})^T \end{aligned}$$

À noter qu'une limitation de cette technique est que le groupe $X^q_{k,i}$ peut contenir un petit nombre d'éléments, si le nombre d'échantillons est choisi trop petit. Pour éviter ce problème, l'échantillonnage d'importance "Importance Sampling" qui permet d'échantillonner à partir d'une nouvelles distribution (distribution d'intérêt) peut être utilisé.

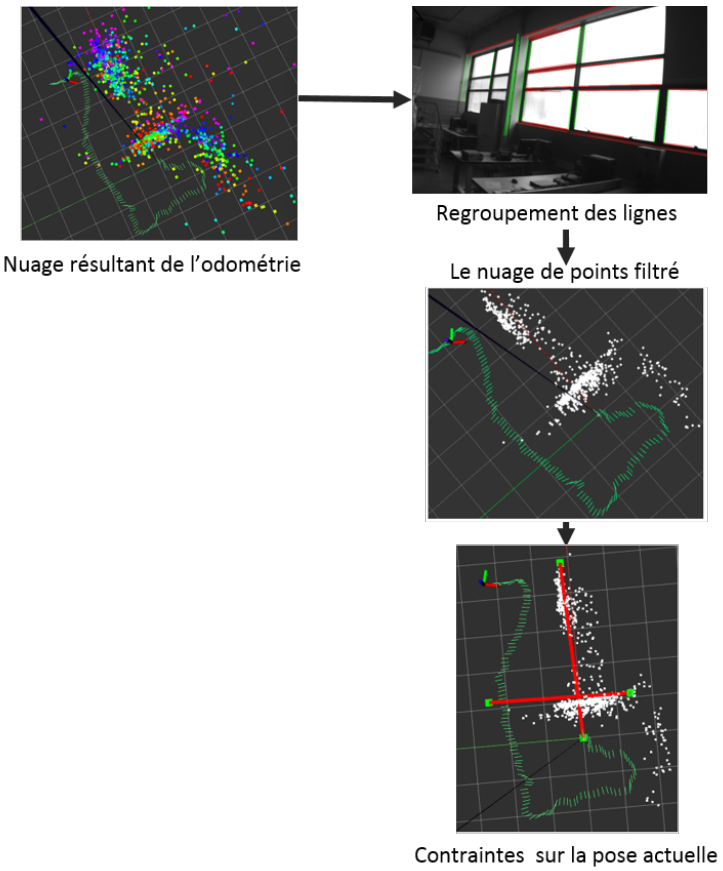


FIGURE 3.11 – Résultat de la combinaison du nuage de points 3d avec les lignes de fuite groupées pour définir des contraintes d'inégalités.

3.6 Localisation collaborative basée-vision

L'observation de la même scène par deux robots permet de trouver la transformation entre les deux si une mise en correspondance entre les points d'intérêt est possible.
Dans ce travail, le déclencheur des mises à jour MSCKF collaboratives basées-caméra est le capteur de distance indiquant une distance entre robots en dessous d'un seuil spécifié. Cela va lancer une *intention* de mise à jour collaborative, qui envoie deux images du drone local et deux autres (on se positionne dans la situation observable [Melnyk et al., 2012]) partagées par le drone voisin à un processus de calcul de faible priorité (Fig. 3.12). Dans ce processus, un calcul de faible priorité, que nous avons appelé *calcul étendu* est réalisé (sur le robot qui réalise une mise à jour collaborative). Il commence en premier par un test de chevauchement entre les images en utilisant des Fast Bags of Binary Words [Galvez-López and Tardos, 2012]. Si le score est élevé, une mise en correspondance de points inter-robots est lancée, tandis que la localisation individuelle continue son exécution en haute priorité et à fréquence fixe (inchangée). Le résultat sera une mise en correspondance de points, reçue plusieurs itérations après l'intention de mise à jour collaborative, ce qui signifie que la mise en correspondance correspond maintenant à des poses anciennes dans le vecteur d'état.

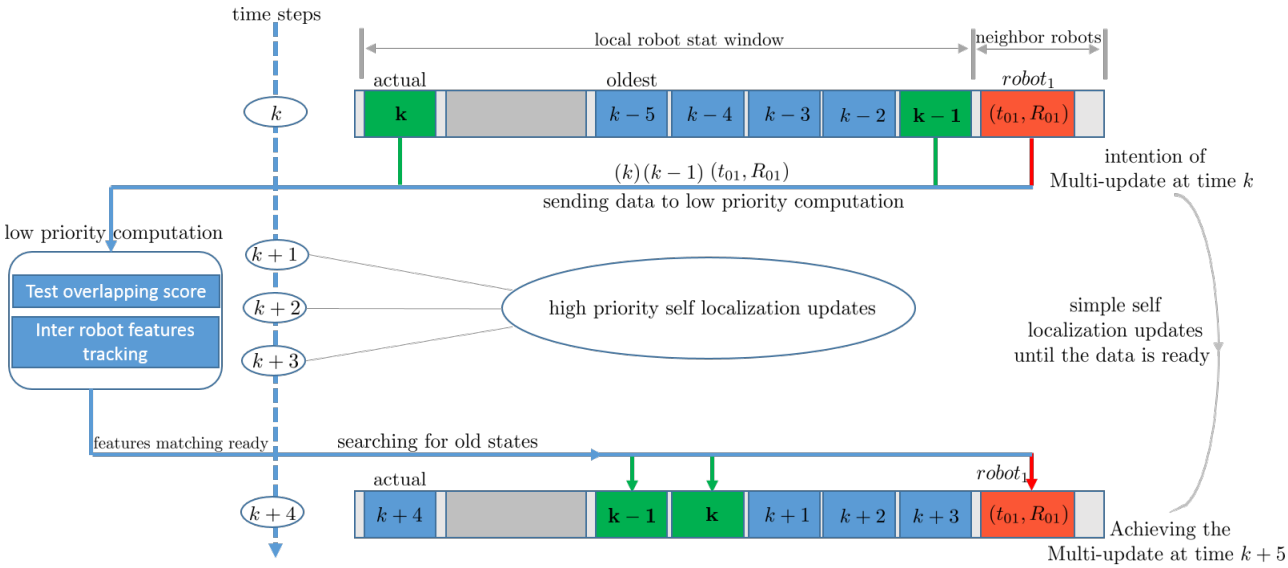


FIGURE 3.12 – Illustration du principe et des différentes opérations comprises dans la préparation "étendue" des données et la mise à jour basée-caméra. A l'instant k une intention de mise à jour est reçue, les images du robot local (en vert) et du robot voisin (en rouge) sont transmises à un processus de calcul à faible priorité. A l'instant $k+4$ les données envoyées à la préparation sont prêtes mais elles correspondent à des anciens états du filtre.

À ce stade, grâce à la structure MSCKF, qui maintient une large fenêtre de poses dans le vecteur d'état, le filtre possède un temps suffisant pour la préparation des données en

mode *calcul étendu* avant la disparition des poses correspondantes aux données préparées. Les résultats du processus de calcul étendu peuvent être intégrés tant que les poses correspondantes aux données préparées se trouvent encore dans le vecteur d'état.

Cette structure collaborative du MSCKF gère la préparation des données pour réduire son impact sur la localisation individuelle. L'algorithme effectue la mise à jour interrobots basée chevauchement de données des caméras à faible fréquence ; cela va de pair avec notre choix d'estimer les repères initiaux des robots voisins au lieu de leur poses actuelles.

Pour la mise à jour basée-caméra, on a décidé d'utiliser l'équation de mesure basée sur le transfert trifocal des points d'intérêt Eq.2.19. Cette équation de mesure déjà testée dans le chapitre précédent, est plus rapide que l'équation de mesure traditionnelle basée sur la triangulation. On a aussi constaté que cette équation de mesure donne de mauvais résultats lorsqu'elle est utilisée sur des mouvements de drone. Ce problème ne se pose pas ici, car on ne l'utilisera que dans les moments de localisation collaborative des drones pour effectuer au maximum deux, voire trois, mises à jour, ce qui est loin du contexte d'odométrie individuelle. Lorsque ce modèle de mesure est utilisé dans ce contexte de collaboration, on exploite l'avantage du temps de calcul sans l'inconvénient de la dérive.

Pour effectuer cette mise à jour, on écrit trois contraintes épipolaires et deux transferts trifocaux à partir des quatre images préparées dans le calcul étendu. L'équation de mesure est alors écrite de la manière suivante, pour deux images du drone local et une image du drone voisin dénommée image x (la même équation est écrite pour la deuxième image du robot voisin) :

$$z_j = h_2\left(\varphi(\tilde{X}^c, \hat{X}^c), \{m_1, m_2, m_x\}_j\right) \tag{3.6}$$

$$= \begin{bmatrix} \tilde{m}_2^T \, {}^{g_i}R_{12}^T \lfloor t_{12}\times \rfloor \tilde{m}_1 \\ \tilde{m}_x^T \, {}^{g_i}R_{2x}^T \lfloor t_{2x}\times \rfloor \tilde{m}_2 \\ \left(\sum_{i=1,2,x} \tilde{m}_{1i} T_i^T\right) l_2 \end{bmatrix} \tag{3.7}$$

$\tilde{m}_1, \tilde{m}_2, \tilde{m}_x$ représentent les coordonnées normalisées de la position du point commun entre deux images locales et une voisine. (R_{12}, t_{12}) et (R_{2x}, t_{2x}) sont les rotations et les translations relatives entre les images utilisées dans cette mise à jour.

Les deux premiers éléments de l'équation de mesure Eq.(3.7) représentent les contraintes épipolaires entre les images. Le troisième élément est la position normalisée du point dans la troisième image, estimée en utilisant sa position dans la première image et le tenseur trifocal :

$$\tilde{m}_x = (\sum_{i=1,2,x} \tilde{m}_{1i} T_i^T) l_2 \tag{3.8}$$

Nous référons le lecteur au chapitre précédent pour plus de détails sur le tenseur T_i.
Dans le cas hétérogène les caméras des robots vont avoir des matrices intrinsèques différentes. Le fait que nous utilisons la position normalisée des points d'intérêt, garantit l'hétérogénéité de notre modèle de mesure. La matrice de calibration K, visible dans l'équation de mesure Eq.2.19 dans le cas de navigation d'un seul robot n'est plus utilisée dans le modèle multi-robot Eq.3.7.
Enfin, la mise à jour de l'état et de la matrice de covariance est obtenue en utilisant la même approche de points Sigma détaillée dans le chapitre précédent.

$$\tilde{X}^{c,l}_{k|k-1} = \tilde{X}^{c}_{k|k-1} \pm \left(\sqrt{(L+\lambda)P_{k|k-1}}\right)_l \tag{3.9}$$

Un nuage de $2L+1$ Sigma points est créé dans l'équation précédente (L est la dimension du vecteur d'état et λ un paramètre d'échelle). Dans l'étape suivante, on injecte chacun des points Sigma dans l'équation de mesure pour avoir une estimation de mesure.

$$\begin{aligned} Z^l_j &= h_2\left(g\left(\hat{X}^c_{k|k-1}, \tilde{X}^{c,l}_{k|k-1}\right), (m_1, m_2, m_3)_j\right) \\ \hat{z}_j &= \sum_{l=0}^{2L} W^l_s Z^l_j \end{aligned} \tag{3.10}$$

Le gain K_k du filtre est calculé par :

$$\begin{aligned} P_{z_j z_j} &= \sum_{l=0}^{2L} \left(Z^l_j - \hat{z}_j\right)\left(Z^l_j - \hat{z}_j\right)^T + R \\ P_{x z_j} &= \sum_{l=0}^{2L} W^l_c \left(\tilde{X}^{c,l}_{k|k-1} - \tilde{X}^c_{k|k-1}\right)\left(Z^l_j - \hat{z}_j\right)^T \\ K_k &= P_{x z_j} P^{-1}_{z_j z_j} \end{aligned}$$

La mise à jour de l'état et de la covariance est donnée par :

$$\begin{aligned} \tilde{X}^c_{k|k} &= \tilde{X}^c_{k|k-1} + K_k\left(z_j - \hat{z}_j\right) \\ P_{k|k} &= P_{k|k-1} - K_k P_{z_j z_j} K^T_k \end{aligned}$$

3.7 Résultats expérimentaux

Dans cette partie du travail, le multi-robots impose le passage dans un premier temps par un simulateur pour tester notre algorithme. Un simulateur a l'avantage de pouvoir donner une réalité terrain dans n'importe quelle condition, sans avoir à disposer de systèmes coûteux comme le VICON. Il permet aussi l'intégration de presque tous les

capteurs utilisés en robotique et dans des environnements et des conditions de notre choix. Enfin, il permet de réaliser des expérimentations qui, dans des tests réels, nécessiteraient la mobilisation de moyens importants et de plusieurs personnes souvent pour plusieurs journées.

3.7.1 Évaluation de la localisation collaborative basée ULB en environnement simulé

Nous utilisons le simulateur Gazebo qui nous a permis la simulation de la dynamique de drones avec la possibilité d'embarquer des capteurs ULB, une caméra et une centrale inertielle. Gazebo permet la publication des données capteurs sous format ROS, ce qui nous a permis de tester sans modification notre implémentation, qui de base est sous ROS Kinetic (ROS : Robot Operating System) et utilise RViz pour l'affichage des résultats (RViz est un outil de visualisation 3D pour ROS).

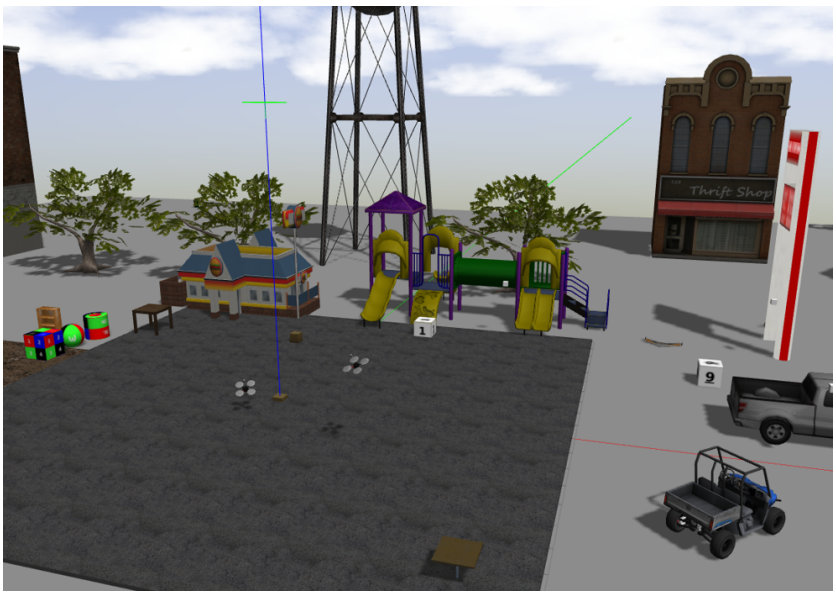


FIGURE 3.13 – Environnement Gazebo construit pour les tests multi-drones.

La Fig.3.13 montre l'environnement utilisé pour effectuer les tests. Dans cet environnement, plusieurs objets sont inclus pour créer de la texture nécessaire pour le fonctionnement de notre algorithme de localisation.

Dans cette partie, on testera le module de localisation collaborative basée ULB, la caméra n'est utilisée que pour la localisation individuelle.

Dans **le premier test** Fig.3.14, on utilisera deux drones, un en vol stationnaire et l'autre effectuant une trajectoire imposée afin de tester la convergence du filtre pour l'estimation

de la position du drone voisin. L'objectif est aussi de tester la quantité de mouvement nécessaire pour la trilatération de la position du robot voisin. Pour faire l'analogie avec la trilatération, on veut savoir si le drone doit impérativement prendre des mesures autour du voisin (360 degrés), comme c'est le cas du positionnement des balises d'un système de localisation à infrastructure ou si des petits déplacements seront suffisants pour atteindre une bonne précision.

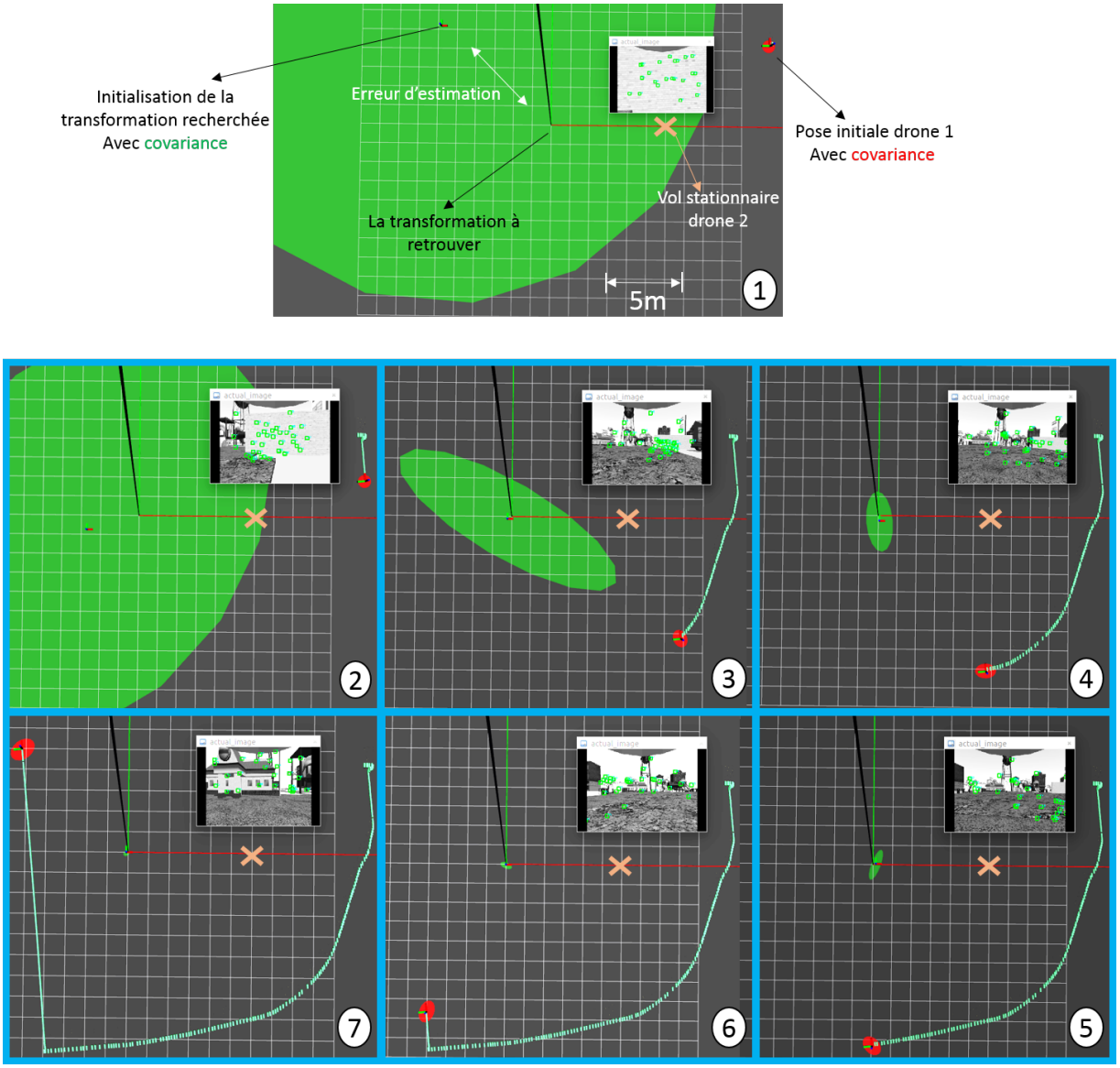


FIGURE 3.14 – Évaluation de la convergence en position sur simulateur. La trajectoire (verte) est la trajectoire effectuée par le drone en mouvement. La croix (jaune) représente la position du vol stationnaire de l'autre drone.

Dans la Fig.3.14①, on peut voir la position du drone en vol stationnaire, montrée par une croix jaune. Le drone en mouvement commence la trajectoire imposée avec une covariance montrée par un ellipsoïde rouge. L'initialisation de la transformation à estimer est aussi visible sur la Fig.3.14① avec un ellipsoïde d'incertitude vert.

De la Fig.3.14① à ⑦ le drone mobile effectue une trajectoire autour du drone en vol stationnaire. Il est possible dans ces figures, de voir la trajectoire parcourue et la convergence de la transformation recherchée et l'incertitude de l'estimation.

Les mesures de distance (ULB) entre drones sont prises espacées, par un déplacement minimum de 1m sur la trajectoire effectuée par le drone en mouvement. Le MSCKF collaboratif fonctionne sur le drone mobile (trajectoire verte), c'est ce drone qui estime la position de l'autre.

On peut constater dans les résultats de cette expérimentation que la convergence du filtre ne nécessite pas de faire le tour du robot voisin. Certes la prise ⑦ de la figure 3.14 permet d'atteindre une précision au centimètre, mais il est aussi vrai qu'a partir de la prise ④ et ⑤, on obtient une assez bonne estimation de la transformation recherchée. Ce résultat est encourageant puisque cette expérience contient le mouvement d'un seul robot, le mouvement simultané des deux robots permettra d'avoir une plus grande quantité d'information.

Dans **le deuxième test**, on vise à faire la même étude mais sur l'estimation de la transformation complète (position et orientation). Pour que la rotation soit observable, les deux robots doivent être en mouvement.

La Fig.3.15① montre les mêmes informations que dans le test 1. Les Fig.3.15① à ⑦ montrent l'évolution des deux drones qui effectuent tous les deux des trajectoires imposées (pas de vol stationnaire). Le MSCKF collaboratif fonctionne sur le drone mobile (trajectoire verte), c'est ce drone qui estime la pose de l'autre drone (trajectoire rouge).

Nous constatons (Fig.3.15) que la convergence est plus lente que dans le premier test et que l'estimation finale est moins précise que dans le premier cas si l'on additionne la longueur des trajectoires effectuées par les deux drones. La rotation est la première à converger et de manière très rapide (prise ③ de la Fig.3.15). Ceci sera d'une grande utilité dans la suite vue que dans des conditions réelles, la fenêtre de prise de mesures peut être très courte.

On remarque aussi que les petites erreurs d'estimation de la partie orientation de la transformation recherchée, impactent directement l'estimation de la partie position de la transformation. Ce deuxième test confirme, qu'il n'est pas impératif que le drone estimateur contourne le voisin pour avoir une convergence du filtre.

À la fin de ces deux tests, on peut pointer un inconvénient ressenti dans notre simulation. Malgré notre tentative de contourner le problème de la texture en ajoutant des objets dans la scène, le problème de manque et de similitude de la texture persiste surtout sur l'objet sol.

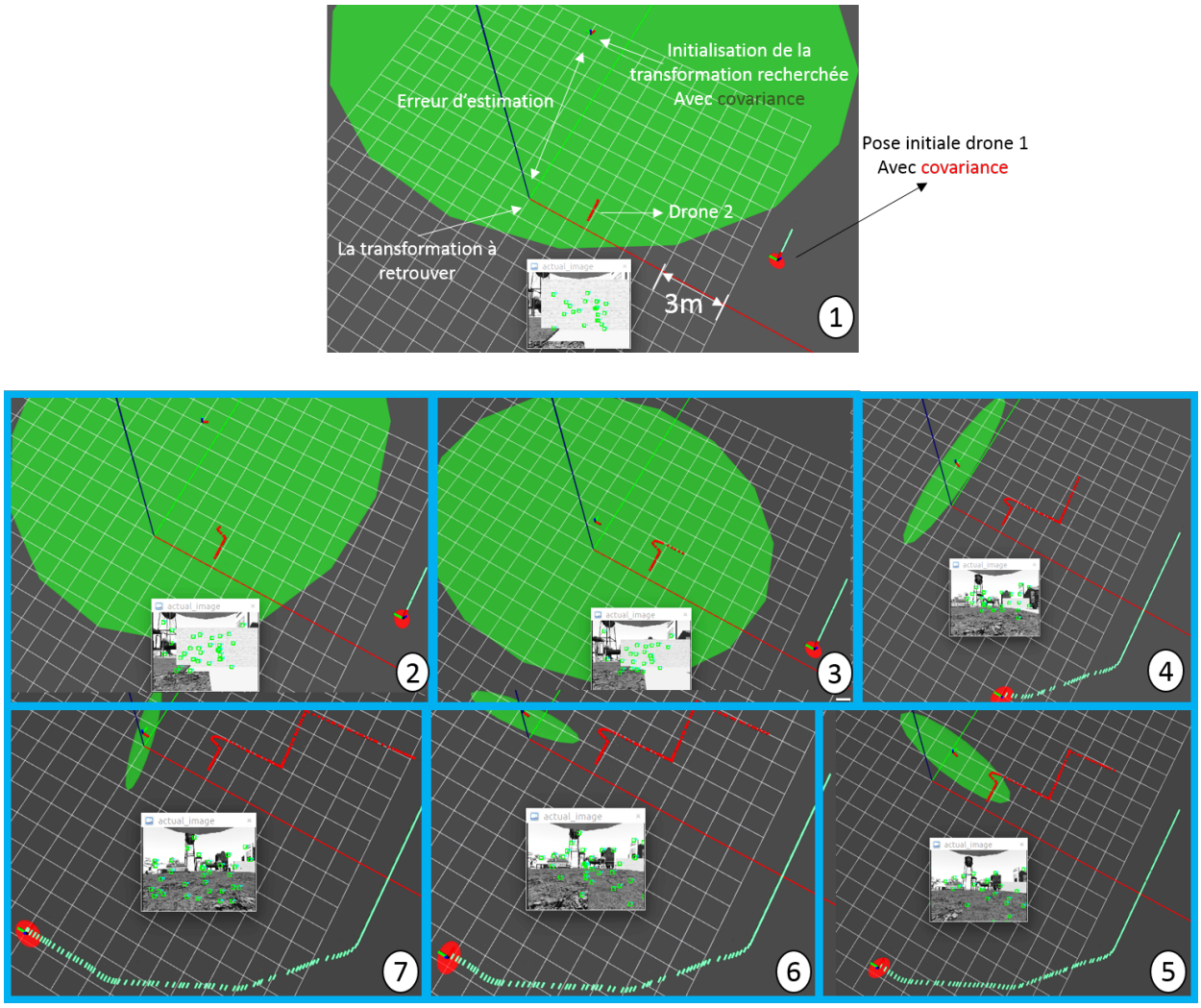


FIGURE 3.15 – Évaluation de la convergence de pose globale sur simulateur. Les trajectoires (rouge) et (verte) sont les trajectoires effectuées par les deux drones.

Dans ce qui suit, notre objectif est de valider notre algorithme dans des conditions réelles et ensuite de pousser notre analyse aux différentes parties constituant notre proposition.

3.7.2 Évaluation de la localisation collaborative basée-distance ULB en conditions réelles

Dans cette expérience Fig.3.16, on utilisera deux drones Bebop2 et le robot mobile Pioneer-3AT Fig.3.17. Les trois robots sont équipés par des balises Marvelmind pour estimer la réalité terrain de la position et un capteur ULB (3Hz) pour avoir des mesures de distance relatives. Le robot mobile terrestre qui va contenir un MSCKF collaboratif utilise une caméra "Bluefox" (20Hz) et une IMU "Xsens" (100Hz). Dans cette expérience, les drones bebop2 n'embarquent pas de calculs de pose. Il utiliseront la position fournie par le système Marvelmind comme odométrie individuelle et donc transmettent leurs positions et

leurs distances relatives au robot mobile terrestre qui va estimer la transformation relative entre la pose de son repère origine et la pose du repère origine de l'odométrie des drones. Dans cette expérimentation, les données caméra du robot terrestre ne sont utilisées que pour la partie estimation de l'odométrie individuelle. Un ROS Network est utilisé pour partager les données entre les robots. Les mesures de distance (ULB) entre robots, sont prises espacées, par un déplacement minimum de 1m sur la trajectoire effectuée par le robot terrestre.

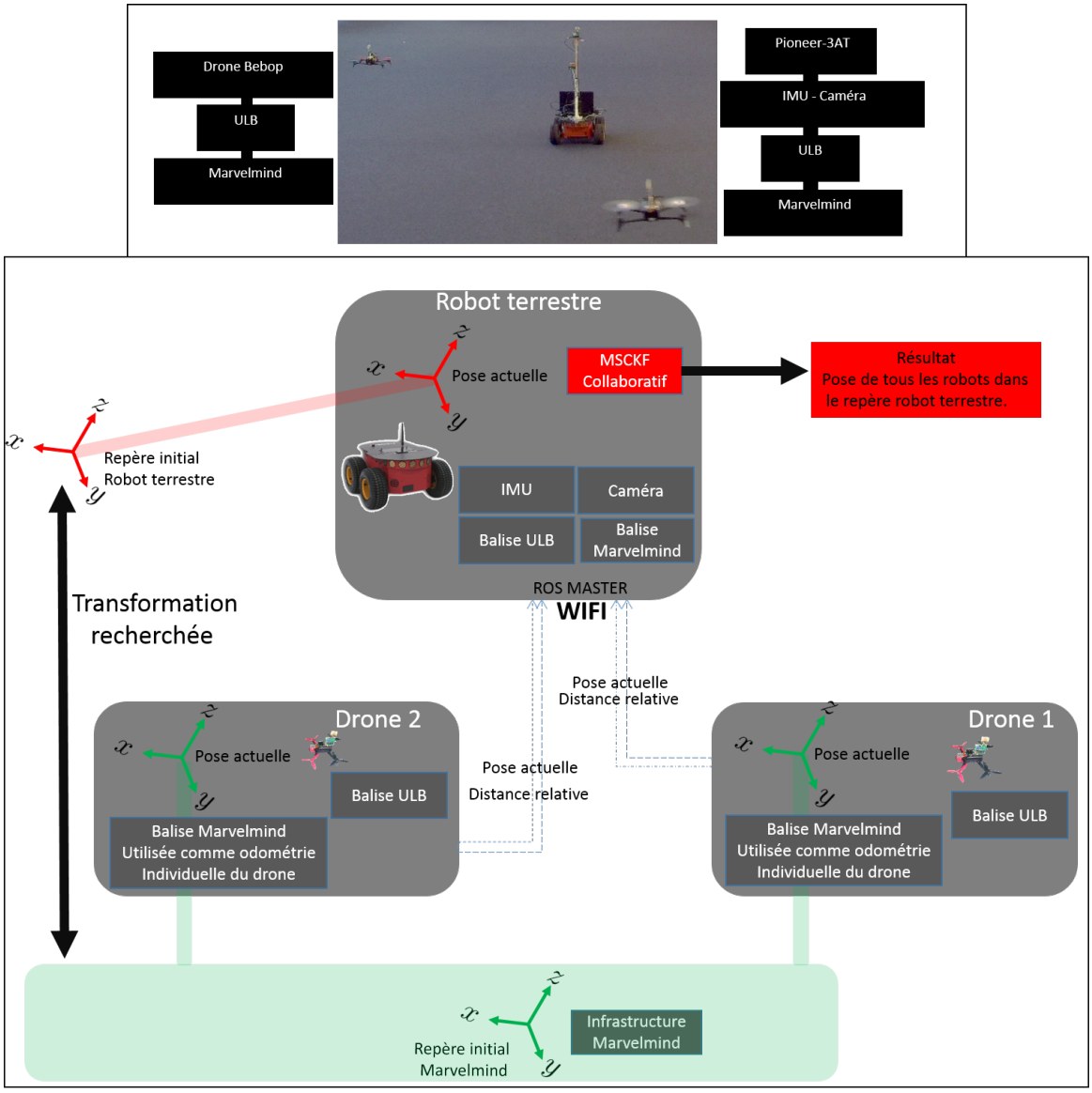


FIGURE 3.16 – Équipement utilisé dans l'expérimentation de la localisation collaborative basée-ULB.

La Fig.3.18 montre les trajectoires estimées des drones en utilisant l'algorithme de localisation collaborative, ainsi que l'évolution de l'odométrie du robot mobile terrestre. Ces résultats sont cohérents en prenant en compte que les odométries données au filtre

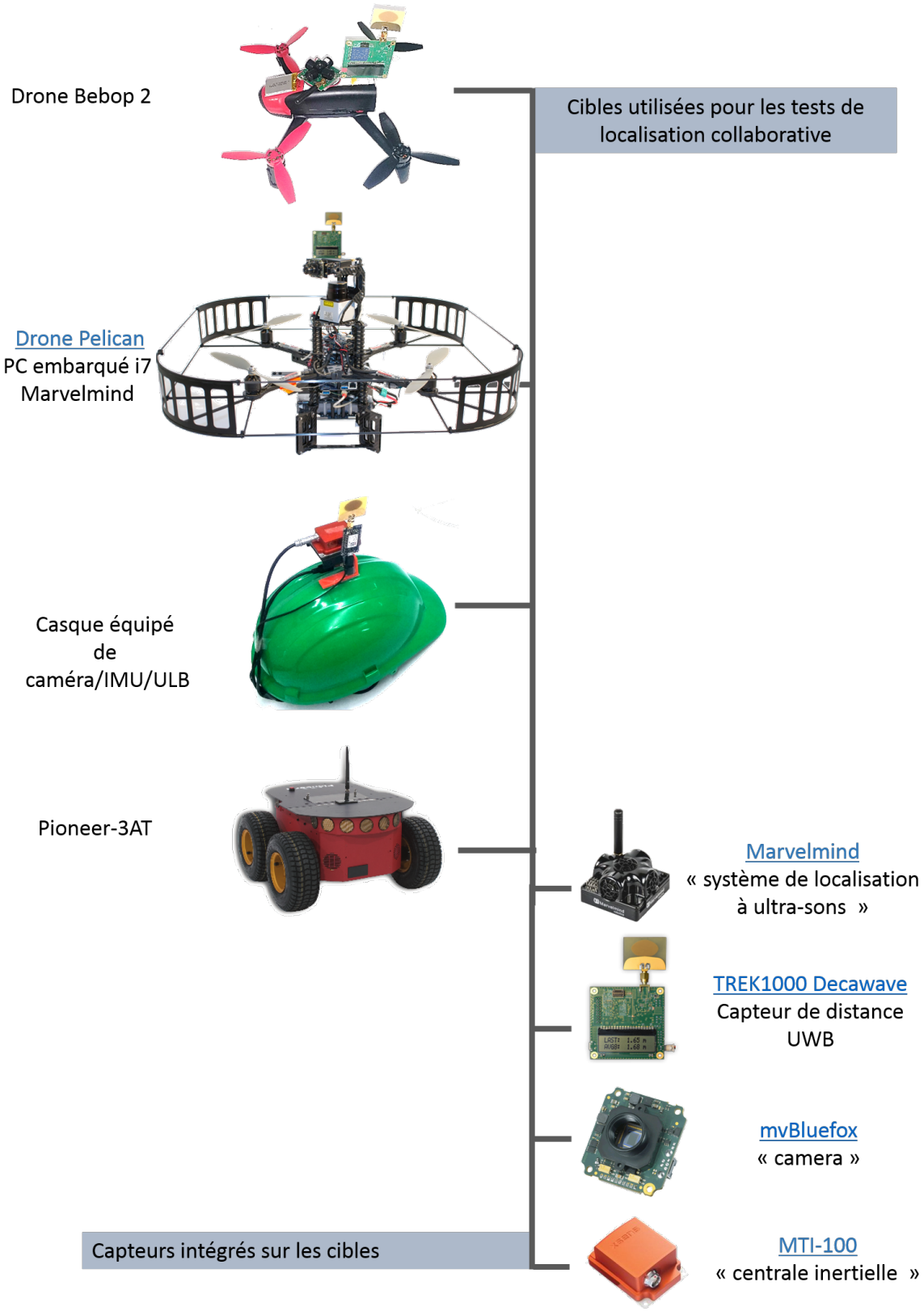


FIGURE 3.17 – Les robots et les capteurs utilisés dans les expériences collaborative.

vont contenir des dérives comme dans le cas du robot mobile où on peut voir la différence entre l'estimation et la réalité terrain Fig.(3.18). L'évolution de la position des Bebops est donnée en utilisant la meilleure estimation de la transformation entre les origines Bebop/Pioneer (après convergence).

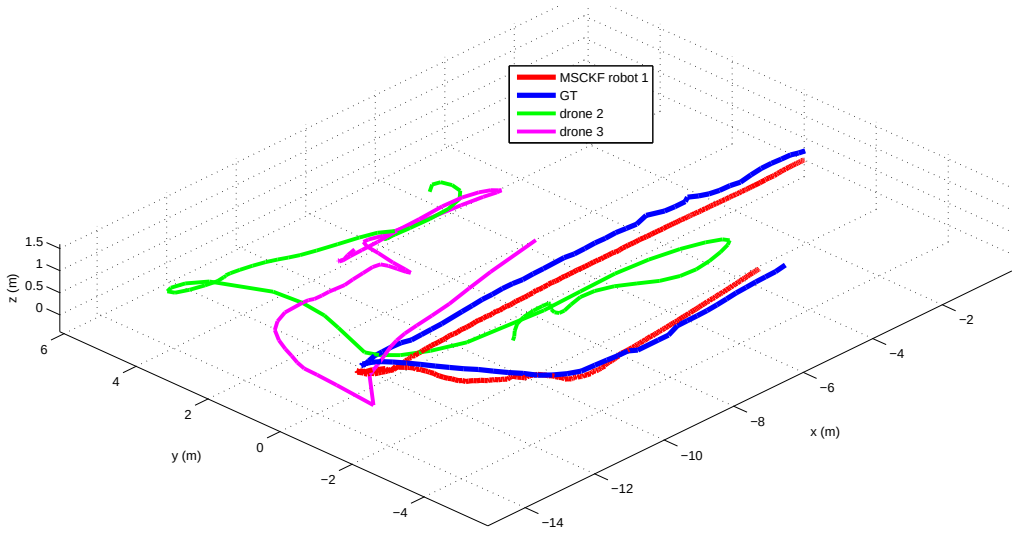
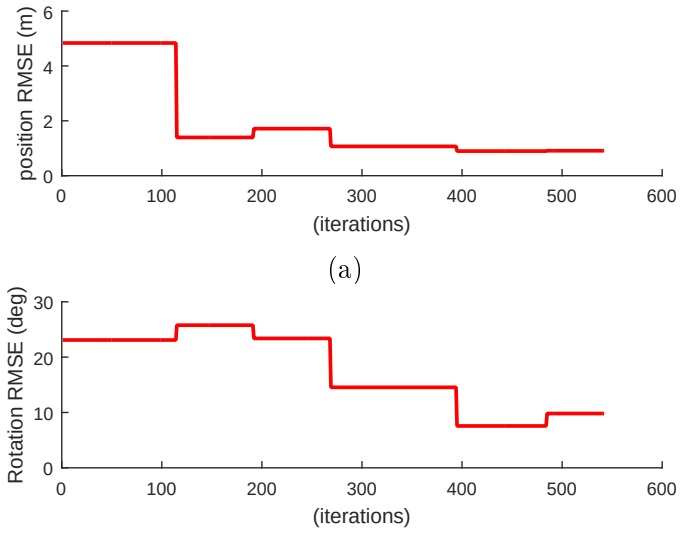


FIGURE 3.18 – Localisation collaborative basée sur les mesures de distance entre un robot mobile et deux drones. Les trajectoires estimées sont données en *rouge* pour le robot terrestre, obtenue par la partie d'odométrie individuelle du filtre, et en *vert* et *magenta* pour les drones obtenues par localisation collaborative. La trajectoire *bleu* représente la réalité terrain de la position du robot mobile terrestre obtenue par le système Marvelmind.



(a)

(b)

FIGURE 3.19 – RMSE de l'estimation de la transformation ((a)translation et (b) rotation) entre les origines des repères. Les déclenchements de la localisation collaborative basée ULB correspondent aux sauts sur les courbes, les plats représentent les intervalles d'acquisition des données.

Dans la Fig.(3.19), on peut voir que le filtre arrive à estimer la transformation entre les origines des repères avec une précision inférieure à 0.8m en position et 7° en rotation.

La Fig.3.19 montre une convergence du filtre. Cependant la convergence contient à certains moments une petite hausse de l'erreur d'estimation. Cette hausse est liée à l'utilisation de mesures de distance contenant des erreurs importantes. Cela se produit lorsque les drones effectuent des manœuvres agressives engendrant un roulis, tangage ou lacet très important des antennes ULB embarquées (voir Annexe.A pour plus d'information sur ces erreurs).

La précision obtenue en orientation et en position, permet d'avoir une indication très importante de la probabilité que deux robots observent la même scène. Cette information est utile pour le déclenchement de la mise à jour basée chevauchement d'images, détaillée dans les étapes suivantes.

Pour mieux tester cette partie basée ULB, nous avons effectué deux autres tests, le premier montre l'impact de la taille de la fenêtre de mesures de distance w et le second, pour tester l'impact des contraintes d'environnement sur la convergence du filtre.

3.7.2.1 Évaluation de l'impact de la fenêtre de mesure

Dans ce scénario, nous faisons varier l'application du filtre collaboratif sur une fenêtre de mesure de une à six mesures consécutives de distances relatives.

Dans la Fig.3.20, on peut voir que le choix de la taille de la fenêtre w, qui correspond au nombre de mesures de distance formant une équation de mesure, affecte la convergence du filtre. Lorsqu'une seule mesure $w = 1$ est utilisée pour construire une équation de mesure, la convergence est très lente. De 2 à 6 mesures, le filtre converge exactement à la même valeur et au même moment. À partir de cette observation, nous pouvons affirmer que l'estimation de la transformation avec une petite fenêtre de mesures ne conduit qu'à un temps de calcul plus élevé, et devrait donc être évitée pour une meilleure efficacité de l'algorithme.

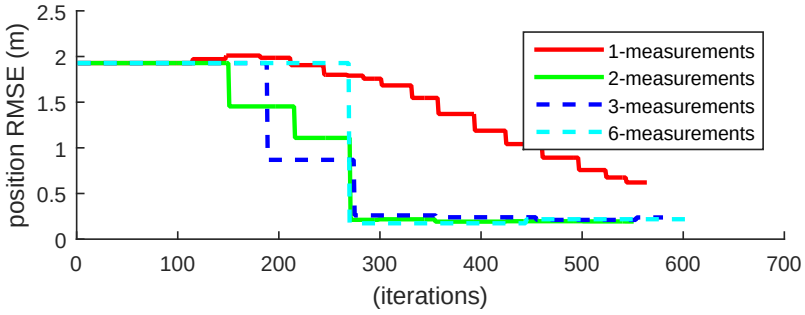


FIGURE 3.20 – Évolution de l'erreur d'estimation de la transformation recherchée en fonction de la taille de la fenêtre de données.

À noter que, avec une seule mesure de distance, la transformation entre les repères

origines des robots n'est pas observable. Ceci explique la difficulté du filtre à converger lorsque $w = 1$. Notons que même avec $w = 1$, le filtre finit par tendre vers la valeur de la transformation ; ceci est le résultat de la mémoire du filtre.

Dans la pratique, le choix de la fenêtre w dépend de plusieurs facteurs comme :

— *La précision du capteur :* si le capteur de distance est précis, il permettra de prendre des mesures avec une fréquence élevée sans avoir besoin de mettre une condition de déplacement minimal à valeur importante. Le mouvement du robot (l'espacement entre deux mesures) doit être à minimum égal à 3 fois l'erreur du capteur. Dans notre cas, le capteur UWB TREK1000 de Decawave permet d'avoir des estimations avec 20cm d'erreur. Un problème rencontré dans cette expérience est que les balises portées par les Bebop sont sensibles aux variations de l'assiette (tangage, roulis, lacet) effectuées par les drones pour se déplacer. Cela induit une augmentation de l'erreur qui peut facilement atteindre 40cm en cas de manœuvres agressives des drones (Annexe.A).

— *Le besoin :* il y a une différence entre l'utilisation des mesures de distance pour localiser les voisins et leur utilisation pour initialiser le filtre pour une future collaboration basée caméra. Si l'objectif est de localiser les robots voisins en n'utilisant que des mesures de distance, l'idéal serait de prendre une fenêtre "w" supérieure à six avec un espacement "physique et/ou temporel" significatif entre les mesures. Si par contre l'objectif est d'initialiser le filtre pour une collaboration basée-caméra, la plage de temps précédant un chevauchement d'images peut contraindre le nombre de mesures possibles à effectuer et donc le temps ou l'espacement entre les mesures ULB.

3.7.2.2 Évaluation de l'impact des contraintes d'environnement

Dans ce test, on utilise le casque Fig.3.17 équipé de caméra/IMU/ULB pour localiser la position d'un autre agent, présent dans la même pièce. Le MSCKF collaboratif exécuté sur le casque effectuera une extraction et une intégration des contraintes d'environnement dans les estimations de la transformation entre les repères origine des agents.

La Fig.3.21 montre l'impact des contraintes d'environnement sur la convergence du filtre. Dans ce test, le MSCKF collaboratif a réussi à extraire trois contraintes en utilisant les lignes de fuite et les points triangulés. L'estimation de la pose du repère origine de l'agent voisin est évaluée avec et sans inclure les contraintes d'environnement. On peut voir que la covariance de position du repère origine du robot voisin est grandement améliorée en intégrant les contraintes d'environnement.

FIGURE 3.21 – Estimation de la transformation entre robots avec covariance correspondante (ellipsoïde *vert*). Avec *(droite)* et sans *(gauche)* prise en compte des contraintes. Les images sont données au même moment dans l'évolution.

3.7.2.3 Application au géo-référencement d'intérieur

Pour conclure les tests de localisation collaborative basée mesures de distance ULB, nous exploitons notre algorithme dans un scénario de géo-référencement d'un environnement intérieur sans GPS. L'idée est de faire collaborer des robots à l'intérieur d'un bâtiment avec des robots à l'extérieur, qui eux ont accès au GPS. Le but est de fusionner la carte extérieure positionnée par GPS à la carte basée points d'intérêt que nous envoie le robot navigant à l'intérieur.

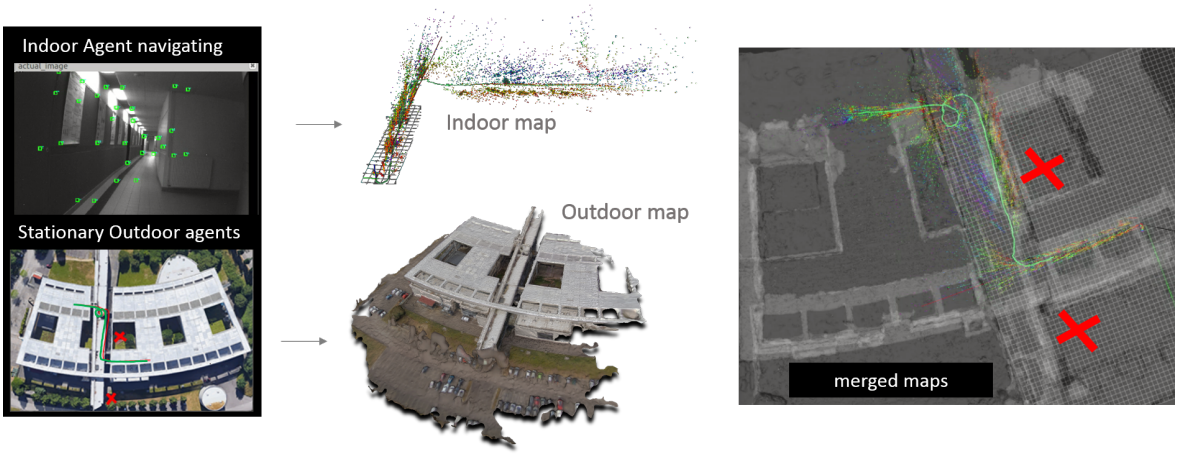


FIGURE 3.22 – Utilisation de la collaboration basée-ULB pour réaliser un géo-référencement. Les croix en *rouge* sont les positions de vol stabilisé des drones Bebop.

Dans ce scénario, deux drones Bebop2 construisent une modèle 3D de l'extérieur d'un bâtiment en utilisant PIX4D, qui est un logiciel qui permet de construire une cartographie en utilisant des images avec leurs positions GPS (voir Annexe.C). Les drones

se positionnent en vol stationnaire dans la carte réalisée et utilisent les balises ULB pour avoir des mesures de distance au robot navigant à l'intérieur. L'un des avantages de l'ULB est que ses propriétés physiques nous permettront d'avoir des mesures de distance même à travers les murs. On utilisera le détecteur de NLOS (Sec.3.4.2) pour ne conserver que les mesures qui n'ont pas un NLOS sévère. L'idée ici n'est pas de réaliser une mission de drones en autonomie totale. Les drones à l'extérieur peuvent être commandés pour se déplacer vers des endroits où il y a le plus de chance d'avoir des mesures directes vers le robot de l'intérieur. Ceci est tout à fait crédible dans une situation de désastre par exemple.

La Fig.3.22 montre les deux cartes à fusionner, le résultat de la fusion des cartes et la reconstruction de la trajectoire du robot intérieur suite à la localisation collaborative.

3.7.3 Évaluation de la localisation collaborative basée-vision en conditions réelles

Dans cette partie, nous utilisons deux robots, un robot mobile terrestre (Pioneer-3AT) et un drone Pelican de Ascending Technologies qui embarque une carte Mastermind i7 Fig.3.23. Alors que le drone réalise une trajectoire dynamique, il utilise d'abord les mesures de distance pour initialiser l'estimation de la transformation entre les deux robots. Une fois qu'un chevauchement est détecté, une mise à jour collaborative basée sur la caméra est effectuée.

FIGURE 3.23 – Environnement de test de la localisation collaborative basée-vision.

Dans les Fig.(3.24)(3.25) (qui représentent le test 1), on peut voir que la mise à jour basée caméra est précédée de cinq mises à jour basées ULB (avant l'itération 470) pour initialiser la transformation à estimer. Cette étape permet au filtre d'éliminer tout problème lié à l'initialisation car elle ramène l'estimation à une erreur inférieure à 1m en position et à 10° en rotation. La mise à jour basée caméra qui arrive à l'itération 470 ramène l'erreur à 3° en rotation et à 0.3m en position.

Les Fig.(3.26)(3.27) sont les résultats du test 2. Dans ce test, seule la position est

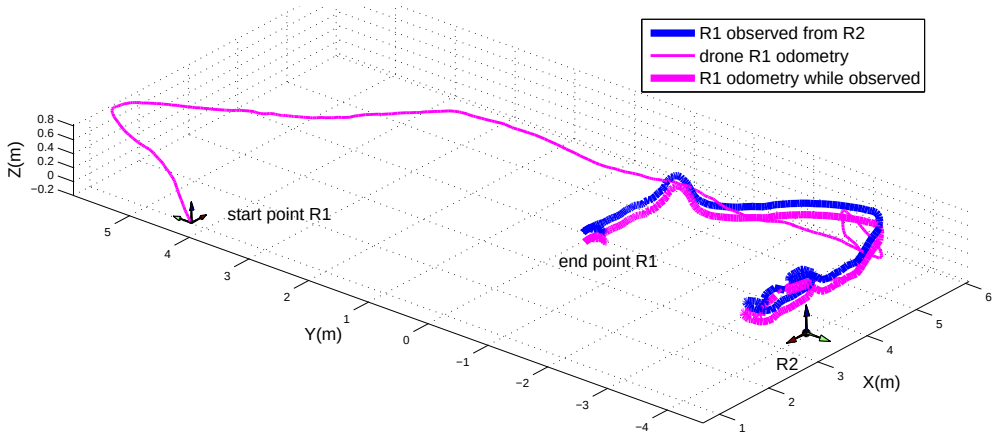
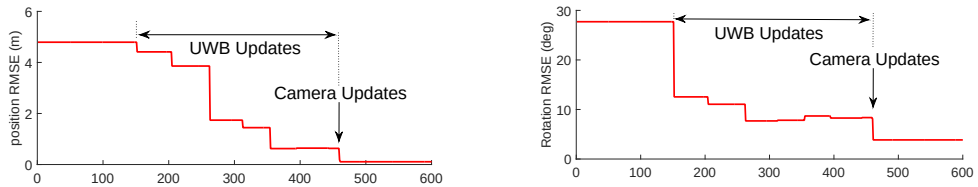


FIGURE 3.24 – Test 1. Localisation collaborative en utilisant le drone Pelican (R_1) et le robot mobile terrestre Pioneer (R_2). La trajectoire du drone estimée par le robot au sol est montrée en *bleu*, la réalité terrain de position du drone est en *magenta*.



(a) (b)

FIGURE 3.25 – Test 1. RMSE de la transformation estimée ((a)position et (b) orientation) entre les repères initiaux des robots sol-air.

observable lors des mises à jour collaboratives basées ULB (de 0 à l'itération 650). On peut voir sur la figure de l'erreur de position Fig.(3.27) que la dernière converge. Lorsqu'un chevauchement d'images entre les robots est détecté l'orientation est estimée et converge après quelques mises à jour basées vision.

Ces deux tests nous ont permis de valider la complémentarité de nos deux types de mise à jour (basée ULB et basée vision). La détection des situations de chevauchement d'images est plus efficace grâce aux mises à jour basées ULB, qui donnent une estimation de pose relative proche de la vraie valeur. Ceci permet de réduire le nombre de tentatives

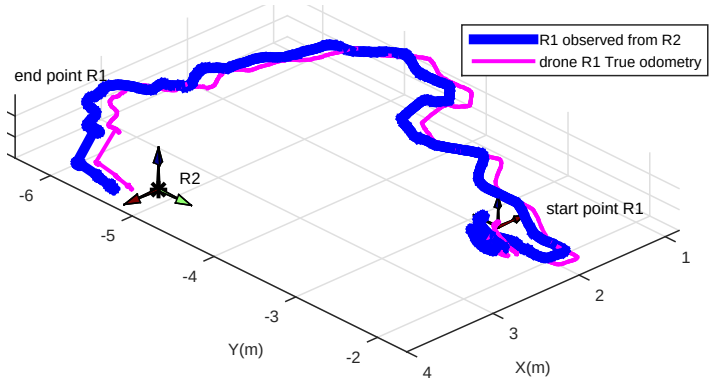
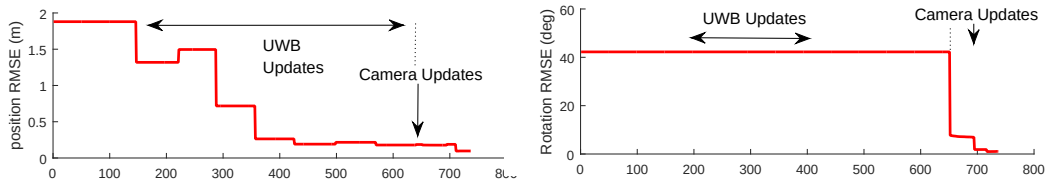


FIGURE 3.26 – Test 2. Localisation collaborative en utilisant le drone Pelican (R_1) et le robot mobile Pioneer (R_2). La trajectoire du drone estimée par le robot au sol est montrée en *bleu*, la réalité terrain de position du drone est en *magenta*.



(a) (b)

FIGURE 3.27 – Test 2. RMSE de la transformation estimée ((a)position et (b) orientation) entre les repères initiaux des robots sol-air.

de mise à jour basées vision, ce qui limite l'impact de la collaboration sur le temps de calcul de l'odométrie individuelle.

3.7.4 Évaluation de l'impact de la collaboration sur le temps de calcul

L'approche MSCKF est généralement réputée pour son efficacité en temps de calcul. Pour évaluer le temps de calcul de l'algorithme proposé, on réalise trois tests de localisation : MSCKF sans collaboration, MSCKF collaboratif basé ULB et un test contenant une collaboration basée ULB et Caméra.

La Fig.3.28 montre le temps de calcul pour chaque cas en utilisant la même séquence qui contient (pour les cas contenant de la collaboration) 7 mises à jour basées ULB suivies de trois mises à jour successives basées caméra. Les 100 premières itérations sont dédiées à l'initialisation de l'algorithme (l'initialisation de l'algorithme de collaboration basée caméra est plus longue en temps de calcul car elle contient plus de blocs à préparer). Nous observons, au cours des 450 itérations qui suivent l'initialisation, qu'il n'est pas possible de différencier l'impact des sept mises à jour basées ULB réalisées sur cet espace

de temps. Ce qui signifie que les mises à jour basées ULB n'affectent pas le temps de calcul du filtre proposé.
Cependant, il est très facile de détecter les pics du temps de calcul des trois mises à jour successives basées sur le chevauchement d'images et qui interviennent après les 450 itérations des mises à jour basées ULB. Ces pics de 25 ms correspondent au temps nécessaire pour les mises à jour basées caméra en utilisant l'équation de mesure 3.7. Ce que l'on ne voit pas par contre est le temps nécessaire à la préparation des données pour ces mises à jour basées caméra (environ 60ms pour l'extraction et la mise en correspondance des points sur 4 images et à chacune des trois mises à jour). La raison est que le calcul étendu a absorbé cette surcharge de temps de calcul dans la fenêtre MSCKF, ce qui a limité l'impact de cette collaboration sur le temps de calcul global au minimum.

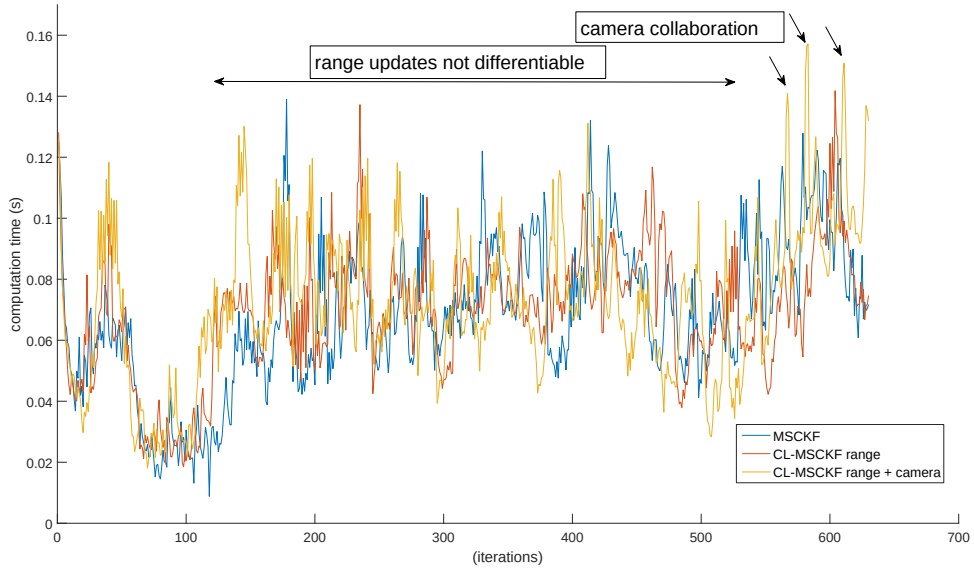


FIGURE 3.28 – Évaluation du temps de calcul du filtre MSCKF individuel, collaboratif basé ULB et collaboratif basée ULB+caméra.

3.8 Conclusion

Dans ce chapitre, une architecture de localisation collaborative de robots hétérogènes a été présentée. En profitant des avantages de la structure MSCKF et des connaissances sur son comportement acquises dans le chapitre précédent, on a réalisé un MSCKF collaboratif qui fonctionne sur trois étages. L'architecture proposée permet une localisation collaborative sans le besoin d'une puissance de calcul externe.
Contrairement à d'autres approches de localisation collaborative qui estiment la pose actuelle des robots voisins, notre approche recouvre les repères initiaux des robots en

collaboration car ces derniers ont une dynamique lente, qui est supposée être constante mais qui en réalité dérive lentement pour compenser les erreurs d'odométries. Cette caractéristique rend ces grandeurs un choix idéal pour le MSCKF multi-robots à basse fréquence proposé dans ce travail.
Le premier étage de l'estimation hiérarchique proposée est la localisation MSCKF individuelle qui a la priorité la plus élevée dans l'architecture proposée pour conserver les propriétés MSCKF d'origine. Le second étage est une mise à jour basée sur les mesures ULB qui inclut des contraintes d'inégalité sur l'état que l'on extrait à partir de l'environnement en combinant le nuage de points triangulés dans le premier étage avec une extraction de lignes de fuite robuste. Cette mise à jour est effectuée à l'aide d'une transformée unscented tronquée TUKF pour inclure les contraintes d'inégalité à la fois dans l'état et dans la matrice de covariance.
Le troisième étage est une mise à jour basée sur le chevauchement des données des caméras que l'on effectue de manière à réduire son impact au minimum sur la fréquence de la localisation individuelle du premier étage. Cela a été réalisé en effectuant un calcul étendu sur la préparation des données ; une opération possible grâce à la structure du MSCKF qui peut maintenir une longue fenêtre d'historique de poses dans son vecteur d'état.
Pour minimiser d'avantage l'impact des mises à jour multi-robots sur l'odométrie individuelle, les mises à jour inter-robots basées caméras utilisent l'équation de mesure basée tenseur trifocal étudiée dans le chapitre précédent et qui permet des mises à jour deux fois plus rapides que les mises à jour du MSCKF original.
Les résultats mettent en évidence la pertinence de la localisation collaborative dans différents scénarios utilisant des robots aériens et terrestres dans des environnements intérieurs et extérieurs. Ceci démontre la validité de l'approche proposée avec un grand potentiel d'utilisation dans des applications comme la cartographie, la surveillance d'environnement et l'exploration coordonnée dans les missions de recherche et de sauvetage.

Chapitre 4 :

Conclusion générale

4.1 Conclusion

Le problème de la localisation collaborative est un problème complexe qui a pour objectif d'augmenter la collaboration entre robots et permettre une amélioration de l'estimation des poses individuelles. Il est souvent possible de se placer à l'un des deux niveaux présents dans cette problématique. Dans le premier niveau, on construit une solution qui vient en aval de la localisation individuelle des robots et qui propose d'utiliser les odométries individuelles comme des boîtes noires dans l'architecture collaborative globale. L'autre solution est une solution qui intervient au même niveau que la localisation individuelle en exploitant au maximum les avantages que peut offrir cette immersion. Si on fait le parallèle avec les solutions d'odométrie visuelle inertielle, les solutions disponibles sont une extension de la problématique du couplage lâche ou serré dans un algorithme VINS.
Dans ce travail, on se positionne comme partisan de la solution en couplage serré des algorithmes de localisation et nous étudions toutes les implications de ce choix sur un algorithme de localisation collaborative.
La première implication d'un tel choix est que le type de l'algorithme de localisation individuelle devient d'une grande importance car c'est lui-même qui doit assurer l'intégration de toutes les fonctions nécessaires pour avoir à la fois une bonne odométrie individuelle et collaborative. Suite à ce même choix de partir sur une solution à couplage serré en localisation collaborative, il devient incohérent de partir sur des approches de localisation individuelle à couplage lâche pour fusionner la caméra et l'IMU. Ceci nous laisse le choix entre les algorithmes de type EKF-SLAM et les algorithmes MSCKF. Ces derniers au lieu de maintenir la position des points d'intérêt dans le vecteur d'état, ils maintiennent une fenêtre glissante d'anciennes poses de la caméra. Cette différence est souvent source de débats dans la localisation individuelle, mais nous avons démontré que la fenêtre d'anciennes poses en MSCKF est un avantage indéniable dans la localisation collaborative.

Cas d'un seul robot :

La première étape de notre travail était d'étudier en profondeur les caractéristiques d'un algorithme de localisation visuelle inertielle basé sur une fusion serrée MSCKF dans le contexte d'une localisation individuelle. Dans le chapitre 3 on a montré qu'il est possible d'avoir deux types de MSCKF dans un système VINS, l'un basé sur un EKF et l'autre basé sur l'UKF. Ce dernier, au lieu de linéariser, utilise une approche de points Sigma. Ceci permet en théorie à la version UKF d'atteindre une précision d'ordre deux dans le développement de Taylor, ce qui signifie qu'il est théoriquement plus précis qu'un

MSC-EKF. Pour tester la théorie, deux modèles de mesure basés sur la formalisation de contraintes de déplacement des points d'intérêt dans les images ont été utilisés. Une différence essentielle entre ces deux modèles est que l'un permet de mettre des contraintes sur les points triangulés et l'autre directement sur les points sans triangulation. Ceci est possible car l'équation de mesure basée sur un transfert trifocal d'un point entre trois images n'a pas besoin de reconstruire l'information 3d. De plus, cette équation de mesure, parce qu'elle n'a pas besoin de triangulation n'est pas affectée par le problème de la corrélation entre l'erreur de triangulation et l'état. Ce problème est inhérent aux algorithmes de cette catégorie, qui ne maintiennent pas les positions des points dans le vecteur d'état. L'une des remarques les plus visibles de cette partie est la facilité d'implémentation de l'UKF quelle que soit l'équation de mesure, une facilité qui nous a poussé à l'utiliser comme structure de test préférée pour le reste de notre travail. L'autre point remarqué à ce stade de la construction était que la différence en temps de calcul d'un UKF par rapport à l'EKF n'est que de 5 ms dans cette application, ce qui est insignifiant.
Pour tester la qualité d'estimation de l'algorithme MSCKF, on a utilisé deux types de données, 1) la dataset KITTI qui permet d'avoir des données IMU et caméra avec réalité terrain d'un véhicule en déplacement et 2) des données réelles d'un drone Pelican avec calcul embarqué, pour tester la capacité et la précision du filtre dans une tâche de navigation en intérieur avec des déplacements dynamiques et de vol stationnaire.
Dans les tests KITTI, le MSC-UKF combiné avec l'équation de mesure sans triangulation a eu des meilleurs résultats que le MSC-EKF utilisant n'importe quelle équation de mesure. Ce qui nous a clairement montré que pour un cas de mouvement fluide et rapide avec visibilité des points n'excédant pas quatre images, la combinaison MSC-UKF utilisant l'équation de mesure sans triangulation est le choix idéal pour nous comme algorithme de départ.
Les résultats de cette partie du travail ont également montré que l'impact de la corrélation de l'erreur de triangulation des points d'intérêt avec l'état, est plus visible sur l'EKF que sur l'UKF si une telle corrélation existe dans le système.
Pour la navigation de type drone en intérieur, les résultats montrent que l'équation de mesure sans triangulation est à éviter dans les mouvements de drones en particulier en vol stationnaire, où les points d'intérêt triangulés donneront un net avantage à l'équation de mesure de triangulation. Ces derniers agiront en effet comme des informations fixes dans le filtre ce qui réduit la dérive.
Pour tester la robustesse du MSCKF aux perturbations visuelles de type manque de points d'intérêt ou mauvaise répartition de ces derniers dans la scène, on a utilisé les datasets KITTI. On a trouvé que sur toutes les trajectoires, le MSC-UKF avec transfert trifocal l'emporte toujours lorsque le nombre de points est au minimum. Ceci démontre à quel point il est important de choisir, ou au moins de bien comprendre, l'impact d'un

environnement et d'un type de mouvement sur un algorithme de localisation visuelle inertielle.
Pour l'évaluation du temps de calcul, nous avons constaté qu'une différence de 5 ms entre une mise à jour EKF et UKF existe et qu'il y a un gain de 20ms si l'équation de transfert trifocal est utilisée. A la fin de cette partie, nous avions acquis une grande maitrise et compréhension du comportement d'un VINS MSCKF, ce qui nous a permis de choisir la variante MSC-UKF avec triangulation comme algorithme de départ pour l'extension vers le multi-robots.

Cas collaboratif :

Dans la dernière partie de ce travail, nous avons proposé une extension de l'algorithme MSC-UKF proposé dans la partie précédente au cas de la localisation collaborative. Pour ce faire, l'état du filtre est augmenté par la transformation entre les repères origines des robots voisins et du robot local. Ce choix d'estimer cette transformation au lieu d'estimer la pose actuelle des voisins vient du fait que estimer la transformation entre les repères origines revient à estimer une grandeur dont la dynamique d'évolution est beaucoup plus faible que celle de la pose actuelle des voisins. En théorie, cette transformation est censée être fixe mais dérive lentement avec le cumul des erreurs d'odométries des voisins. Cette basse dynamique nous permet d'estimer cette variable avec des intervalles de temps beaucoup plus espacés que si l'on estimait la pose actuelle d'un voisin, où il serait nécessaire de procéder à des mises à jour à haute fréquence qui impacteront sévèrement l'odométrie individuelle. La deuxième raison justifiant ce choix est liée aux résultats de notre étude du cas mono-robot qui ont montré une bonne précision de l'odométrie individuelle à court terme. Cette odométrie du robot voisin combinée avec l'estimation à basse fréquence de la transformation entre les repères d'origines, permet d'avoir une information précise sur la pose actuelle du voisin.

Cas collaboratif ULB :

La localisation collaborative proposée est constituée de deux étages. Le premier est une localisation basée sur les distances relatives ULB entre robots. Cette estimation utilise une fenêtre de mesures de distances entre robots à chaque mise à jour. Cet étage permet d'avoir une première estimation que l'on peut raffiner en utilisant des contraintes extraites de l'environnement intérieur. Pour les extraire, on combine les sorties du MSC-UKF avec les observations robustes de lignes de fuite.
Pour tester cette partie de localisation collaborative basée ULB, on a effectué plusieurs expérimentations démontrant l'impact du nombre de mesures de distances par mise à jour sur la convergence du filtre et l'effet des contraintes d'environnement sur l'estimation. Un

robot mobile terrestre et deux drones sont utilisés dans une expérience où le robot mobile terrestre doit localiser les drones. Suite à ce test, nous avons décidé de faire un test terrain de cette solution pour réaliser un géo référencement d'un agent navigant à l'intérieur d'un bâtiment et qui est localisé par deux drones à l'extérieur du bâtiment équipés de balises ULB et un accès au GPS. Les résultats montrent l'efficacité de notre algorithme à fusionner les deux cartes et à tracker la trajectoire de l'agent avec une bonne précision.

Cas collaboratif Caméra :

L'estimation basée ULB est utilisée comme une initialisation pour une mise à jour basée chevauchement d'images. Dans ce deuxième étage de localisation collaborative, on a fait deux propositions pour une collaboration basée caméra avec un minimum d'impact sur l'odométrie individuelle. La première est une architecture de calcul qui utilise le fait qu'un MSCKF maintient la pose actuelle durant m itérations futures, ce qui permet de faire un calcul étendu de la préparation des données nécessaires pour une mise à jour collaborative. La deuxième proposition est l'utilisation de l'équation de mesure du transfert trifocal dans cette mise à jour, pour intégrer les observations visuelles communes avec les robots voisins. L'étude effectuée dans le cas mono-robot a montré la rapidité et la précision de ce modèle dans ce contexte.

4.2 Perspectives

Les perspectives de ce travail gravitent essentiellement autour de la partie collaborative. En particulier pour la localisation collaborative basée ULB, il serait très intéressant d'explorer la possibilité d'intégrer dans le filtre une partie qui, en plus de la détection des NLOS, corrige les mesures de cette situation. Ceci nous permettrait d'utiliser les mesures en situation de NLOS sévère au lieu de les éliminer. L'autre perspective est de construire une cartographie collaborative de type "matière des éléments construisant l'environnement" en se basant sur les mesures ULB collaboratives. En effet, les capteurs ULB, grâce à leur capacité de vision à travers les obstacles, permettraient dans le cas des multi-robots, en s'appuyant sur l'information d'atténuation du signal, de remonter au type de la matière constituant la zone de réflexion des signaux ULB. Ceci sera un grand complément à notre application de géo-référencement de l'intérieur basé sur les ULB.

Un point d'amélioration de la partie localisation collaborative basée vision, serait de trouver une manière pour passer de l'utilisation du nuage de points triangulés résultant du MSC-UKF à l'utilisation d'une reconstruction plus dense de l'environnement sans impacter le temps de calcul. Ceci permettrait au système une extraction plus robuste des

contraintes d'environnement. L'une des possibilités serait de construction un réseau de neurones capable d'effectuer une reconstruction dense directement à partir du nuage de points déjà triangulés dans le filtre.

L'utilisation de ces contraintes d'environnement pourrait aussi être étendue à la collaboration pour le géo référencement de l'intérieur, ce qui permettrait une convergence plus rapide de la fusion des cartes des robots.

De manière générale, l'intégration de nouveaux modèles de mesure collaboratifs pour d'autres types de données que la pose et la distance relatives, permettra d'atteindre une plus grande tolérance à l'hétérogénéité des robots en collaboration. Ceci permettrait de faciliter le passage à l'échelle et le test sur un grand nombre de robots dans des conditions réelles.

Annexes

A Erreurs de mesure des capteurs Marvelmind et ULB TREK1000

Marvelmind : Ce système est un ensemble de balises Ultrasons à positionner dans l'environnement de travail pour récupérer la réalité terrain d'un mouvement. Pour tester la réalité terrain Marvelmind, une balise est placée sur une plateforme tournante dans différentes positions d'une scène de travail du capteur. Sur la Fig.4.1 on peut constater la précision et la répétabilité de l'estimation de position.

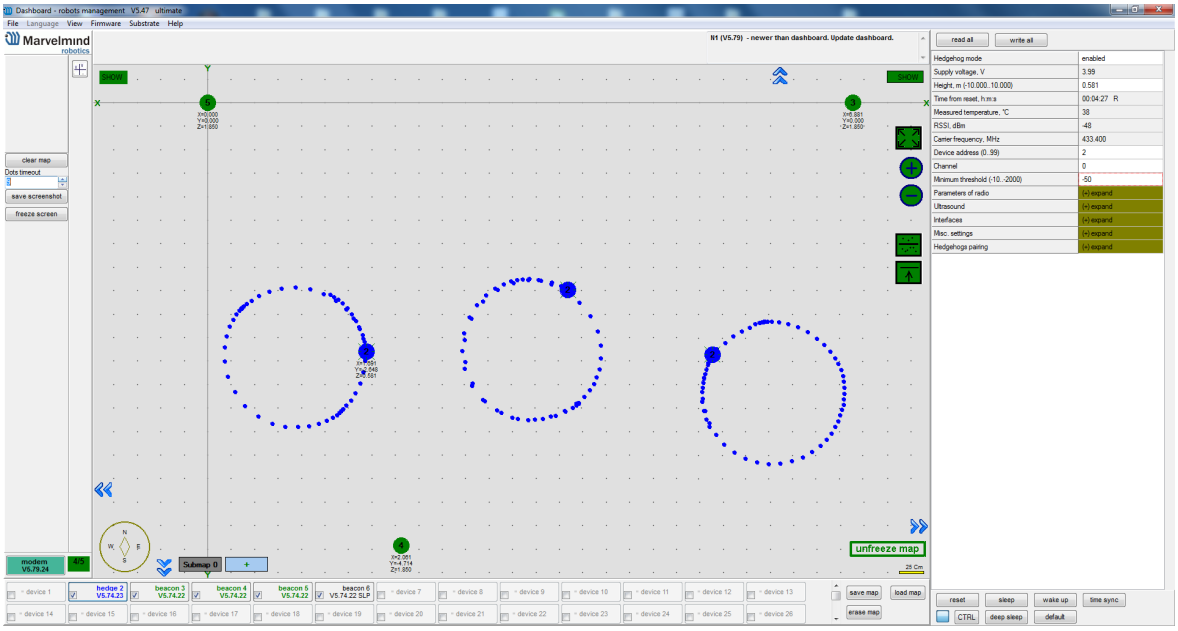


FIGURE 4.1 – Interface du système réalité terrain Marvelmind.

D'après nos expériences, la précision de 2 cm promise par le fabriquant est tout à fait atteignable dans le cas d'une bonne visibilité des balises.

ULB TREK1000 : Ce sont des cartes d'évaluation qui permettent de piloter le circuit intégré DW1000 de Decawave [dec,]. Ce kit permet à l'utilisateur de tester la précision de mesure de distance entre deux balises ULB. Il est basé sur la mesure du temps d'arrivée en ligne de vue et non en ligne de vue. La localisation d'une balise est possible sous trois modes : Tracking,Geo-Fencing et Navigation.
Dans ce Kit, la méthode de mesure de distance utilisée est appelée le "two way ranging". Elle utilise trois messages pour effectuer des mesures à deux tours pour ensuite calculer la distance (Fig.4.2).
Dans l'algorithme d'estimation de distance, il y a deux rôles, une extrémité est désignée pour agir en tant que TAG et l'autre extrémité est désignée pour jouer le rôle d'ANCRE. L'ancre est à l'écoute principalement d'un message du TAG. Une opération de télémétrie

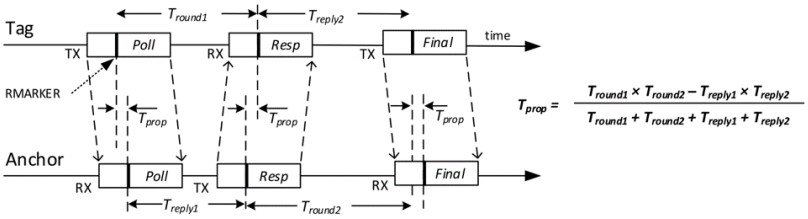


FIGURE 4.2 – Calcul du temps de vol dans un Trek1000.

est déclenchée par l'envoi d'un message de Poll "interrogation" par la balise. Lorsque l'ancre reçoit ce Poll, il envoie un message de réponse. Le TAG reçoit alors le message de réponse et envoie un message final avec des horodatages d'émission et de réception intégrés pour le Poll, la réponse et le message final lui-même (Fig.4.2).
L'ancre utilise cette information avec ses propres horodatages d'émission et de réception pour calculer le temps de vol d'un saut, qui équivaut à la distance multipliée par la vitesse de la lumière dans l'air.
Dans la figure suivante (Fig.4.3), nous avons effectué un test pour estimer les erreurs produites par le système ULB TREK1000. Ce test consiste à mesurer la distance entre deux balises en simulant le tangage et le lacet que ces balises peuvent subir lorsqu'elles sont embarquées sur un drone. La valeur réelle de distance entre les balises est fixe 1,81m. L'objectif est d'évaluer l'impact de l'orientation des antennes sur les estimations de distance.

α_1/α_2	$P1(\psi = 0°)$	$P2(\psi = 40°)$	$P3(\psi = 70°)$	$P4(\psi = 90°)$
0° / 0°	1,89	1,91	1,92	1,92
25° / 0°	1,88	1,97	1,94	1,95
40° / 0°	1,97	2,04	2,01	2,00
40° / 10°	2,00	2,06	2,03	1,94

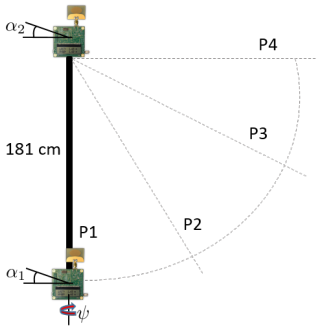


FIGURE 4.3 – Estimation de l'erreur du système ULB dans différentes configurations.

On peut voir dans Fig.4.3 que l'erreur à cette distance et dans des conditions de ligne de vue avec un tangage et un lacet à zero est de 10cm en moyenne. Cependant, malgré que l'antenne utilisée soit omnidirectionnelle on peut voir (en rouge) que pour certaines configurations l'erreur passe à 26cm. Cette erreur est obtenue en ligne de vue et à une distance faible. Ceci est équivalent à la situation où deux drones se trouvent à 1,81m de distance, en effectuant des manœuvres un peu agressives. Avec plus d'interférence, cette valeur d'erreur peut augmenter significativement.

B Triangulation des points d'intérêt

Pour remonter à la position 3D d'un point d'intérêt à partir de sa projection sur une succession d'images, le problème sera formalisé comme une minimisation à résoudre avec l'algorithme de Gauss-Newton. Ce dernier pour résoudre un système non linéaire quadratique, applique une fonction d'erreur f de la forme :

$$f_i(\theta) = z_i - h(\theta) \tag{4.1}$$

Avec θ les paramètres, z_i les mesures et h la fonction de projection des paramètres dans l'espace des mesures.
Un point 3D ${}^{G}p_f$ traqué sur une fenêtre n aura la position ${}^{C_i}p_f = ({}^{Ci}X, {}^{Ci}Y, {}^{Ci}Z)^T$ dans le repère de la camera C_i. La mesure est la position en pixel (u_i, v_i) donc $z_i = h({}^{Ci}X, {}^{Ci}Y, {}^{Ci}Z)$

$${}^{C_i}p_f = {}^{C_i}R_{C_0}({}^{C_0}p_f - {}^{C_0}p_{C_i}) \tag{4.2}$$

$${}^{C_i}p_f = {}^{C_i}R_{C_0}\,{}^{C_0}p_f + {}^{C_i}p_{C_0} \tag{4.3}$$

Avec C_0 la première observation du point d'intérêt. En utilisant la "inverse depth parametrization" qui améliore la stabilité numérique et permet d'éviter les minimas locaux, on peut reformuler ${}^{C_i}p_f$ comme :

$$\begin{aligned}
{}^{C_i}p_f &= {}^{C_i}R_{C_0} \begin{bmatrix} {}^{C_0}X \\ {}^{C_0}Y \\ {}^{C_0}Z \end{bmatrix} + {}^{C_i}p_{C_0} \\
&= {}^{C_0}Z \left({}^{C_i}R_{C_0} \begin{bmatrix} {}^{C_0}X/{}^{C_0}Z \\ {}^{C_0}Y/{}^{C_0}Z \\ 1 \end{bmatrix} + \frac{1}{{}^{C_0}Z}\, {}^{C_i}p_{C_0} \right) \\
&= {}^{C_0}Z \left({}^{C_i}R_{C_0} \begin{bmatrix} \alpha \\ \beta \\ 1 \end{bmatrix} + \rho\, {}^{C_i}p_{C_0} \right) \\
&= {}^{C_0}Z \;\; \mathbf{g}_i \begin{pmatrix} \alpha \\ \beta \\ \rho \end{pmatrix}
\end{aligned} \tag{4.4}$$

Où $\mathbf{g}_i$ est une fonction de α, β et ρ, avec :

$$\alpha = \frac{{}^{C_0}X}{{}^{C_0}Z} \quad \beta = \frac{{}^{C_0}Y}{{}^{C_0}Z} \quad \rho = \frac{1}{{}^{C_0}Z}$$

On peut alors écrire la fonction d'erreur comme :

$$f_i(\theta) = z_i - h(\mathbf{g}_i(\theta)) \quad avec \quad \theta = \begin{pmatrix} \alpha \\ \beta \\ \rho \end{pmatrix} \tag{4.5}$$

Minimisation Gauss-Newton : cette méthode est utilisée pour la résolution de problèmes de moindres carrés non linéaires.

$$S(\theta) = \sum_{i=1}^{n} f_i(\theta)^2 \tag{4.6}$$

En utilisant une solution initiale $\theta^{(0)}$ et la jacobienne $\mathbf{J}_f$ de $\mathbf{f}$, les paramètres peuvent être mis à jour itérativement comme suit :

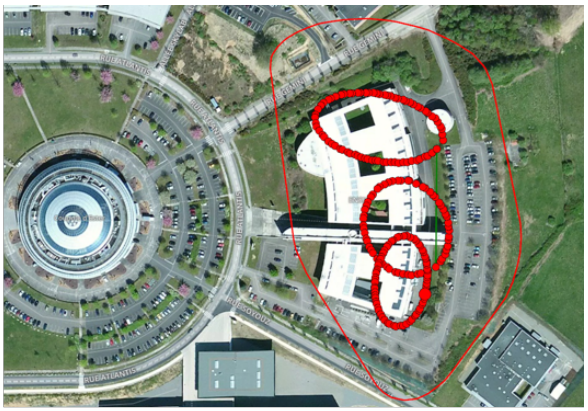
$$\theta^{(s+1)} = \theta^{(s)} - (\mathbf{J}_f^T \mathbf{J}_f)^{-1} \mathbf{J}_f^T f(\theta^{(s)}) \tag{4.7}$$

Ce qui permet d'avoir la position finale du point avec :

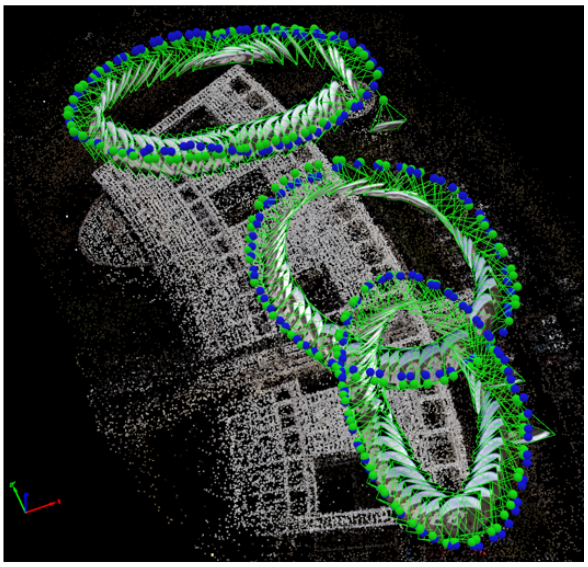
$${}^{C_i}\hat{p}_f = \frac{1}{\hat{\rho}} \, {}^{G}R_{C_0} \begin{bmatrix} \hat{\alpha} \\ \hat{\beta} \\ 1 \end{bmatrix} + {}^{G}p_{C_0} \tag{4.8}$$

C PIX4D

Acquisition des images, réalisée par les drones Bebop.



Environnement ''rayCloud''de PIX4D liant chaque point triangulé aux images correspondantes (permet la détection des sources d'erreurs de reconstruction).



Modèle 3D obtenu après création des surfaces pour améliorer la planéité et combler les trous dans les zones critiques.

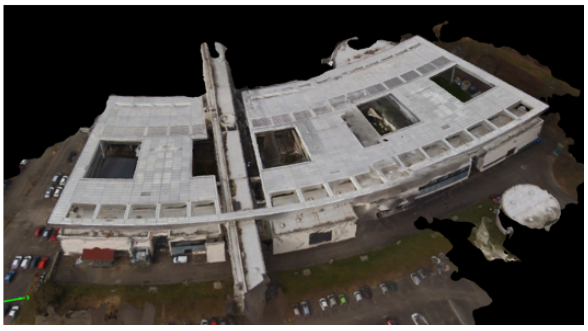


FIGURE 4.4 – Réalisation du modèle 3D, utilisé dans la partie localisation collaborative : Application au géo-référencement d'intérieur.

Bibliographie

[dec,] decawave. http://www.decawave.com/products/dwm1000-module. Accessed : 2018-08-30.

[Achtelik et al., 2011] Achtelik, M. W., Weiss, S., Chli, M., Dellaerty, F., and Siegwart, R. (2011). Collaborative stereo. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2242–2248.

[Al-Jazzar et al., 2011] Al-Jazzar, S. O., Muchkaev, A., Al-Nimrat, A., and Smadi, M. (2011). Low complexity and high accuracy angle of arrival estimation using eigenvalue decomposition with extension to 2d aoa and power estimation. *EURASIP Journal on Wireless Communications and Networking*, 2011(1) :123.

[Badino and Kanade, 2011] Badino, H. and Kanade, T. (2011). A head-wearable short-baseline stereo system for the simultaneous estimation of structure and motion. In *MVA*.

[Bai et al., 2017] Bai, Y., Hu, H., Li, Y., Zhao, C., Luo, L., and Wang, R. (2017). Research Methods for Human Activity Space Based on Vicon Motion Capture System. In *2017 5th International Conference on Enterprise Systems (ES)*, pages 202–206.

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf : Speeded up robust features. volume 3951, pages 404–417.

[C. Tomasi and J. Shi, 1994] C. Tomasi and J. Shi (1994). good features to track. *in Proc. CVPR*, pages pp.593–600.

[Calonder et al., 2010] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief : Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer.

[Caruso et al., 2015] Caruso, D., Engel, J., and Cremers, D. (2015). Large-scale direct SLAM for omnidirectional cameras. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 141–148.

[Civera et al., 2010] Civera, J., Grasa, O. G., Davison, A. J., and Montiel, J. (2010). 1-point ransac for extended kalman filtering : Application to real-time structure from motion and visual odometry. *Journal of Field Robotics*, 27(5) :609–631.

[Clement et al., 2015] Clement, L. E., Peretroukhin, V., Lambert, J., and Kelly, J. (2015). The battle for filter supremacy : A comparative study of the multi-state constraint kalman filter and the sliding window filter. In *Computer and Robot Vision (CRV), 2015 12th Conference on*, pages 23–30. IEEE.

[Coppola et al., 2016] Coppola, M., McGuire, K., Scheper, K. Y. W., and de Croon, G. C. H. E. (2016). On-board communication-based relative localization for collision avoidance in Micro Air Vehicle teams. *Autonomous Robots*, pages 1–19.

[Cornejo and Nagpal, 2014] Cornejo, A. and Nagpal, R. (2014). Distributed range-based relative localization of robot swarms. In *WAFR*.

[Cui et al., 2015] Cui, J. Q., Phang, S. K., Ang, K. Z. Y., Wang, F., Dong, X., Ke, Y., Lai, S., Li, K., Li, X., Lin, F., Lin, J., Liu, P., Pang, T., Wang, B., Wang, K., Yang, Z., and Chen, B. M. (2015). Drones for cooperative search and rescue in post-disaster situation. In *2015 IEEE 7th International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pages 167–174.

[Dardari et al., 2009] Dardari, D., Conti, A., Ferner, U., Giorgetti, A., and Win, M. Z. (2009). Ranging With Ultrawide Bandwidth Signals in Multipath Environments. *Proceedings of the IEEE*, 97(2) :404–426.

[Davison et al., 2007] Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). Monoslam : Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6).

[Deutsch et al., 2016] Deutsch, I., Liu, M., and Siegwart, R. (2016). A framework for multi-robot pose graph SLAM. In *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 567–572.

[Droeschel and Behnke, 2018] Droeschel, D. and Behnke, S. (2018). Efficient continuous-time slam for 3d lidar-based online mapping. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9.

[Engel and Cremers, 2014] Engel, J. and Cremers, D. (2014). Lsd-slam : Large-scale direct monocular slam. In *In ECCV*.

[Engel et al., 2015] Engel, J., Stückler, J., and Cremers, D. (2015). Large-scale direct slam with stereo cameras. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942.

[Engel et al., 2013] Engel, J., Sturm, J., and Cremers, D. (2013). Semi-dense Visual Odometry for a Monocular Camera. In *2013 IEEE International Conference on Computer Vision*, pages 1449–1456.

[Falanga et al., 2017] Falanga, D., Zanchettin, A., Simovic, A., Delmerico, J. A., and Scaramuzza, D. (2017). Vision-based autonomous quadrotor landing on a moving platform. *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, pages 200–207.

[Fan and Chen, 2016] Fan, H. and Chen, Z. (2016). WiFi based indoor localization with multiple kernel learning. In *2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*, pages 474–477.

[Fink et al., 2012] Fink, A., Lange, J., and Beikirch, H. (2012). Radio-based indoor localization using the eZ430-Chronos platform. In *2012 IEEE 1st International Symposium on Wireless Systems (IDAACS-SWS)*, pages 19–22.

[Forster et al., 2014] Forster, C., Pizzoli, M., and Scaramuzza, D. (2014). SVO : Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22.

[Fuchs, 2010] Fuchs, S. (2010). Multipath Interference Compensation in Time-of-Flight Camera Images. In *2010 20th International Conference on Pattern Recognition*, pages 3583–3586.

[Furgale et al., 2013] Furgale, P., Rehder, J., and Siegwart, R. (2013). Unified temporal and spatial calibration for multi-sensor systems. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1280–1286. IEEE.

[Galvez-López and Tardos, 2012] Galvez-López, D. and Tardos, J. D. (2012). Bags of Binary Words for Fast Place Recognition in Image Sequences. *IEEE Transactions on Robotics*, 28 :1188–1197.

[Geiger et al., 2013] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics : The kitti dataset. *The International Journal of Robotics Research*, 32(11) :1231–1237.

[Geiger et al., 2011] Geiger, A., Ziegler, J., and Stiller, C. (2011). Stereoscan : Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV)*.

[Ghavami et al., 2007] Ghavami, M., Michael, L., and Kohno, R. (2007). *Ultra Wideband Signals and Systems in Communication Engineering*. John Wiley & Sons. Google-Books-ID : vX7eCq8TDp4C.

[Gioi et al., 2010] Gioi, R. G. v., Jakubowicz, J., Morel, J. M., and Randall, G. (2010). LSD : A Fast Line Segment Detector with a False Detection Control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Guo et al., 2014] Guo, C. X., Kottas, D. G., DuToit, R., Ahmed, A., Li, R., and Roumeliotis, S. I. (2014). Efficient visual-inertial navigation using a rolling-shutter camera with inaccurate timestamps. In *Robotics : Science and Systems*. Citeseer.

[Hamzeh and Elnagar, 2015] Hamzeh, O. and Elnagar, A. (2015). A Kinect-based indoor mobile robot localization. In *2015 10th International Symposium on Mechatronics and its Applications (ISMA)*, pages 1–6.

[Harris and Pike, 1987] Harris, C. G. and Pike, J. M. (1987). 3d Positional Integration from Image Sequences. pages 32.1–32.4. Alvey Vision Club.

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

[He and Zhao, 2017] He, Z. and Zhao, L. (2017). The Comparison of Four UAV Path Planning Algorithms Based on Geometry Search Algorithm. In *2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, volume 2, pages 33–36.

[Hu and Chen, 2014] Hu, J.-S. and Chen, M.-Y. (2014). A sliding-window visual-imu odometer based on tri-focal tensor geometry. In *Robotics and automation (ICRA), 2014 IEEE international conference on*, pages 3963–3968. IEEE.

[Huang and al, 2011] Huang and al (2011). Observability-based consistent ekf estimators for multi-robot cooperative localization. *Autonomous Robots*, pages 99–122.

[Huntsberger et al., 2004] Huntsberger, T. L., Trebi-Ollennu, A., Aghazarian, H., Schenker, P. S., Pirjanian, P., and Nayar, H. D. (2004). Distributed Control of Multi-Robot Systems Engaged in Tightly Coupled Tasks. *Autonomous Robots*.

[Javanmardi et al., 2017] Javanmardi, E., Javanmardi, M., Gu, Y., and Kamijo, S. (2017). Autonomous vehicle self-localization based on probabilistic planar surface map and multi-channel LiDAR in urban area. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8.

[Jeon and Lee, 2016] Jeon, S. and Lee, J. (2016). Vehicle routing problem with pickup and delivery of multiple robots for hospital logistics. In *2016 16th International Conference on Control, Automation and Systems (ICCAS)*, pages 1572–1575.

[Karrer et al., 2018] Karrer, M., Schmuck, P., and Chli, M. (2018). Cvi-slam—collaborative visual-inertial slam. *IEEE Robotics and Automation Letters*, 3 :2762–2769.

[Khodjaev et al., 2010] Khodjaev, J., Park, Y., and Malik, A. S. (2010). Survey of NLOS identification and error mitigation problems in UWB-based positioning algorithms for dense environments. *annals of telecommunications*, 65(5-6) :301–311.

[Klein and Murray, 2007] Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '07, pages 1–10, Washington, DC, USA. IEEE Computer Society.

[Knuth and Barooah, 2015] Knuth, J. and Barooah, P. (2015). Distributed collaborative 3d pose estimation of robots from heterogeneous relative measurements : an optimization on manifold approach. *Robotica*, 33 :1507–1535.

[Kok et al., 2015] Kok, M., Hol, J. D., and Schön, T. B. (2015). Indoor Positioning Using Ultrawideband and Inertial Measurements. *IEEE Transactions on Vehicular Technology*, 64(4) :1293–1303.

[Kong et al., 2015] Kong, X., Wu, W., Zhang, L., and Wang, Y. (2015). Tightly-coupled stereo visual-inertial navigation using point and line features. In *Sensors (Basel, Switzerland)*, volume 15, pages 12816–33.

[Krishnan et al., 2007] Krishnan, S., Sharma, P., Guoping, Z., and Woon, O. H. (2007). A UWB based Localization System for Indoor Robot Navigation. In *2007 IEEE International Conference on Ultra-Wideband*, pages 77–82.

[Lee et al., 2009] Lee, D., Son, S., Yang, K., Park, J., and Lee, H. (2009). Sensor fusion localization system for outdoor mobile robot. In *2009 ICCAS-SICE*, pages 1384–1387.

[Lee and Yoon, 2015] Lee, J.-K. and Yoon, K.-J. (2015). Real-time joint estimation of camera orientation and vanishing points. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1866–1874.

[Lee and Chung, 2015] Lee, W. and Chung, W. (2015). Position estimation using multiple low-cost GPS receivers for outdoor mobile robots. In *2015 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 460–461.

[Leitinger et al., 2015] Leitinger, E., Meissner, P., Rüdisser, C., Dumphart, G., and Witrisal, K. (2015). Evaluation of Position-Related Information in Multipath Components for Indoor Positioning. *IEEE Journal on Selected Areas in Communications*, 33(11) :2313–2328.

[Leutenegger et al., 2011] Leutenegger, S., Chli, M., and Siegwart, R. Y. (2011). BRISK : Binary robust invariant scalable keypoints. In *2011 International conference on computer vision*, pages 2548–2555. IEEE.

[Leutenegger et al., 2013] Leutenegger, S., Furgale, P. T., Rabaud, V., Chli, M., Konolige, K., and Siegwart, R. (2013). Keyframe-based visual-inertial slam using nonlinear optimization. In *Robotics : Science and Systems*.

[Li and Mourikis, 2011] Li, M. and Mourikis, A. I. (2011). Consistency of ekf-based visual-inertial odometry. *University of California Riverside, Tech. Rep.*

[Lim et al., 2012] Lim, J.-M., Park, J., Lee, K.-J., Im, H., Cho, Y.-J., and Sung, T. (2012). A precise trajectory estimation method using carrier-smoothed GPS for mobile robots. In *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 344–346.

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

[Lynen et al., 2013] Lynen, S., Achtelik, M. W., Weiss, S., Chli, M., and Siegwart, R. (2013). A robust and modular multi-sensor fusion approach applied to mav navigation. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3923–3929. IEEE.

[Melnyk et al., 2012] Melnyk, I. V., Hesch, J. A., and Roumeliotis, S. I. (2012). Cooperative vision-aided inertial navigation using overlapping views. In *2012 IEEE International Conference on Robotics and Automation*.

[Meng et al., 2016] Meng, C., Guo, B., and Liu, X. (2016). Simultaneous localization and mapping using monocular direct method. In *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1–5.

[Morrison et al., 2016] Morrison, J. G., Gálvez-López, D., and Sibley, G. (2016). Moarslam : Multiple Operator Augmented RSLAM. In *Distributed Autonomous Robotic Systems*, Springer Tracts in Advanced Robotics, pages 119–132. Springer, Tokyo.

[Mourikis and Roumeliotis, 2007] Mourikis, A. I. and Roumeliotis, S. I. (2007). A multi-state constraint kalman filter for vision-aided inertial navigation. In *Robotics and automation, 2007 IEEE international conference on*, pages 3565–3572. IEEE.

[Mur-Artal et al., 2015a] Mur-Artal, R., Montiel, J. M. M., and Tardós, J. D. (2015a). ORB-SLAM : A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5) :1147–1163.

[Mur-Artal et al., 2015b] Mur-Artal, R., Montiel, J. M. M., and Tardós, J. D. (2015b). Orb-SLAM : A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5) :1147–1163.

[Nerurkar et al., 2009a] Nerurkar, E. D., Roumeliotis, S. I., and Martinelli, A. (2009a). Distributed maximum a posteriori estimation for multi-robot cooperative localization. In *2009 IEEE International Conference on Robotics and Automation*, pages 1402–1409.

[Nerurkar et al., 2009b] Nerurkar, E. D., Roumeliotis, S. I., and Martinelli, A. (2009b). Distributed maximum a posteriori estimation for multi-robot cooperative localization. *2009 IEEE International Conference on Robotics and Automation*, pages 1402–1409.

[Nister et al., 2004] Nister, D., Naroditsky, O., and Bergen, J. (2004). Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I.

[Özyesil et al., 2017] Özyesil, O., Voroninski, V., Basri, R., and Singer, A. (2017). A survey on structure from motion. *CoRR*, abs/1701.08493.

[Piasco et al., 2016] Piasco, N., Marzat, J., and Sanfourche, M. (2016). Collaborative localization and formation flying using distributed stereo-vision. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1202–1207.

[Piniés and Tardós, 2007] Piniés, P. and Tardós, J. D. (2007). Scalable slam building conditionally independent local maps. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3466–3471. IEEE.

[Pittet et al., 2008] Pittet, S., Renaudin, V., and Merminod, B. (2008). Uwb and mems based indoor navigation. *Journal of Navigation*, 61.

[Reddy et al., 2011] Reddy, C., Tech, M., and B R, S. (2011). TDOA Computation Using Multicarrier Modulation for Sensor Networks. *Int. J. Comput. Sci. Commun. Netw*, 1.

[Rosten and Drummond, 2006] Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer.

[Rublee et al., 2011] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB : An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision*, pages 2564–2571. IEEE.

[Santoso et al., 2017] Santoso, F., Garratt, M. A., and Anavatti, S. G. (2017). Visual–inertial navigation systems for aerial robotics : Sensor fusion and technology. *IEEE Transactions on Automation Science and Engineering*, 14(1) :260–275.

[Scaramuzza and Fraundorfer, 2011] Scaramuzza, D. and Fraundorfer, F. (2011). Visual Odometry [Tutorial]. *IEEE Robotics Automation Magazine*, 18(4) :80–92.

[Schmuck and Chli, 2017] Schmuck, P. and Chli, M. (2017). Multi-UAV collaborative monocular SLAM. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3863–3870.

[Schöps et al., 2014] Schöps, T., Engel, J., and Cremers, D. (2014). Semi-dense visual odometry for AR on a smartphone. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 145–150.

[Segura et al., 2012] Segura, M., Mut, V., and Sisterna, C. (2012). Ultra wideband indoor navigation system. *Sonar Navigation IET Radar*, 6(5) :402–411.

[Sejong Heo, 2017] Sejong Heo, Jaehyuck Cha, C. G. P. (2017). Monocular visual inertial navigation for mobile robots using uncertainty based triangulation. IFAC.

[Tardif, 2009] Tardif, J. P. (2009). Non-iterative approach for fast and accurate vanishing point detection. In *2009 IEEE 12th International Conference on Computer Vision*.

[Teixeira et al., 2008] Teixeira, B. O. S., Torres, L. A. B., Aguirre, L. A., and Bernstein, D. S. (2008). Unscented filtering for interval-constrained nonlinear systems. In *2008 47th IEEE Conference on Decision and Control*.

[Tews et al., 2004] Tews, A. D., Sukhatme, G. S., and Mataric, M. J. (2004). A multi-robot approach to stealthy navigation in the presence of an observer. In *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04.*

[Toldo and Fusiello,] Toldo, R. and Fusiello, A. Robust Multiple Structures Estimation with J-Linkage. In *Computer Vision – ECCV 2008*. Springer, Berlin, Heidelberg.

[Troiani et al., 2014] Troiani, C., Martinelli, A., Laugier, C., and Scaramuzza, D. (2014). 2-point-based outlier rejection for camera-imu systems with applications to micro aerial vehicles. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5530–5536. IEEE.

[Wang et al., 2017a] Wang, M., Xue, B., Wang, W., and Yang, J. (2017a). The design of multi-user indoor UWB localization system. In *2017 2nd International Conference on Frontiers of Sensors Technologies (ICFST)*, pages 322–326.

[Wang et al., 2017b] Wang, P., Gao, C., Guo, W., Li, M., Zha, F., and Cong, Y. (2017b). Robust and fast mapping based on direct methods. In *2017 2nd International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 592–595.

[Wang et al., 2014] Wang, S., Chen, L., Gu, D., and Hu, H. (2014). Single beacon based multi-robot cooperative localization using Moving Horizon Estimation. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1625–1630.

[Wang et al., 2016] Wang, S., Gu, D., Chen, L., and Hu, H. (2016). Single Beacon-Based Localization With Constraints and Unknown Initial Poses. *IEEE Transactions on Industrial Electronics*, 63 :2229–2241.

[Weiss et al., 2013] Weiss, S., Achtelik, M., Lynen, S., Achtelik, M., Kneip, L., Chli, M., and Siegwart, R. (2013). Monocular vision for long-term micro aerial vehicle state estimation : A compendium. *J. Field Robotics*, 30 :803–831.

[Weiss et al., 2012] Weiss, S., Achtelik, M. W., Chli, M., and Siegwart, R. (2012). Versatile distributed pose estimation and sensor self-calibration for an autonomous mav. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 31–38. IEEE.

[Weiss and Siegwart, 2011] Weiss, S. and Siegwart, R. (2011). Real-time metric state estimation for modular vision-inertial systems. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4531–4537. IEEE.

[Xu et al., 1990] Xu, L., Oja, E., and Kultanen, P. (1990). A new curve detection method : Randomized hough transform (rht). *Pattern Recogn. Lett.*, 11(5) :331–338.

[Yi and Chu-na, 2010] Yi, H. and Chu-na, W. (2010). A new moving sound source localization method based on the time difference of arrival. In *2010 International Conference on Image Analysis and Signal Processing*, pages 118–122.

[Zhang and Koch, 2011] Zhang, L. and Koch, R. (2011). Hand-Held Monocular SLAM Based on Line Segments. In *2011 Irish Machine Vision and Image Processing Conference*, pages 7–14.

[Zou and Tan, 2013] Zou, D. and Tan, P. (2013). Coslam : Collaborative visual slam in dynamic environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35 :354–366.

[Zuliani et al., 2005] Zuliani, M., Kenney, C. S., and Manjunath, B. S. (2005). The multiRANSAC algorithm and its application to detect planar homographies. In *IEEE International Conference on Image Processing 2005*, volume 3, pages III–153.

Titre thèse français

Résumé : résumé français.

Mots clés : mots clés français.

Titre thèse anglais

Abstract : résumé anglais.

Keywords : mots clés anglais.

LABO - UMR CNRS n° xxxx
xxx, rue xxx - 870xx LIMOGES